mi computer

CURSO PRACTICO DEL ORDENADOR PERSONAL, EL MICRO Y EL MINIORDENADOR

TOMO 2

Editorial Delta, S.A.

mi COMPUTER


mi COMPUTER

Director: José Mas Godayol
Director editorial: Gerardo Romero
Jefe de redacción: Pablo Parra
Coordinación editorial: Jaime Mardones
Francisco Martín
Asesor técnico: Jesús Nebra

Redactores y colaboradores: G. Jefferson, R. Ford, S. Tarditti, A. Cuevas
Para la edición inglesa: R. Pawson (editor), D. Tebbutt (consultant editor), C. Cooper (executive editor), D. Whelan (art editor), Bunch Partworks Ltd. (proyecto y realización)

Realización gráfica: Luis F. Balaguer

ISBN: 84-85822-90-0 (tomo 2) 84-85822-82-X (obra completa)
Depósito legal: B. 52/1984
Impresión: Cayfosa, Santa Perpètua de Mogoda (Barcelona)
Impreso en España - Printed in Spain - 1984

mi COMPUTER

VOLUMEN 2

Editorial Delta, S.A.

Billar electrónico

El Pinball Construction Set —un avance notable en el diseño de software— permite diseñar juegos en la pantalla del Apple

Incluso en una industria de desarrollo tan rápido como la de los microordenadores, en la cual, razonablemente, cabe esperar que los adelantos de importancia sean casi una cosa corriente, resulta difícil encontrar un producto que sea radicalmente diferente tanto en concepto como en calidad. Un software que reúne estos requisitos es el PCSA *(Pinball Construction Set)*, de la firma Budgeco. Funciona con un Apple II de 48 Kbytes, con una unidad de disco y una palanca de mando. Este paquete realiza una función aparentemente sencilla. Suministra al usuario la figura de una máquina de «millón» (billar electrónico) vacía, y un menú de 38 tipos diferentes de «mobiliario» que sirven para equiparla según desee el propio jugador. Además, existe una serie de funciones de entre las cuales elegir los «útiles» a emplear.

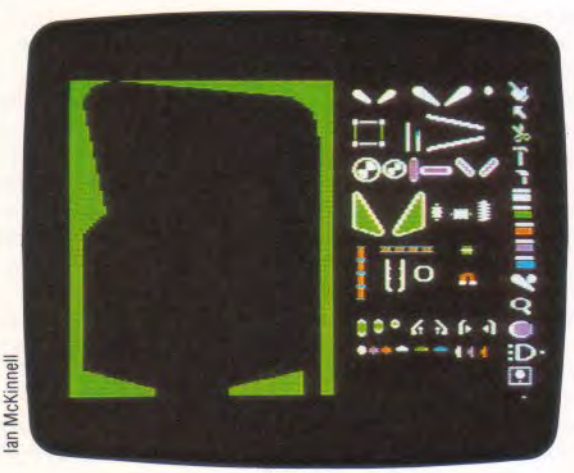
Una vez se haya llenado la mesa de juego según los deseos del usuario —se pueden situar hasta 128 piezas, pero no existe límite en el número de veces que es posible emplear una de cualquier tipo—, ya se puede empezar a jugar. Para ello se selecciona aún otra función con la palanca de mando.

Pueden jugar por turno hasta cuatro personas, pero cada una de ellas sólo puede emplear una bola, en vez de las tres de que se dispone en la mayoría de las máquinas de «millón». Al final del juego, apretando ESCAPE se pueden utilizar otra vez todos los componentes que se desee. Después de cada juego, la forma más interesante de continuar es introduciendo nuevos elementos para hacerlo más complicado.

Por su concepción y realización, el PCS se orienta hacia un software que sea agradable para el usuario. Tan pronto como se carga el programa (lo cual sólo requiere introducir el disco y apretar RETURN), se controla prácticamente toda la acción mediante la palanca de mando. El primer útil a emplear es una mano. Ésta se mueve de forma que apunte a un objeto de los existentes en el menú —que puede ser un parachoques (bumper) o una aleta (flipper)—, y al apretar el botón de la palanca de mando, la mano «recoge» el objeto indicado. Enseguida ésta lo coloca en la posición deseada, y cuando se suelta el botón de la palanca de mando, el objeto queda fijado firmemente en su emplazamiento.

Aquí, lo más interesante es que no sólo se está moviendo la colección de datos que define la forma del objeto, sino también el conjunto de normas que dirigirán la manera de actuar cuando se empiece a jugar. Una aleta, por ejemplo, siempre se mueve 45 grados, primero hacia arriba y luego otra vez hacia abajo. Un parachoques siempre repele la bola, al mismo tiempo que la acelera según un factor de reacción definible. La bola obedece las leyes de Newton sobre movimiento, y cae de acuerdo con la ley de gravedad.

No obstante, además de lo ya enumerado, existe un útil (representado con mucha propiedad por el símbolo de un planeta iluminado parcialmente por el Sol) que permite alterar los parámetros del mundo real: ¡la fuerza de gravedad, por ejemplo, o incluso el tiempo! Esta función es también controlada por la palanca de mando. La posición de cada valor según una escala se altera igual que el mando



Ian McKinnell

Juegos "hágalo usted mismo"
El PCS (Pinball Construction Set) consta de una mesa vacía, diversos tipos de "mobiliario" (parachoques, blancos, aletas, etc.) y, en la columna de la derecha, los útiles para situar los objetos en la mesa. Esta columna contiene también las funciones para ajustar el tamaño, forma, color y grado de influencia recíproca entre las piezas, así como para guardar en disco los juegos una vez completos

The Image Maker

Cosa de niños
El PCS produce incluso sonidos auténticos y el equivalente a luces centelleantes. Pero realmente es más divertido idear y construir los juegos que jugar con ellos. Ahora bien, si tuviera incorporado un mando de FALTA (TILT)...

Ian McKinnell

Paso a paso
Estas figuras muestran varios niveles de la construcción de un juego de "millón". Primero se colocan las piezas básicas y luego se añade un polígono para formar una isla central. El polígono es deformado y pintado de color naranja. Por último, algunos objetos están unidos entre sí (mediante una puerta AND) de forma que se obtiene una bonificación cuando son activados a un tiempo

Listo para funcionar
Una vez se ha compuesto el juego en la mesa, puede guardarse en disco. Debido a que todas las funciones operativas "viajan" con la mesa, no se necesita el paquete de software original para volver a hacer funcionar el programa

Ian McKinnell

de volumen de tipo deslizante de un sistema de audio, desplazándolo hacia arriba o hacia abajo.

Se dispone también de todas las demás funciones que uno espera encontrar en un paquete de gráficos bien desarrollado. Existen "útiles" que sirven para ensanchar y deformar líneas, extendiéndolas entre nodos predeterminados (llamados *bandas elásticas*); otros se emplean para pintar los bloques con uno de los colores de la paleta y para ampliar zonas pequeñas de la imagen.

Sin embargo, la característica más importante del PCS no reside en sus funciones individuales y su versátil capacidad, sino, sobre todo, en su filosofía de funcionamiento. La programación orientada de objetos —en la que cada elemento del paquete de software contiene los detalles de su funcionamiento y de la influencia que ejerce sobre los otros objetos o elementos— lo hace apto para la producción de programas que no necesiten demasiada experiencia o aptitud en informática por parte de los usuarios. Es un hecho que este método de programación será empleado en prácticamente todos los ordenadores de la quinta generación que están en vías de desarrollo. La programación orientada de objetos es considerada de manera unánime como el adelanto más importante en el campo de la ciencia del software, desde que los lenguajes de alto nivel fueron introducidos a fines de los años cincuenta.

La mayor parte de los ordenadores personales tienen una capacidad de memoria suficiente y potencia de procesamiento para cubrir las necesidades del usuario. Toda ampliación de esta capacidad y potencia tiene como finalidad permitir una mayor comodidad en el manejo del ordenador por parte del usuario. Uno de los rasgos más notables del PCS es su habilidad para alcanzar un alto grado de simplicidad funcional con sólo 48 Kbytes.

Si bien la programación orientada de objetos se aplica de forma directa a juegos y otros programas gráficos, ésta requiere algo más de inventiva al programar si se desea aplicarla en el campo del software de gestión. Aunque no empleen los gráficos como medio principal de comunicación, los paquetes de hojas electrónicas son, en cierta manera, objetos en los que cada campo o celda puede contener unos datos y las relaciones que los definen.

Otro ejemplo lo constituye el sistema Lisa, de la firma Apple, que utiliza un "ratón" para desplazar un cursor por la pantalla a fin de seleccionar el programa (representado por un símbolo gráfico) que se desea hacer funcionar. La palabra procesador, por ejemplo, está representada por una hoja de papel en blanco, y el programa de trazado de gráficos, por una hoja de papel cuadriculado.

Quizá la más fascinante de todas sus funciones es el método que usa Lisa para trasladar datos de un programa a otro. Uno de sus *iconos* (representaciones pictóricas de funciones en la pantalla) es una tablilla con clips. Si se quiere tomar una pequeña sección de una hoja electrónica y reproducirla en forma de gráfica, sólo es necesario definir el recuadro sobre la hoja, trasladarlo a la tablilla con clips (que es un área de almacenamiento temporal) y a continuación transportarlo al programa trazador de gráficos.

Cuando se habló de juegos recreativos (véase p. 221), se puso de manifiesto la gran cantidad de juegos de diferente tipo que existían. El PCS podría muy bien entrar a formar parte de un nuevo grupo.

Llegados a este punto, es lógico suponer que el próximo paso que dará la industria de los juegos para ordenadores personales será la producción de este tipo de máquinas de que hablamos, aplicadas a laberintos, invasores del espacio, etc.; a partir de entonces, muchos escritores de programas para juegos podrán considerarse innecesarios.

Chris Stevens

Perspectiva objetiva

Además de ser un juego educativo y fascinante, el Pinball Construction Set es un buen ejemplo de programación orientada de objetos. En programación normal, la estructura de los datos está definida, y las rutinas del programa están escritas para manipularlos. En programación orientada de objetos, los cálculos y procedimientos son inseparables de los datos. En el programa que estamos tratando, mover el símbolo de una aleta (flipper) para situarlo en la mesa, no sólo establece el dato (en este caso la forma de la aleta), sino que dispone las rutinas asociadas de forma que sean capaces de activar la aleta.
La programación orientada de objetos se presta para aplicaciones visuales. Las hojas electrónicas constituyen otro ejemplo: el campo que muestre un resultado contendrá también la fórmula para obtener este resultado. La tendencia actual a la representación de los componentes de un centro de gestión deriva de la misma idea. Indicando en la pantalla la imagen de una hoja de papel en blanco, se activa el procesador de textos, mientras que al hacer referencia a un esquema en miniatura de un gabinete de archivo, se obtiene la clasificación de los resultados

Sistema de control del Cruise

Los controvertidos misiles Cruise, de fabricación norteamericana, poseen una interesante tecnología, como la memoria de burbuja, que pronto incorporarán los ordenadores personales

Cuando Neil Armstrong dio su primer paso sobre la Luna, ello fue posible, en gran medida, gracias a los sistemas de guía computerizados. Sin duda, la cohetería interplanetaria se basa en una ingeniería de alta precisión, pero sin el hardware y software de los ordenadores, no hubiera sido posible realizar los cálculos de posición con la suficiente rapidez y precisión como para permitir a un objeto entrar en contacto con otro a tan gran distancia, incluso con un cuerpo de las dimensiones de la Luna.

Cuando se tienen en cuenta las necesidades militares actuales, que exigen dar en un blanco con un margen de error limitado a 20 o 30 m después de sobrevolar un continente entero, la envergadura de la potencia de procesamiento de datos necesaria para realizar los cálculos resulta enorme.

La experiencia militar anterior mostró que el problema fundamental de los misiles era que, una vez lanzados, no era posible corregir su trayectoria. El primer avance de importancia llegó con el desarrollo de sistemas de guía sencillos que permitían determinar dónde estaba el cohete en relación con un punto de la superficie de la Tierra (el lugar de lanzamiento), deduciendo la distancia recorrida y en qué dirección. Pero incluso un sistema de este tipo adolecería de un margen de error importante.

Otro método de mayor precisión emplea satélites situados en una órbita geoestacionaria como puntos de referencia. El principal inconveniente que presenta este sistema es que la trayectoria de vuelo del misil —y probablemente su blanco— pueden ser determinados por el enemigo al poco tiempo de su lanzamiento, debido a los modernos sistemas de radar. Para combatir esta vulnerabilidad, el requisito ideal era un misil que volara a muy baja cota y dotado de un pequeño radar de sección transversal que pudiera decidir por sí mismo la trayectoria a seguir hasta alcanzar su blanco. Y por ello nació el misil Cruise, que actualiza constantemente su posición, analizando el relieve del terreno sobre el que vuela. Esto se realiza haciendo coincidir una sucesión de lecturas de las alturas sobre el suelo, mediante un altímetro-radar muy preciso, con un mapa de curvas de nivel del terreno almacenado en una memoria de burbuja incorporada.

Este sistema, desarrollado por McDonnell Douglas, se conoce como *TERCOM* (*TERrain COntour Matching*: comparación de la curva de nivel del terreno), o DPW-23. Cada misil almacena en su memoria de burbuja unos 25 "perfiles de trayectoria", que compara con el terreno cuando vuela sobre él. Sin embargo, existen algunos inconvenientes. Por ejemplo, el sistema no se puede utilizar sobre el agua. Tampoco posee una precisión fiable sobre la arena del desierto, en constante movimiento, ni en pleno invierno en los países del norte de Europa, debido a las notables alteraciones del terreno por las prolongadas nevadas.

Cortesía Aerospace Publishing Ltd.

Misil autodirigido
El misil Cruise tierra-tierra de la General Dynamics "Tomahawk" tiene una longitud de 6,40 m y pesa menos de una tonelada y cuarto (1 200 kg). Se dispara desde un tubo montado sobre una base móvil. Su trayectoria empieza como la de un cohete convencional, pero al poco tiempo despliega unas pequeñas alas y se sitúa a pocos metros del suelo. Es propulsado por un compacto y diminuto turborreactor

El Cruise no utiliza este sistema de guía desde el momento de su lanzamiento. Permanece inactivo mientras vuela alto en el espacio aéreo no peligroso. Cuando puede sufrir un ataque desde el aire o tierra, entra en picado hasta situarse a 15 m sobre el suelo, para su vuelo sobre territorio enemigo. Aunque en este punto pueda haberse desviado hasta 1 km de su trayectoria correcta, podrá volver a ella con precisión gracias a uno de sus 25 mapas.

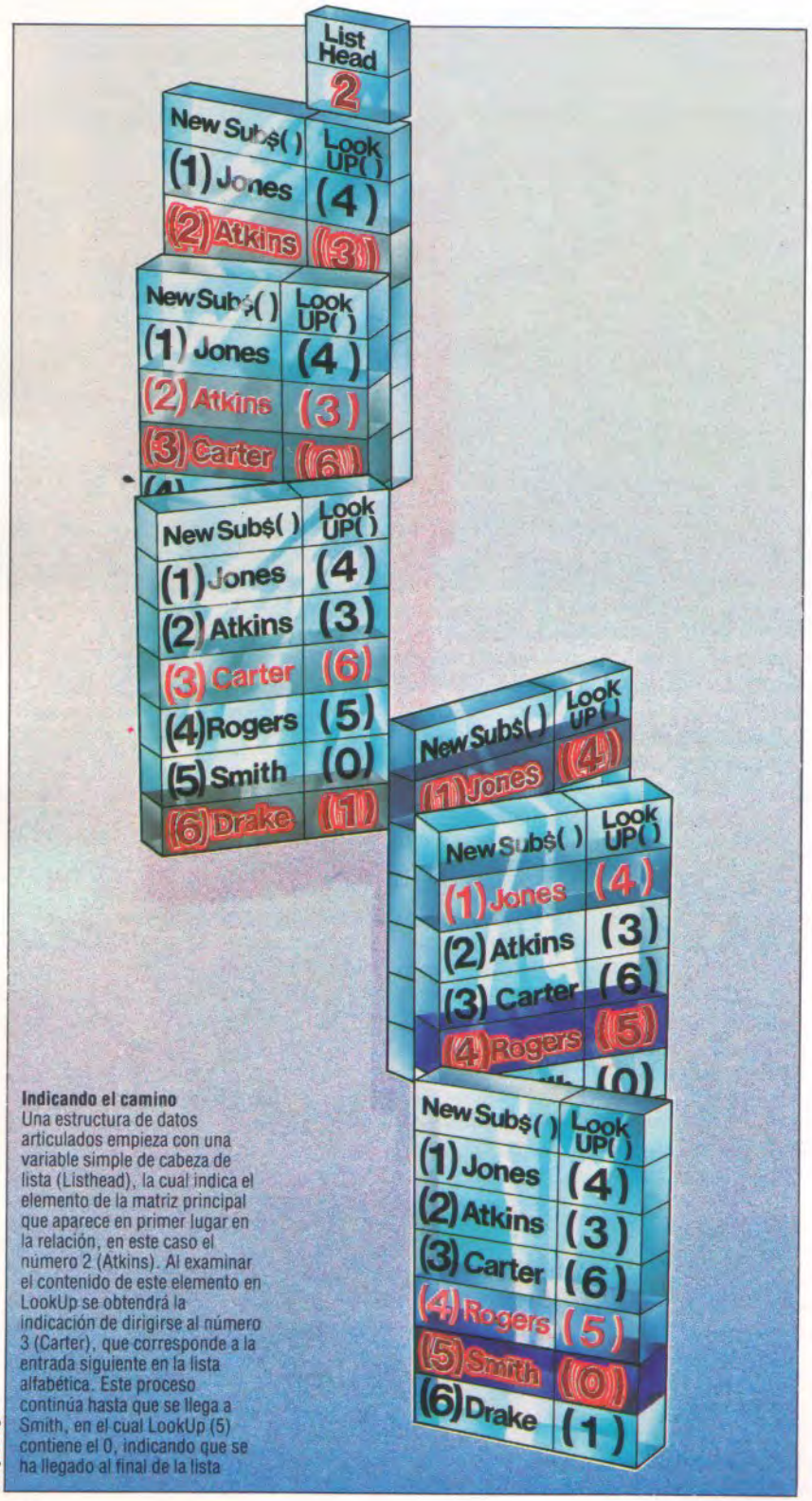
Cuando el misil se encuentra cerca del blanco, conecta una unidad terminal de comparación, que contiene —también en una memoria de burbuja— un grabado digital detallado del área del objetivo. Las pruebas han demostrado que este sistema es capaz de tener una precisión notable, con un margen de error de apenas 18 m de distancia respecto al blanco, tras un vuelo de unos 2 800 km.

Cortesía New Scientist

Memorias de burbuja
En este tipo de memoria se obtiene un "1" cuando se crea una "burbuja" de fuerza magnética, mientras que su ausencia representa el "0", en un diminuto chip de granate. Las ventajas que presenta son su densidad (un millón de bits, o 128 Kbytes por chip) y la no pérdida de contenido cuando se desconecta la fuente de energía. Pero las memorias de burbuja reaccionan más lentamente que las RAM convencionales

Cota de malla

Clasificar es una manera de estructurar grandes cantidades de datos, por ejemplo, nombres y direcciones. La lista articulada o cadena es una forma alternativa que presenta claras ventajas



Indicando el camino
Una estructura de datos articulados empieza con una variable simple de cabeza de lista (Listhead), la cual indica el elemento de la matriz principal que aparece en primer lugar en la relación, en este caso el número 2 (Atkins). Al examinar el contenido de este elemento en LookUp se obtendrá la indicación de dirigirse al número 3 (Carter), que corresponde a la entrada siguiente en la lista alfabética. Este proceso continúa hasta que se llega a Smith, en el cual LookUp (5) contiene el 0, indicando que se ha llegado al final de la lista

Tony Lodge

En la memoria de un ordenador sólo hay datos, byte tras byte de información, almacenados como diferencias de potencial. Estos bytes adquieren su significado por la estructura de los datos que impone el procesador central. Esas diversas formas deciden si un byte determinado debe ser interpretado como componente de una instrucción, o como dígito de un número, o como un código de un carácter.

El usuario del ordenador parte del hecho de que algunos tipos de estructuras están prácticamente incorporados en el ordenador. Por lo general, los lenguajes de programación exigen que los datos sean dispuestos en un número limitado de formas. El lenguaje BASIC impone la idea de tipos numéricos y series de datos, y suministra variables y formas matriciales para manipular estos datos. Otros lenguajes soportan esas estructuras y otras adicionales. La fuerza y variedad de sus tipos de datos son los componentes principales para la determinación de la potencia de un lenguaje. Las estructuras de datos en BASIC —variables y matrices— será lo único que se necesite para simular otras formas de considerar el significado de los datos.

La matriz de clasificación es una estructura de datos práctica, y fácilmente realizada en BASIC. Sin embargo, tiene sus limitaciones, en particular cuando los datos de referencia cambian con frecuencia y/o imprevisiblemente. Supongamos que British Telecom posee un archivo de sus nuevos suscriptores para su inclusión en la próxima edición del listín telefónico. Hasta ese momento, los nombres y direcciones deben mantenerse en orden alfabético, para que puedan ser manejados con facilidad. Pero el archivo crece constantemente, y estos nuevos datos llegan de una manera imposible de predecir; y así, un lunes cualquiera, el archivo New Sub$ ( ) ("nuevos suscriptores") podría tener la siguiente forma:

NewSub$ ( )	Index ( )
(1) Jones	(2)
(2) Atkins	(3)
(3) Carter	(6)
(4) Rogers	(1)
(5) Smith	(4)
(6) Drake	(5)

La matriz Index ( ) muestra el orden en el que leer NewSub$ ( ), de manera que las entradas se encuentren en orden alfabético. Así, el primer suscriptor sería NewSub$ (2), Atkins. El segundo NewSub$ (3), Carter. En este ejemplo sólo se indican los nombres, pero, de hecho, una entrada en el listín comprendería el nombre y dirección —por lo general, unos 60 caracteres—. Desplazar bloques de 60 caracteres a través de la memoria es lento (puesto que su orde-

nación requiere numerosos movimientos de datos); por tanto, es más práctico dejar NewSub$ ( ) sin clasificar, y crear, en su lugar, un Index ( ). Ahora supongamos que se debe añadir un nuevo nombre, Bull, al archivo; la matriz quedará de esta manera:

NewSub$ ( )	Index ( )
(1) Jones	(2)
(2) Atkins	(7)
(3) Carter	(3)
(4) Rogers	(6)
(5) Smith	(1)
(6) Drake	(4)
(7) Bull	(5)

Nótese que el contenido de Index ( ) posterior al nuevo elemento no ha cambiado, y los elementos anteriores están en el mismo orden que antes, pero todos han sido desplazados un lugar. Por tanto, la inserción de un nuevo dato requiere hallar la posición del nuevo elemento, incrementar en una unidad cada elemento comprendido entre aquél y el final del índice, y escribir la nueva entrada. Esto es preferible a hacer lo mismo con los datos reales, NewSub$, pero es aún relativamente lento si el índice es largo.

Supóngase ahora que se estructuran los datos de forma distinta. Se deja sin clasificar NewSub$ ( ), debido a que su manipulación es lenta y cara, y se establece una matriz paralela llamada LookUp ( ), cuyo contenido son simples números que hacen referencia a las posiciones en NewSub$ ( ).

ListHead (2)

NewSub$ ( )	LookUp ( )	Index ( )
(1) Jones	(4)	(2)
(2) Atkins	(3)	(3)
(3) Carter	(6)	(6)
(4) Rogers	(5)	(1)
(5) Smith	(0)	(4)
(6) Drake	(1)	(5)

La primera diferencia es que se necesita una variable simple, denominada ListHead: hace referencia a NewSub$ (2), que es alfabéticamente el primer elemento de NewSub$ ( ). La siguiente diferencia es que se ha utilizado el número (0) en LookUp (5): esto indica que NewSub$ (5) es alfabéticamente el último elemento de la matriz.

Otra diferencia es el contenido de Index ( ) y LookUp ( ). La interpretación de Index ( ) debe ser: "el primer elemento está en NewSub$ (2), el segundo en NewSub$ (3), el tercero en NewSub$ (6)"... etc. Mientras que la de ListHead ( ) sería: "el primer elemento está en NewSub$ (2)". Luego LookUp (2) dice que el próximo elemento está en NewSub$ (3); LookUp (3) indica que el siguiente se encuentra en NewSub$ (6); y así sucesivamente. LookUp (5) informa que NewSub$ (5) es el último elemento.

Index ( ) da una posición absoluta para los elementos del archivo, mientras que LookUp ( ) proporciona sólo posiciones relativas: un componente de LookUp ( ) únicamente dice dónde encontrar el próximo elemento, y no proporciona ninguna información sobre la posición absoluta. El número en Index (4) indica el cuarto componente del archivo ordenado alfabéticamente, mientras que el número en LookUp (4) hace referencia sólo al elemento que aparece tras NewSub$ (4) en el archivo ordenado. LookUp ( ) lleva a cabo la estructura de datos llamada *lista escalonada*. Leer una lista de este tipo es como partir en una expedición "en busca del tesoro": al principio se conoce sólo el primer punto adonde ir. Cuando se ha alcanzado éste, se encuentra una pista que indica el próximo destino, y así sucesivamente. Leer una matriz de clasificación es como participar en un rally de coches: al principio se reciben indicaciones de todos los puntos a los que hay que ir y el orden que hay que seguir.

La gran ventaja de la estructura de lista es su flexibilidad. Véase la lista tras insertar el nuevo elemento, Bull:

ListHead (2)

NewSub$ ( )	Index ( )
(1) Jones	(4)
(2) Atkins	(7)
(3) Carter	(6)
(4) Rogers	(5)
(5) Smith	(0)
(6) Drake	(1)
(7) Bull	(3)

La matriz LookUp ( ) sólo cambia en dos lugares:

a) LookUp (2), que antes hacía referencia a NewSub$ (3) como poseedor del próximo elemento alfabético después de NewSub$ (2), y ahora indica hacia NewSub$ (7), puesto que éste es ahora el siguiente elemento por orden alfabético después de NewSub$ (2)
b) LookUp (7), que no se había utilizado, hace referencia a NewSub$ (3) como próximo componente tras NewSub$ (7) en la ordenación alfabética.

Esto ilustra sobre el proceso general de inserción en una lista escalonada: encontrar el elemento de la lista que debe ir antes del elemento nuevo; luego, hacer que aquél haga referencia a este último, y que el nuevo elemento indique el componente que ha sido desplazado. Estas operaciones sencillas será lo único que se requiera para la inserción en una lista escalonada, y sólo la primera de ellas es afectada por el tamaño de la lista. Insertar un elemento en la lista es como introducir un nuevo eslabón en una cadena: decidir dónde hay que poner el eslabón, abrir la cadena, unir el eslabón precedente al nuevo, y éste al posterior. A veces, las listas escalonadas se conocen como *listas encadenadas*. Los números de LookUp ( ) —los eslabones— se denominan también *indicadores*.

Una sorprendente cualidad de las listas es su acusada serialidad; es imposible encontrar un elemento si no se parte desde el principio y se inspecciona cada elemento hasta hallar el adecuado. La lista, en este caso, se realiza mediante el uso de matrices, las cuales están proyectadas como estructuras de acceso directo, pero, de hecho, la lista las ha convertido en archivos secuenciales. En otros lenguajes, como, por ejemplo, el LISP y el PASCAL, esta característica está incorporada en el ordenador. Las listas son estructuras prácticas para el manejo de datos dinámicos (datos que cambian con regularidad), y pueden constituirse en eficaces herramientas cuando se trata con lenguajes naturales (como reconocimiento de la voz) o artificiales (compilación de programas), donde los mismos datos forman de manera natural una lista de elementos.

Presentamos "Sonido...

"Sonido y luz" es una nueva sección que le enseñará a obtener el máximo partido del sonido y de los gráficos de su ordenador personal

A medida que los ordenadores personales se han ido perfeccionando durante los últimos años, sus configuraciones se han vuelto mucho más versátiles y eficaces. Los juegos han tenido una influencia decisiva en la popularidad obtenida por cada nuevo ordenador, y se ha dedicado mucho tiempo y esfuerzos a desarrollar sofisticados gráficos en color. Aunque a primera vista no parezcan tan importantes, las características de sonido y elaboración de música han sufrido un desarrollo similar. Si se preguntara a algún escritor de programas de primera línea cuál es la importancia de las rutinas de sonido en sus programas, probablemente la situaría en un tercer lugar, tras el concepto del juego y de los gráficos. Un uso inteligente de los efectos sonoros y de la música contribuye en gran manera al entretenimiento y al interés de los juegos recreativos.

Además de las aplicaciones para juegos, cabe la posibilidad de ampliar los conocimientos musicales mediante la utilización de las características de sonido que poseen los ordenadores personales. En muchos casos se dispone de órdenes especiales para música en lenguaje BASIC, que permiten escribir programas breves para interpretar melodías bastante complejas que hasta pueden incluir acordes. Algunos ordenadores poseen también formas de cambiar la naturaleza del sonido para hacerlo más agradable al oído o convertirlo en algo parecido al de los instrumentos musicales convencionales. En todos los casos, el teclado del ordenador puede ser configurado mediante un programa adecuado, de forma que actúe como el teclado de un piano, permitiendo tocar música a "tiempo real".

Incluso en el caso de que se tengan pocos conocimientos de programación, es posible escribir programas cortos y sencillos que produzcan sonidos musicales relativamente complejos. Si se desea obtener el máximo partido de las características de sonido, muchas casas de software producen programas extensos de música que permiten escribir y tocar melodías inmediatamente. Para cualquier tipo de utilización que se haga, es conveniente que el usuario comprenda cómo el ordenador genera la producción de sonido, le da forma y la controla.

...y luz"

Alta y baja resolución

Los gráficos de los microordenadores pueden dividirse en dos categorías: de alta y baja resolución. La mejor forma de describir la diferencia entre ambas es considerar cómo se forma un carácter (una letra, número o figura).

Si se mira desde cerca un carácter normal impreso en una pantalla de televisión, se puede ver que está compuesto por un grupo de pequeños cuadros. Estos cuadros reciben el nombre de *pixels* (contracción de *picture cell*: unidad pictórica), y cada carácter o figura que aparece en la pantalla es una ordenación de aquéllos. En la mayoría de los ordenadores personales, los caracteres se forman a partir de un cuadrado de 64 pixels, agrupados en ocho filas y ocho columnas. La letra "A" puede lograrse de un modelo de pixel similar a éste:

Cada pixel iluminado en el cuadriculado puede representarse en la memoria por un "1", y cada pixel oscuro por un "0". Ocho bits crean un byte, por tanto cada fila del cuadriculado del carácter puede almacenarse en una sola localización de la memoria. Así, para obtener un carácter simple será preciso ocupar ocho localizaciones de memoria.

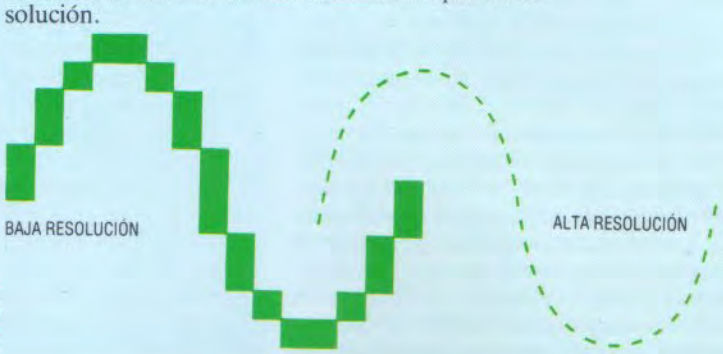
A veces, la visualización de los gráficos se construye con bloques del tamaño completo, medio o cuarto del cuadriculado de un carácter. Los gráficos proyectados empleando estos bloques sencillos y grandes se dice que son de baja resolución. En muchos ordenadores domésticos es posible diseñar gráficos compuestos por pixels simples. Éstos corresponden a visualizaciones de alta resolución. Una buena manera de demostrar la diferencia entre los dos tipos es comparar las siguientes curvas senoidales, obtenidas mediante ambos tipos de resolución.

FORMA BINARIA

0	0	0	1	1	0	0	0
0	0	1	1	1	1	0	0
0	1	1	0	0	1	1	0
0	1	1	1	1	1	1	0
0	1	1	0	0	1	1	0
0	1	1	0	0	1	1	0
0	1	1	0	0	1	1	0
0	0	0	0	0	0	0	0

MODELO DE PIXEL

Liz Dixon



Osciladores

Los osciladores son circuitos electrónicos que producen señales repetitivas. Cuando estas señales son amplificadas y enviadas a un altavoz, producen sonidos de un tono dado. El número de osciladores que posee un ordenador personal puede variar entre uno y cuatro: cuantos más osciladores haya, más notas podrán tocarse al mismo tiempo.

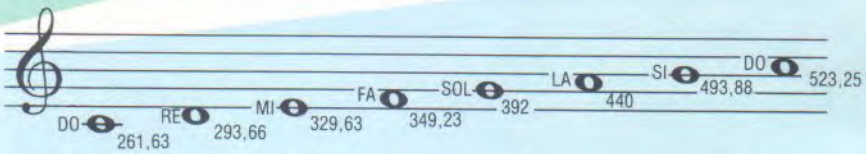
El sonido creado se define por tres características: frecuencia, envoltura (que incluye el volumen) y forma de la onda. La primera de ellas la estudiaremos en esta ocasión, y las otras dos las trataremos en el próximo capítulo.

Frecuencia

Ésta es la característica más importante que es necesario controlar, pues determina el tono del sonido. La frecuencia se define como el número de veces que se repite una señal en un segundo y su unidad de medición es el hertz (Hz, ciclos por segundo). Los sonidos que puede percibir el oído humano están comprendidos entre 20 y 20 000 Hz. Aunque no podamos oír frecuencias por debajo de los 20 Hz, éstas pueden utilizarse para modificar las características de un sonido audible. Esta técnica recibe el nombre de *modulación*, y entre los ordenadores personales, en la actualidad, sólo se puede aplicar en el Commodore 64.

Sin embargo, no es preciso profundizar más en la explicación de la frecuencia. Lo que es necesario conocer es cómo tocar las notas musicales. La facilidad con que pueda hacerse esto varía mucho de una máquina a otra. Algunas poseen órdenes en BASIC que determinan por sí mismas la frecuencia, y, por tanto, lo único que hay que especificar es el número de tono o incluso nada más que la nota musical: *la, la #, si*, etc. En otras, sin embargo, es mucho más difícil, pues sólo suministran una tabla en el manual del usuario, en la cual debe buscarse la frecuencia necesaria para cada nota y, mediante la orden POKE, introducir su valor en una posición de la memoria. La tabla proporciona los valores de conversión exactos para la escala en *do mayor*. Esta tabla también es útil si se desea programar música en código máquina, pues el lenguaje BASIC no sirve para ayudar a calcular las frecuencias.

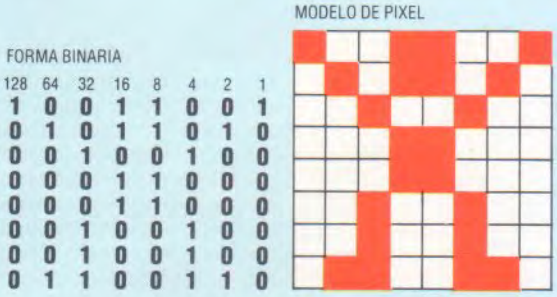
Frecuencias de las notas musicales
Se puede determinar la frecuencia de cada nota de la escala musical multiplicando la de la nota situada un semitono más abajo por 1,0594631. Esto puede parecer un poco complicado, pero si se efectúa la multiplicación 12 veces, se obtendrá una frecuencia doble de la primera. En una octava hay 12 semitonos. Por tanto, al doblar la frecuencia se logra un sonido de una octava superior. La tabla adjunta proporciona las conversiones exactas de las notas musicales (para la escala en *do mayor*) a sus frecuencias



Caracteres definidos por el usuario

Para crear en la pantalla visualizaciones atractivas y poco comunes, con frecuencia es útil tener disponibles caracteres que sean diferentes a los del conjunto alfanumérico normal. El Vic-20 y el Commodore 64 tienen un juego especial de formas gráficas que puede emplearse directamente con el teclado, pero incluso éstas no cubren todas las posibilidades. En la mayor parte de los ordenadores personales es posible crear caracteres nuevos. Esto, por lo general, se logra volviendo a definir las formas binarias de las ocho localizaciones de memoria en que está almacenado un carácter.

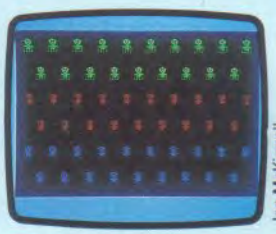
Durante el proceso, el conjunto antiguo de formas binarias a menudo se pierde, y el "definido por el usuario" adquiere algunas de las propiedades del sustituido en la memoria. Así el nuevo puede usarse con la orden PRINT, simplemente apretando la tecla del carácter que ha sido reemplazado. A continuación se muestra un ejemplo, junto con sus códigos binarios asociados:



La facilidad con que se pueden establecer varía mucho según el tipo de ordenador que se utilice. Por ejemplo, en el Sinclair Spectrum, con la orden USR, lo único que se necesita es introducir las formas binarias apropiadas; sin embargo, en el Commodore 64, el usuario, en primer lugar, tiene que desplazar todo el conjunto de caracteres desde ROM a RAM antes de (mediante la orden POKE) introducir en la memoria los ocho equivalentes decimales de las formas binarias que crean la figura. No obstante, es posible adquirir, a través de distribuidores independientes, cierto número de programas de diseño de caracteres que pueden facilitarle de manera notable la labor al usuario del ordenador Commodore 64.

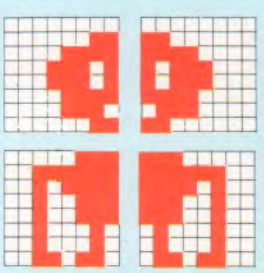
Para crear figuras más grandes, es posible agrupar dos o más caracteres. La figura del "extraterrestre" que podemos ver en la esquina inferior derecha está formada con cuatro caracteres definidos por el usuario. El programa en funcionamiento en el Commodore 64 imprime el grupo en la pantalla en tres colores diferentes. Los caracteres se crearon mediante una rutina corta para desplazar el conjunto normal de ROM a RAM y sustituir las formas gráficas ▞ , ▐ , ▟ , y ` ' leyendo números decimales de sentencias DATA y utilizando órdenes POKE para colocarlos en las posiciones apropiadas. Este tema lo trataremos con mayor profundidad en próximos capítulos.

Incluso cuando se dispone de "sprites" (véase p. 152), frecuentemente existe un límite en el número que se puede visualizar en la pantalla en cada momento. Por eso los gráficos definidos por el usuario cobran toda su importancia cuando es necesario representar muchas figuras similares a un mismo tiempo.



Ian McKinnell

Extraterrestres
Estas raras criaturas se crearon a partir de cuatro caracteres, cada uno de ellos definido por el programador. Este sistema puede usarse en máquinas que no poseen "sprites"



Pronósticos más precisos

El uso de ordenadores de alta velocidad para procesar imágenes de satélites y analizar las estructuras de datos ha permitido que las predicciones meteorológicas sean mucho más precisas

Los resultados de muchas de las tareas de proceso de datos más complejas están presentes en nuestra vida cotidiana, con frecuencia sin que seamos conscientes de ello. Una de las aplicaciones más avanzadas del ordenador, que requiere una capacidad de procesamiento de datos mayor que casi cualquier otra, nos da información diaria sobre el estado del tiempo y su predicción. Dada la complejidad del pronóstico del tiempo, quizá parezca sorprendente que los meteorólogos acierten con tanta frecuencia. La ayuda de los ordenadores representa para ellos una inmensa ventaja en el manejo de tan vasta gama de posibilidades.

Los factores climatológicos que afectan al tiempo en las regiones occidentales europeas son muy complejos. En primer lugar, están condicionados por la proximidad al polo Norte y al océano Atlántico. Al estar situadas al este del Atlántico, las regiones occidentales europeas son más proclives a los fenómenos climatológicos creados en dicho océano, debido al efecto Coriolis, fenómeno que se debe al giro de la Tierra de oeste a este. Esto se comprenderá mejor si recordamos que en el ecuador un objeto situado en la superficie de la Tierra viaja a más de 1 600 km/h; y este potente movimiento de rotación, combinado con las corrientes de aire normales del polo al ecuador, crea los vientos predominantes del oeste en el hemisferio Norte. Es este constante ataque de aire húmedo —que aumenta o disminuye según las variaciones locales de temperatura— el que determina las condiciones climatológicas en el oeste de Europa.

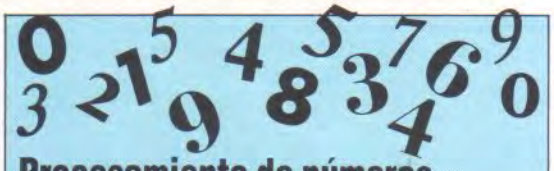
Para sus predicciones, los servicios de meteorología cuentan con estaciones de recogida de datos situadas en lugares estratégicos del Atlántico —bar-

Fotografías desde el espacio
El satélite meteorológico Meteosat 2, lanzado en junio de 1981, está situado en una órbita geoestacionaria (es decir, no se mueve con relación a la Tierra) a unos 35 880 km sobre el ecuador, en el meridiano cero. Reúne información de un gran número de estaciones terrestres

cos meteorológicos, boyas, globos y aviones de reconocimiento—, que suministran datos sobre las condiciones más inmediatas. Y con esta información se predice qué sucederá cuando los fenómenos climatológicos originados en el océano lleguen a tierra firme, de acuerdo con el comportamiento de fenómenos anteriores.

Antes de marzo de 1979, cuando el satélite meteorológico Meteosat I fue puesto en órbita, el único método de predicción disponible por los hombres del tiempo era trasladar los informes de las estaciones climatológicas a un mapa para formar una gráfica isobárica. Las isobaras son líneas imaginarias que unen puntos con igual presión barométrica, de la misma forma que las líneas de nivel de un mapa unen puntos de la misma altura. Basándose en las isobaras, es posible determinar la velocidad y dirección de frentes fríos o cálidos —y, en asociación con ellos, de borrascas y anticiclones— y de esta forma predecir el estado del tiempo.

Si bien los mapas isobáricos son sin duda los más utilizados, no son, sin embargo, los únicos que se trazan en los diversos institutos meteorológicos. A partir de la base de datos climáticos que poseen sus sistemas de ordenadores, éstos levantan mapas que muestran temperaturas medias, precipitaciones, horas de sol por día, etc.



Procesamiento de números

Uno de los principales usos de los grandes ordenadores en la investigación científica consiste en el procesamiento de información exclusivamente numérica en forma de ecuaciones muy extensas y complejas. Las aplicaciones puramente científicas, como por ejemplo de física nuclear, o de ciencia aplicada, como las de meteorología, exigen requisitos similares. Si bien existe la posibilidad de llevar a cabo cálculos de este grado de complejidad con un microordenador, el tiempo necesario para ello sería excesivo, como consecuencia no sólo del número de términos de la ecuación, sino por la desmesurada magnitud de los números a tratar. Para poder realizar esta operación en un tiempo razonable, es necesario disponer de ordenadores muy rápidos y con gran capacidad de memoria

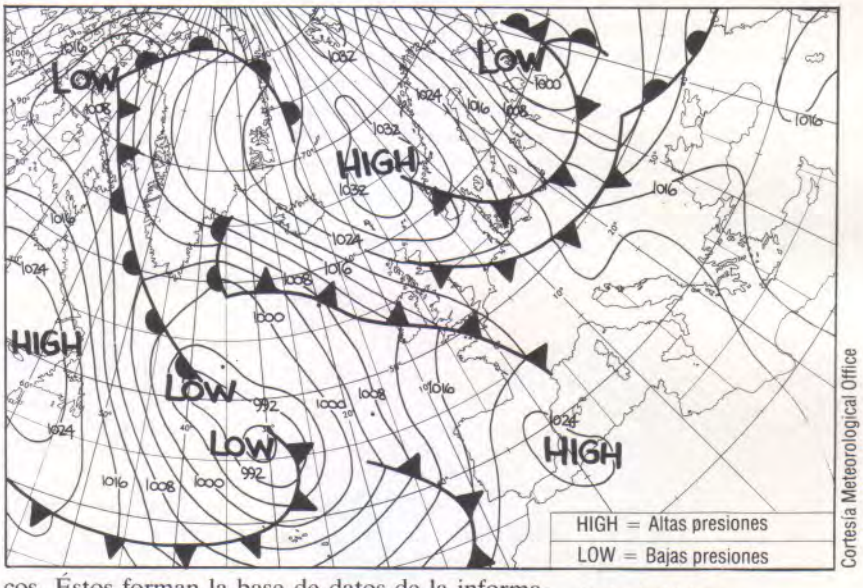
Los servicios meteorológicos siguen aún este procedimiento para trazar sus precisos mapas del estado del tiempo, pero hoy en día emplean también las imágenes recibidas desde el Meteosat. Éstas son señales analógicas que son digitalizadas por el ordenador para su procesamiento y visualización en forma de mapas coloreados artificialmente. Las imágenes crean un vívido panorama de la composición del tiempo en ese momento. Estos mapas se rehacen cada cuatro minutos aproximadamente, y, por lo tanto, el meteorólogo puede observar las variaciones climáticas en tiempo real.

El Meteosat 2, que reemplazó al satélite anterior en junio de 1981, está situado en una órbita geoestacionaria a unos 35 880 km sobre el ecuador, en el meridiano cero. Reúne información que le proporcionan un gran número de estaciones terrestres esparcidas por toda la superficie del globo, que es transmitida a quienquiera que desee suscribirse al sistema.

Teóricamente sería posible analizar e interpretar esta información (aunque no en tiempo real) mediante un ordenador personal, escribiendo en un disco los datos que se reciben del satélite. Sin embargo, la señal es de tipo analógico, y, por consiguiente, la conversión podría ser difícil. También sería necesario instalar una antena parabólica alineada exactamente con el satélite. El procesamiento de estas imágenes transmitidas por el satélite es sólo una función muy pequeña del sistema de ordenadores del Servicio Meteorológico Nacional. Junto con otras organizaciones de este mismo carácter localizadas en otras partes del mundo, mantiene un modelo del sistema del tiempo global y extrae de éste una gran cantidad de datos estadísticos. Éstos forman la base de datos de la información histórica a partir de la cual se pueden pronosticar tendencias meteorológicas y establecer las características del clima local y global, que incluyen no sólo datos barométricos, sino también detalles sobre la velocidad y dirección del viento, precipitaciones y temperaturas al nivel del suelo y del mar.

El conjunto de estos datos tiene un valor determinante para el análisis histórico. De la misma manera, esta información es de importancia vital para muchas industrias, como asimismo para la economía y ecología de continentes enteros, pues únicamente por este medio pueden determinarse los cambios en el clima.

Estaciones terrestres
Las antenas parabólicas de recepción de señales emitidas desde satélites pueden ser muy diferentes en complejidad y tamaño. La que aparece aquí es capaz de recibir y transmitir señales, y no se limita a captar información desde los satélites geoestacionarios. Posee un sofisticado control por ordenador que le permite seguir la trayectoria de un satélite



Cortesía Meteorological Office

Cartas isobáricas
Los "mapas del tiempo" que vemos en televisión son en realidad cartas de la presión barométrica. Las líneas concéntricas unen puntos de igual presión atmosférica. Los vientos giran en dirección contraria a las agujas del reloj alrededor de un centro de "bajas" presiones, y en sentido inverso alrededor de un punto de "altas" presiones (al contrario en el hemisferio Sur), y la velocidad del viento es directamente proporcional a la distancia entre las isobaras

Sord M5

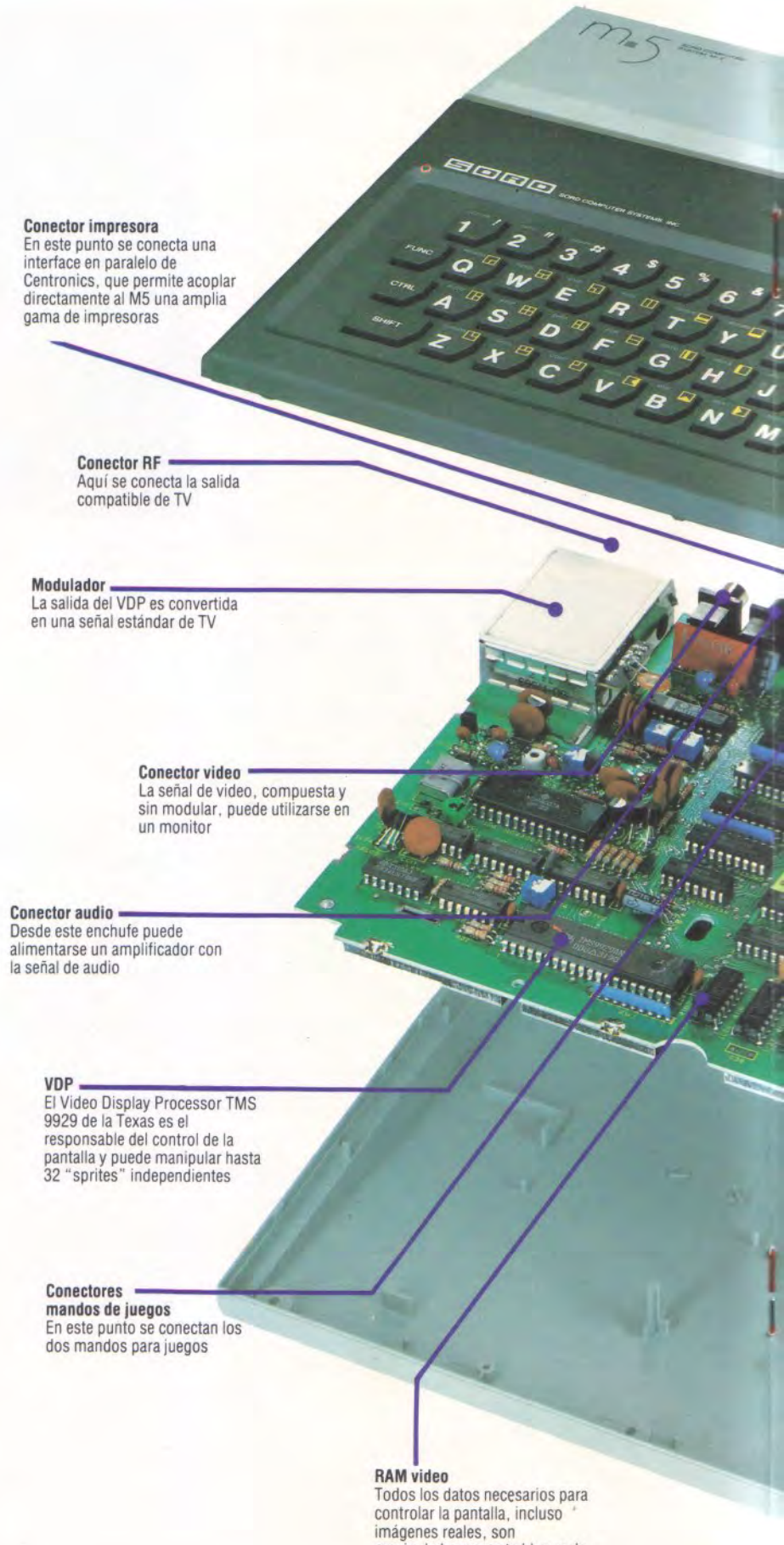
A pesar de que este ordenador, en su versión estándar, posee sólo cuatro Kbytes de memoria, sus magníficas características gráficas permiten escribir valiosos programas

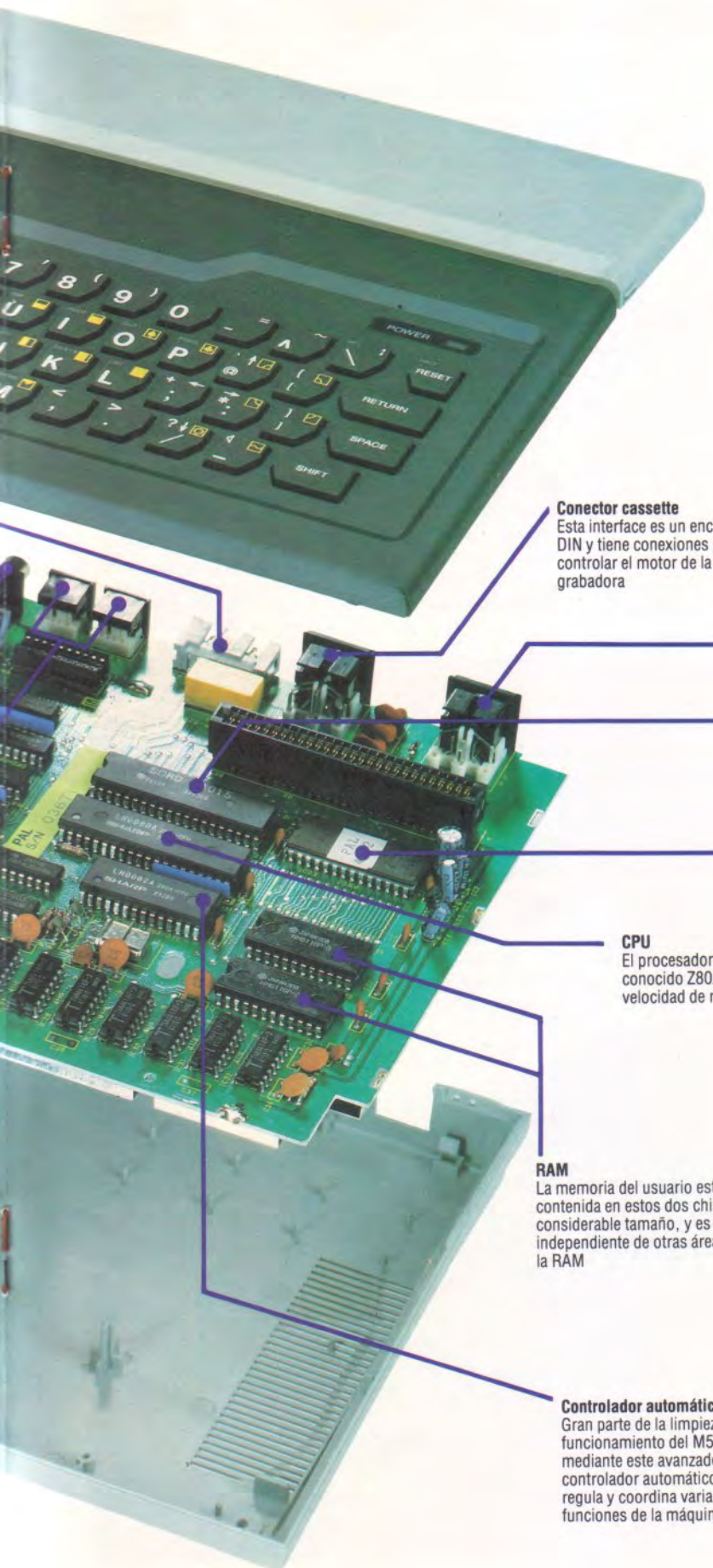
La mayor parte de los ordenadores personales de la primera época fueron diseñados en California (Estados Unidos). Recientemente, también las máquinas proyectadas en Gran Bretaña han empezado a conquistar buena parte del mercado mundial. Sin embargo, es previsible que dentro de poco tiempo los japoneses dominen la escena, tal como ha sucedido en otros sectores del mercado de la electrónica. Ciertamente, el Sord M5 no es el primer microordenador japonés, pero sí ha sido el primero en tener un éxito importante entre los ordenadores personales, a diferencia de lo sucedido en el mercado de los comerciales.

Es una máquina sólida y compacta, de tamaño similar al del Sinclair Spectrum, pero considerablemente más pesada y da la sensación de ser mucho más poderosa. En otros muchos aspectos tiene una capacidad similar, con una CPU Z80A, una sola tecla para entrada de BASIC, y almacenamiento de programas e información en cassette. No obstante, en su interior es mucho más sofisticado, tal como se pone de manifiesto por la conexión de la impresora Centronics que lleva incorporada la máquina. Pero las dos diferencias más importantes son el tamaño de la memoria RAM —que con sus 4 Kbytes (ampliables a 36) es mucho más pequeña— y la inclusión de chips de sonido y gráficos.

Los gráficos son elaborados por un TI 9918, 9928

Cartucho de ROM
Una de las características más sobresalientes del M5 es la posibilidad de cambiar de lenguaje, debido a que éste se encuentra almacenado en un cartucho de ROM. El M5 dispone de tres versiones de BASIC: BASIC-I (simple, para principiantes); BASIC-G (muy eficaz en gráficos), y BASIC-F (científico y matemático). Existe también un programa especial de uso general, llamado FALC, que consta de una combinación de hojas electrónicas, archivo y funciones gráficas, para desarrollar aplicaciones complejas para uso personal o de gestión





Mandos de juegos
Son los equivalentes de las palancas de mando. Funcionan enviando una señal a cada uno de los cuatro puntos en diagonal. Estas señales interrumpen a la CPU, por lo que el tiempo de respuesta es muy breve

Conector cassette
Esta interface es un enchufe tipo DIN y tiene conexiones para controlar el motor de la grabadora

Conector potencia
Desde aquí se obtiene la fuente de alimentación, a través de un pequeño transformador

Chip de diseño propio
El M5 utiliza una pieza de sofisticado y exclusivo diseño para realizar sus funciones superiores a un precio razonable

ROM
Los únicos programas incorporados en la máquina son un conjunto de programas de control de bajo nivel, que son utilizados por el programa del usuario. Cuidan de los detalles del manejo de la pantalla, teclado y cassette

CPU
El procesador del Sord M5 es el conocido Z80A. Éste tiene una velocidad de reloj de 3,58 MHz

RAM
La memoria del usuario está contenida en estos dos chips de considerable tamaño, y es independiente de otras áreas de la RAM

Controlador automático
Gran parte de la limpieza de funcionamiento del M5 se logra mediante este avanzado controlador automático, que regula y coordina varias funciones de la máquina

o 9929 (según sea el país en que se vende el ordenador), que da una resolución de 192 × 256 puntos de hasta 16 colores diferentes. Hay cuatro modelos gráficos principales, tres de los cuales permiten tener hasta 32 "sprites" capaces de moverse independientemente, pudiendo ser de tamaño normal o ampliado. La máquina puede visualizar letras mayúsculas y minúsculas, signos de puntuación y caracteres numéricos. Posee símbolos de dibujo de líneas y bloques, así como una amplia gama de letras minúsculas acentuadas, y, como cada carácter puede redefinirse, las posibilidades que ofrece son muy amplias.

Otros ordenadores emplean los mismos chips de gráficos —en particular el TI99/4A (véase p. 189)— y es el uso de éstos lo que hace que el Sord M5 sea tan efectivo a pesar de su falta de memoria RAM. Puesto que la memoria de pantalla es totalmente independiente de la del programa, el único contenido de la RAM principal será el programa real, además, por supuesto, de los datos necesarios para las variables.

SORD M5

DIMENSIONES
185 x 70 x 55 mm

PESO
1 kg

CPU
Z80A

VELOCIDAD DEL RELOJ
3,58 MHz

MEMORIA
8 Kbytes de ROM
20 Kbytes de RAM, de los cuales 16 se utilizan para visualizaciones gráficas.
Con la adición de cartuchos, la ROM puede ampliarse a 16 Kbytes, y la RAM a 32

VISUALIZACION EN VIDEO
Hasta 16 colores, que pueden emplearse en diferentes "planos". Consta de gráficos "sprite" y cuatro tipos de pantalla diferentes: dos para gráficos, uno para texto y otro "multicolor"

INTERFACES
Cassette, impresora (Centronics), mandos de juegos, cartucho de ROM, audio

LENGUAJE SUMINISTRADO
El cartucho de lenguaje es el BASIC completo: BASIC-I

OTROS LENGUAJES DISPONIBLES
BASIC-G (gráficos), BASIC-F (BASIC de coma flotante), FALC (lenguaje para hojas electrónicas y base de datos)

VIENE CON
Adaptador de fuente de energía, cable cassette, cable televisión, dos palancas de mando con cables, un cartucho de lenguaje BASIC-I y una cassette con dos juegos

TECLADO
55 teclas: 8 teclas de cambio proporcionan todos los caracteres alfanuméricos, 28 funciones BASIC y 64 formas gráficas

DOCUMENTACION
Una guía del usuario en la que se describe la forma de conectar el ordenador, y cómo cargar y hacer funcionar los dos juegos; una página está dedicada a detectar errores simples. No existe descripción alguna sobre el lenguaje BASIC o del uso del cassette o de otras interfaces no relacionadas con los juegos suministrados

Un tema arduamente discutido en la actualidad en la industria de los ordenadores personales es el que se refiere a la propuesta de implantación de unas "normas MSX", desarrolladas por un grupo integrado por los fabricantes más impoortantes del Japón, incluida la Sord. La idea es que si los fabricantes se atienen a estas normas de estandarización en el diseño de los ordenadores personales (que afectan tanto al hardware como a la versión de BASIC a emplear), sería posible escribir software que funcionara en todas las máquinas, sin modificación alguna. Desde el punto de vista de los chips para gráficos, el Sord cumple estas normas.

Sin embargo, las MSX también especifican que el chip de sonido debe ser el AY-3-8910 de la General Instruments. Para producir los sonidos, el Sord M5 (al igual que el BBC Micro) utiliza un chip TI 76489, el cual tiene un mejor control sobre la gama de sonidos producidos que el chip GI, aunque se asemeja a éste al tener tres canales para tonos y uno para la creación de efectos especiales. Esto significa que el M5 no cumple todas las especificaciones que debería reunir una máquina MSX.

Con el cartucho de ROM pueden suministrarse tres versiones diferentes de BASIC, diversos complementos, juegos y varias aplicaciones, y como puede ampliarse hasta 16 Kbytes, es probable que aparezcan en el mercado programas útiles para esta máquina.

Chris Stevens

El teclado del Sord M5
El teclado de caucho es ligeramente mayor que el del Sinclair Spectrum, y su funcionamiento más suave le proporciona una mayor comodidad al usuario. Se pueden emplear un total de 55 teclas en varias formas, para obtener caracteres alfanuméricos, símbolos gráficos, o funciones en BASIC, mediante la tecla FUNC. Todas las teclas, mientras se mantengan apretadas, continuarán funcionando automáticamente, lo cual es muy práctico para procesar textos en pantalla

El control de paridad

La "paridad par" asegura que el número de bits 1 de un byte sea siempre par, lo que hace más fácil detectar errores de transmisión

Una de las principales ventajas de los ordenadores digitales, en comparación con los dispositivos analógicos, es que los errores e inexactitudes que tienen lugar en todos los circuitos eléctricos no se acumulan al pasar una señal a través de numerosos circuitos (véase p. 239). Sin embargo, si se transmiten datos a cierta distancia —ya sea mediante una interface serial y un par de cables, o a través de una línea telefónica—, el "ruido" eléctrico de fondo que exista en la línea puede ser suficiente para cambiar un bit sencillo de 0 a 1, o viceversa. Por lo general, el ordenador receptor no estaría en condiciones de detectar esta alteración y, en consecuencia, aceptaría el dato erróneo como correcto.

Veamos lo que sucede si un bit del código ASCII para la letra Q resulta alterado en la transmisión:

[ ] 1010001 (código ASCII transmitido para Q)
[ ] 1000001 (código ASCII recibido para A)

Un error como este en la transmisión de datos sería, como mínimo, un grave contratiempo y podría ser potencialmente catastrófico. Sin embargo, se recordará que los códigos ASCII se asignan únicamente a valores hasta 127, que requieren sólo siete bits (numerados de 0 a 6). El MSB (*Most significant Bit:* bit más significativo) —el bit siete— se emplea, en consecuencia, a menudo como un bit "de paridad", con el fin de detectar si ha ocurrido un error.

Existen dos acuerdos para la utilización de bits de paridad: "paridad par" y "paridad impar". Nosotros consideraremos la primera de ellas.
"Paridad par" significa que el bit de paridad (bit 7 en un código ASCII) está regulado de forma que el número total de bits 1 en el byte sea siempre un número par. Con la paridad par, las letras A y Q tendrán la siguiente composición:

[0] 1000001
(código ASCII para A con paridad par)
[1] 1010001
(código ASCII para Q con paridad par)

En el código ASCII para A hay dos bits 1; por lo tanto, el bit de paridad será 0 con el fin de que el total de los ocho bits sea par. En el código ASCII para Q existen tres bits 1; por consiguiente, el bit de paridad será 1. Esto hace que el número total de bits 1 sea cuatro, que es un número par.

Ahora veamos qué sucedería si el bit 4 de nuestro código ASCII para la letra Q fuera alterado como en el ejemplo anterior.

[1] 1000001 (código ASCII para Q alterado)

Al comprobar la paridad del byte (ya sea mediante software o por hardware especial), se ve que la Q correcta tiene un número par de "unos" (incluyendo el bit de paridad). La Q alterada, por el contrario, tendría el bit 4 accidentalmente cambiado de 1 a 0, pero el bit de paridad original —bit 7— continúa siendo 1. Al comprobar la paridad de este byte alterado, se encontraría un número impar de bits 1, y, por lo tanto, resultaría evidente que el byte estaba alterado y sería rechazado. Si se piensa sobre esto, se verá que incluso si el propio bit de paridad fuera el que se alterara en la transmisión, el hecho de que había ocurrido un error sería también detectado en el proceso de control de paridad, y, en consecuencia, el byte sería rechazado.

Si observa los códigos ASCII de cualquier ordenador, probablemente comprobará que el bit 7 (el MSB: el bit más significativo) se utiliza en realidad, pero no como bit de paridad. Se ha dispuesto así a fin de permitir que el ordenador tenga un juego de caracteres adicionales (en general, un juego de caracteres gráficos). Además, hay que considerar que los errores de transmisión de datos *en el interior* de un ordenador son muy poco frecuentes. Normalmente la paridad se emplea sólo cuando se transmiten datos a larga distancia, o cuando se graban datos en una superficie magnética (por ejemplo una cassette o un disco), que también es susceptible de incurrir en "errores de bits".

El control de paridad es adecuado para indicar que un byte dado ha sido transmitido incorrectamente, pero no aporta información sobre cuál de los bits de un byte ha sido transmitido erróneamente, y, por consiguiente, el error no puede ser corregido por el ordenador que lo recibe. Incluso puede presentarse una circunstancia peor: si en un byte hay dos bits alterados, un byte incorrectamente transmitido puede ser tomado como correcto.

Sin embargo, en los casos en que el dispositivo receptor detecta un error, puede enviar un mensaje de error y el software, por su parte, puede dar las órdenes adecuadas para que el byte incorrecto se emita otra vez. Finalmente, debemos agregar que se han desarrollado sistemas muy sofisticados para la detección y corrección de errores; son capaces de indicar qué bit o bits han sido alterados, permitiendo, de esta manera, que sean corregidos automáticamente. El tema de los códigos de corrección de errores lo trataremos con detenimiento en una próxima oportunidad.

Sólo para controlar
El último dígito del International Standard Book Number (ISBN) tiene sólo funciones de control, equivalentes a las que desempeña el par de paridad en un ordenador. Si se multiplica el primer dígito (aquí 0) por 10, el segundo (5) por 9, y así sucesivamente, y se suma el resultado, se comprobará que el dígito de control se ha elegido de forma tal que el resultado sea divisible por 11

Clasificar y archivar

Como continuación de nuestro proyecto de programa para desarrollar una agenda computerizada, veremos cómo el archivo de datos deberá ser dividido en registros y campos

El capítulo anterior de nuestro curso de programación acababa tratando el tema de refinar los elementos de un ejercicio de programación mediante uno o más niveles de "seudolenguaje", hasta alcanzar un punto en donde los ejemplos pudieran ser codificados en BASIC. Empezaremos por revisar este ejercicio y dar algunas soluciones. La primera "enunciación de objetivos" para el ejercicio era:

INPUT
Un nombre (en cualquier formato)
OUTPUT
1. El nombre de pila
2. El apellido

En el primer nivel de refinamiento encontramos que podía ser dividido en seis etapas (más tarde vimos que la última podía omitirse). Éstas eran:

1. Leer el nombre completo (★LEER★)
2. Convertir en mayúsculas todas las letras (★CONVERT★)
3. Hallar el último espacio (★ESPACIO★)
4. Leer el apellido (★LEER APELL$★)
5. Leer el nombre de pila (★LEER NOMB$★)
6. Eliminar los caracteres no alfabéticos del nombre de pila

Todos estos temas se están tratando como subrutinas y el nombre que se ha asignado a cada una de ellas está descrito entre paréntesis. Desafortunadamente, la mayor parte de versiones de BASIC no pueden hacer referencia a las subrutinas por los nombres y será necesario, al escribir el programa final, insertar números de líneas después de los respectivos GOSUB. Durante la fase de desarrollo, sin embargo, resulta mucho más fácil referirse a ellas por el nombre. Posteriormente, estas denominaciones pueden incorporarse en sentencias REM. Este uso de subrutinas con nombres se indica colocando éstos entre asteriscos. En los lenguajes que pueden referirse a las subrutinas por el nombre (por ejemplo, el PASCAL), éstas son designadas, por lo general, como "procedimientos".

Incluso en el caso de que una versión de BASIC determinada no pudiera hacer uso de procedimientos, es recomendable que, mientras se está programando al nivel de seudolenguaje, se proceda como si pudiera emplearlos. Del mismo modo, alguna versión de BASIC no podrá tratar nombres largos de variables, tales como PROVINCIA o NOMBCALLE$, pero al nivel de seudolenguaje resultará mucho más fácil y claro darlo por supuesto. Trate de que los nombres sean descriptivos. Es mucho más claro denominar a una variable transitoria de una serie (string) VARTRAN$ que llamarle XV$. Afortunadamente, numerosas versiones de BASIC permiten usar nombres de variables más largos.

Ya hemos desarrollado la segunda de las etapas (convertir en mayúsculas todas las letras) mediante un segundo y tercer nivel de refinamiento, y se ha creado un programa corto en BASIC para realizar este cometido. Ahora efectuaremos lo mismo para las otras etapas:

2.º REFINAMIENTO
3. (Hallar último espacio)
BEGIN
LOOP mientras los caracteres no explorados permanecen en NOMBRE$
  IF carácter = " "
    THEN anotar posición en una variable
    ELSE no hacer nada
  ENDIF
ENDLOOP
END

3.er REFINAMIENTO
3. (Hallar último espacio)
BEGIN
READ NOMBRE$
LOOP (mientras los caracteres no explorados permanecen)
  FOR L = 1 TO longitud de NOMBRE$
  READ carácter de entre NOMBRE$
  IF carácter = " "
    THEN LET CONT = posición de carácter
    ELSE no hacer nada
  ENDIF
ENDLOOP
END

Ahora estamos en condiciones de codificar en lenguaje de programación a partir del seudolenguaje:

```
10 INPUT "INTRODUCIR NOMBRE COMPLETO";
   NOMBRE$
20 FOR L = 1 TO LEN (NOMBRE$)
30 LET CAR$ = MID$ (NOMBRE$,L,1)
40 IF CAR$ = " " THEN LET CONT = L
50 NEXT L
60 PRINT "ULTIMO ESPACIO ESTA EN POSICION";
   CONT
70 END
```

Nótese que la línea 10 es una entrada falsa, para verificar la rutina; la línea 60 es una salida falsa, también para verificación; y la línea 70 deberá ser cambiada por RETURN cuando la rutina se use como subrutina.

Y ahora el mismo proceso para la etapa cuarta:

2.º REFINAMIENTO
4. (Leer apellido)
BEGIN
Asignar a APELL$ los caracteres a la derecha del último espacio de NOMBRE$
END

3.er REFINAMIENTO

```
4. (Leer apellido)
BEGIN
   READ NOMBRE$
   Localizar último espacio (llamar la subrutina
   ★ESPACIO★)
   LOOP mientras permanezcan caracteres en serie
   tras ESPACIO
      READ caracteres y asignar a APELL$
   ENDLOOP
END
```

Antes de continuar codificando en BASIC, se deben tener en cuenta algunos peligros potenciales. Al localizar el último espacio en el refinamiento anterior, el seudolenguaje exige el uso de la subrutina ★ESPACIO★, pero no sería posible escribirla en BASIC y comprobarla si ésta aún no se había escrito. Por regla general, no es práctico codificar cada módulo en BASIC (o cualquier otro lenguaje de alto nivel) hasta que no se haya desarrollado todo el programa en seudolenguaje. Sin embargo, si se desea comprobar un módulo, puede que resulte necesario escribir algunos valores de variables, entradas y salidas falsas. En el ejemplo anterior, CONT es la variable que tiene el valor de la posición del último espacio en NOMBRE$. Al comprobar, podemos hacer una pequeña trampa suponiendo que la rutina para realizar esto funciona correctamente:

```
10 LET NOMBRE$ = "PEDRO GONZALEZ"
20 LET CONT = 6
30 FOR L = CONT + 1 TO LEN (NOMBRE$)
40 LET APELL$ = APELL$ + MID$
   (NOMBRE$, L, 1)
50 NEXT L
60 PRINT "APELLIDO ES "; APELL$
70 END
```

A continuación se indica el proceso para encontrar el nombre de pila (etapa cinco). Recuérdese que se estableció que un nombre de pila es una concatenación de todos los caracteres alfabéticos hasta el último espacio del nombre completo. Los puntos, apóstrofos, espacios, etc., debían descartarse.

2.º REFINAMIENTO

```
5. (Leer nombre de pila)
BEGIN
LOOP hasta el último espacio mientras permanezcan
   caracteres en NOMBRE$
   Examinar caracteres
   IF carácter no es una letra
      THEN no hacer nada
      ELSE añadir carácter a NOMB$
   ENDIF
ENDLOOP
END
```

3.er REFINAMIENTO

```
5. (Leer nombre de pila)
BEGIN
LOOP hasta CONT mientras permanezcan caracteres
   LET CARTRAN$ = L° carácter en la variable
   IF CARTRAN$ no es una letra
      THEN no hacer nada
      ELSE LET NOMB$ = NOMB$ + CARTRAN$
   ENDIF
ENDLOOP
```

Ahora ya se puede codificar en BASIC, pero como se está en una etapa intermedia, vamos a utilizar sentencias en este lenguaje sin numerar, en un formato estructurado, de modo que se pueda comparar la estructura con la etapa anterior:

CODIFICACION

```
5. (Leer nombre de pila)
REM BEGIN
REM LOOP
   FOR L = 1 TO CONT −1
      LET CARTRAN$ = MID$ (NOMBRE$,L,1)
      LET CAR = ASC(CARTRAN$)
      IF CAR > 64 THEN NOMB$ =
         = NOMB$ + CHR$ (CAR)
      REM ENDIF
   NEXT L: REM ENDLOOP
REM END
```

En BASIC corriente esto tendría la forma siguiente:

```
10 FOR L = 1 TO CONT −1
20 LET CARTRAN$ = MID$ (NOMBRE$,L,1)
30 LET CAR = ASC(CARTRAN$)
40 IF CAR > 64 THEN NOMB$ = NOMB$ +
   + CHR$ (CAR)
50 NEXT L
60 END
```

Tal como está, sin embargo, este programa no funcionaría. Existen tres problemas: CONT requiere que se le asigne un valor; no está previsto introducir un nombre (asignando una variable a NOMBRE$); y no existe "salida" en forma de una frase impresa para que se pueda verificar si ha funcionado de forma correcta.

Si esta rutina formara parte de una subrutina, los parámetros pasados a ella (input) y los pasados desde ella (output) tendrían que ser tratados en otra parte del programa. Ésta es una consideración muy importante: el flujo de información en el interior de un programa debe ser siempre pensado con sumo cuidado antes de empezar a codificar en BASIC. Esto es particularmente importante cuando se emplean variables (CONT, por ejemplo) y se utiliza el mismo nombre de variable en diferentes partes del programa. De nada sirve llamar a una subrutina que emplea una variable como CONT si aquélla no tiene forma de saber cuál es el valor que se le atribuye. Si una subrutina indica el valor de CONT, éste seguirá siendo el mismo a menos que posteriormente se le asigne un nuevo valor, tal vez en otra subrutina. Ésta es una razón del porqué no es buena práctica en la programación dejar a medias un bucle, puesto que el valor de la variable del bucle será desconocido. Veamos a continuación las consecuencias de tener estos dos fragmentos del programa como partes de subrutinas diferentes en un programa:

Parte de subrutina X

```
FOR L = 1 TO LEN (PALABRA$)
LET CAR$ = MID$ (PALABRA$,L,1)
IF CAR$ = " . " THEN GOTO 1550
NEXT L
```

Parte de subrutina Y

```
FOR Q = 1 TO LIMIT
LET A(L) = P(Q)
NEXT Q
```

Esta parte de la subrutina Y está llevando valores a una variable que posee un subíndice determinado, el cual corresponde a la variable L. Si la subrutina Y

se utiliza después de la X, y la condición de la prueba en esta última ha sido encontrada (que uno de los caracteres sea un "."), el valor de L sería completamente impredecible y, por lo tanto, no se podría saber qué elemento de entre los valores de la variable le sería asignado en la subrutina Y. Aparte del error de ramificar un bucle, esta subrutina emplea también un GOTO, y esta práctica debiera evitarse. Los GOTO producen confusión y siempre que sea posible debería eludirse su utilización.

Para evitar confusión en el uso de variables, una buena norma es confeccionar una lista de ellas cuando se está en las etapas de seudolenguaje en el desarrollo del programa, junto con notas que indiquen para qué se están utilizando. Algunos lenguajes (pero no el BASIC) permiten establecer las variables como "locales" o "globales": esto es, tienen valores que se aplican a una parte del programa (locales) o a todo él (globales). Muchas variables, como, por ejemplo, las empleadas en bucles (la L en LET L = 1 TO 10), son casi siempre locales, por eso es muy conveniente dar el valor inicial de la variable antes de utilizarla (por ejemplo LET L = 0). Algunos lenguajes, como el PASCAL, hacen hincapié en esto; y aunque el BASIC siempre presupone que el valor inicial de una variable es 0 (mientras no se indique lo contrario), es recomendable efectuar lo anteriormente indicado.

Hasta ahora hemos formulado una definición razonable de un nombre, adecuada para la agenda computerizada que se intenta realizar, y se han desarrollado algunas rutinas que pueden manejar nombres en varias formas, que utilizaremos en nuestro programa completo. Ahora vamos a prescindir de los detalles de codificación del programa y entraremos a considerar la estructura de los "registros" en nuestro "archivo" de direcciones.

Los términos "registro" (*record*), "archivo" (*file*) y "campo" (*field*) tienen significados muy específicos y precisos en el mundo de la informática. Un *archivo* es un conjunto completo de información relacionada entre sí. En un sistema de ordenadores, sería un tema identificable, almacenado en un disco flexible o en una cinta de cassette y tendría su denominación propia, normalmente haciendo referencia al tema que trata. Podemos considerar la agenda completa como un archivo, y lo llamaremos AGENDA.

En un archivo tenemos *registros*, que son también conjuntos de información relacionada entre sí. Si imaginamos que el conjunto de direcciones constituye un fichero, el archivo sería la caja completa, llena de tarjetas, y cada ficha constituiría un registro: cada una de ellas con su nombre, dirección y número de teléfono.

En cada registro tenemos *campos*. Éstos pueden ser considerados como una o más filas de información afín contenida en un registro. Cada uno de los registros de nuestro archivo AGENDA constará de los siguientes campos: NOMBRE, DIRECCION y NUMERO TELEFONO. Un registro normal sería de la siguiente manera:

```
José Luis Padilla
General Dávila, 8
Madrid-3
Madrid
(91) 234 72 93
```

En este registro existen tres campos: el del nombre, que comprende letras alfabéticas (y probablemente el apóstrofo en ciertos nombres como Leopoldo O'Donnell, por ejemplo); el campo de la dirección, que consta de algunos números y bastantes letras; y el del número telefónico, que incluye únicamente números, sin tener en cuenta la eventualidad de que haya algún paréntesis en números como (91) 234 56 78. Antes de empezar a escribir un programa que maneje con flexibilidad información compleja como ésta, tenemos que decidir de qué manera representar los datos en el ordenador. Una forma podría ser considerar toda la información contenida en un registro como si fuera sólo una larga serie de caracteres. El problema que se presentaría con este tratamiento es que la extracción de una información específica sería extremadamente laboriosa. Supongamos por un momento que la siguiente entrada no es más que una larga serie de caracteres:

```
PERCIVAL R. BURTON
1056 AVENUE OF THE AMERICAS
RIO DEL MONTENEGRO
CALIFORNIA
U.S.A.
(1213) 884 5100
```

Si estuviéramos buscando los registros para hallar el número de teléfono de PERCIVAL R. BURTON, ¿sería correcto considerar que los últimos 15 caracteres del registro representan el número? ¿Qué sucedería si se hubiera incluido el prefijo de llamadas internacionales, de esta manera: 07 (1213) 884 5100? Entonces el número hubiera tenido un total de 18 caracteres. Para superar esta dificultad, se le asigna un campo separado, y el programa nos dará todos los caracteres (o números) de este campo cuando se le pida.

La dificultad de este planteamiento reside en que tiene que haber alguna forma de relacionar los diversos campos independientes, de modo que al referirnos a un campo (al del nombre, por ejemplo) nos pueda dar también los otros campos del registro. Una forma en que podría abordarse esto sería disponer de un campo suplementario, asociado con el registro únicamente con fines de clasificación. Si un registro fuera, por ejemplo, el decimoquinto del archivo, este campo de clasificación contendría el número 15. Esto podría utilizarse para indicar los elementos en un número de matrices. Para ilustrar este ejemplo, supongamos el registro siguiente:

```
Carmen Montero      campo NOMBRE
Balleneros, 12      campo CALLE
San Sebastián       campo CIUDAD
Guipúzcoa           campo PROVINCIA
(943) 45 72 73      campo NUMERO TELEFONO
017                 campo CLASIFICACION
```

Si supiéramos el nombre de esta persona y quisiéramos conocer su número de teléfono, todo lo que tendríamos que hacer sería buscar el nombre de entre los elementos de la matriz que poseyeran esta información. Encontraríamos, entonces, que el elemento de la matriz en el cual estaba el nombre era el 17. Entonces lo único que faltaría por hacer sería encontrar el elemento decimoséptimo en la matriz NUMERO TELEFONO para obtener así el número de teléfono correcto.

Si tuviéramos varios amigos en la zona del Valle de Arán, es posible que un día quisiéramos utilizar el programa para buscar los datos de los que vivieran en la localidad de Viella, incluida en el campo CIUDAD. El programa, entonces, examinaría todos los campos CIUDAD y anotaría la localización de cada dato de Viella. Luego lo único que restaría por hacer, para que se imprimieran los nombres y direcciones de todos estos amigos, sería extraer todos los elementos que tengan el mismo número, pertenecientes a todas las matrices de cada registro "Viella". Al emplear este método, no sería necesario inspeccionar el campo CLASIFICACION. Esta técnica, además, cuenta con la ventaja de ser relativamente sencilla de realizar.

En el próximo capítulo de nuestro curso de programación estudiaremos algunos de los problemas que se presentan al buscar datos específicos a través de listas.

Ejercicio

■ Supongamos que unos registros que contengan los siguientes campos fueran adecuados para nuestra agenda computerizada:

```
campo NOMBRE
campo CALLE
campo CIUDAD
campo PROVINCIA
campo NUMERO TELEFONO
```

Supongamos que una de las opciones ofrecidas por un menú sea:

5. CREAR UNA NUEVA ENTRADA

Se teclea 5, y el programa se bifurca hacia la sección donde se han creado nuevos registros (se puede suponer que en la agenda ya no quedan entradas). Como el programa está completamente guiado por el menú, siempre se recibirán indicaciones sobre qué entradas hay que efectuar, con instrucciones tales como DAR ENTRADA AL NOMBRE, DAR ENTRADA A LA CALLE, DAR ENTRADA A LA CIUDAD, etc. A continuación se muestra el resultado que cabría esperar:

1. Un elemento en una matriz, para el nombre
2. Un elemento en una matriz, para la calle
3. Un elemento en una matriz, para la ciudad
4. Un elemento en una matriz, para la provincia
5. Un elemento en una matriz, para el teléfono

El problema consiste en desarrollar esto, mediante un proceso de programación *top-down* (de arriba abajo), utilizando un seudolenguaje hasta donde sea posible una conversión directa a BASIC. El seudolenguaje puede seguir las normas que usted le indique; sólo le sugerimos que emplee mayúsculas para palabras clave como IF, LOOP, etc., y minúsculas para descripciones en lenguaje corriente de las operaciones que deben realizarse. (Antes de comenzar el programa conviene examinar detenidamente el recuadro de "Complementos al BASIC".)

Complementos al BASIC

SPECTRUM

Paso 3

```
10 INPUT "DE ENTRADA AL NOMBRE
   COMPLETO";N$
15 LET CONT = 0
20 FOR L = 1 TO LEN N$
30 LET C$ = N$(L)
40 IF C$ = " " THEN LET CONT = L
50 NEXT L
60 PRINT "ULTIMO ESPACIO ESTA EN
   POSICION";CONT
70 STOP
```

En este proyecto de programa, las funciones en serie MID$, LEFT$ y RIGHT$ serán muy utilizadas. En el BASIC del Sinclair sustituir:

```
LEFT$(N$,N)   por  N$( TO N)
RIGHT$(N$,N)  por  N$(LEN(N$)−N+1 TO )
MID$(N$,P,N)  por  N$(P TO P+N−1)
MID$(N$,P,1)  por  N$(P)
```

Téngase en cuenta que los nombres de las variables alfanuméricas en el Spectrum no pueden estar formados por más de una letra (además de "$").

Paso 4

```
 5 LET A$ = " "
10 LET N$ = "PEDRO GONZALEZ"
20 LET CONT = 6
30 FOR L = CONT + 1 TO LEN N$
40 LET A$ = A$ + N$(L)
50 NEXT L
60 PRINT "APELLIDO ES ";A$
70 STOP
```

Paso 5

```
 5 LET C$ = " "
10 FOR L = 1 TO CONT − 1
20 LET T$ = N$(L)
30 LET CAR = CODE T$
40 IF CAR > 64 THEN LET C$ =
   = C$ + CHR$ CAR
50 NEXT L
60 STOP
```

En este fragmento ha surgido el problema que acarrea el designar las variables alfanuméricas con una sola letra: N$ es el equivalente, en el Spectrum, a la variable NOMBRE$; por lo tanto, C$ debe corresponder a la variable NOMB$.

Parte de subrutina X

```
FOR L = 1 TO LEN P$
LET C$ = P$(L)
IF C$ = "." THEN GOTO 1550
NEXT L
```

Parte de subrutina Y

```
FOR Q = 1 TO LIMIT
LET A(L) = P(Q)
NEXT Q
```

VARIABLES

Entre los ordenadores personales más populares, sólo el BBC Micro admite nombres de variables largos, tal como NOMBRE$. El Spectrum permite usar nombres de variables numéricas largos, pero únicamente una letra para nombres de variables alfanuméricas. El Dragon 32, Vic-20 y Commodore 64 pueden utilizar nombres de variables largos, pero sólo los dos caracteres primeros son significativos, por tanto usar NOMBRE$ es válido, pero hará referencia a la misma posición de memoria que NOMB$: tienen iguales los dos primeros caracteres. En el Oric-1, los nombres de las variables no pueden incluir más de dos caracteres (primero una letra, luego un número o una letra), mientras que el Lynx admite sólo una letra, si bien se pueden emplear mayúsculas y minúsculas con significado distinto.

Papel de calco

Las figuras dibujadas en papel pueden trasladarse al ordenador mediante un digitalizador o un tablero de gráficos

Una de las características más importantes de la actual generación de ordenadores personales es su capacidad de crear gráficos. Con unas sencillas instrucciones se pueden crear dibujos y formas y cambiar colores. Todo ello requiere tener conocimientos de programación, puesto que aún no es posible crear una imagen sobre el papel y luego cargarla en el ordenador. Los lápices ópticos (véase p. 156) facilitan la edición y manipulación de una imagen una vez ésta aparece en la pantalla, pero no se pueden utilizar para copiar una figura a partir de una hoja de papel.

Los proyectistas de coches, aeroplanos y microprocesadores, así como los decoradores de interiores, arquitectos de parques y creadores de moda pueden beneficiarse del sistema de gráficos de un ordenador. Una vez que el diseño se ha almacenado de forma segura en la memoria del ordenador, puede intentarse la realización de adiciones y alteraciones sin echar a perder materias primas valiosas. Por lo tanto, será necesario disponer de un dispositivo de entrada que sea capaz de trasladar las líneas y curvas del dibujo o diseño a un lenguaje que pueda ser entendido por el ordenador.

En el mercado profesional, el "tablero de gráficos" se ha estado empleando casi tanto como el ordenador. Sin embargo, sólo desde hace muy poco tiempo es asequible para el usuario del ordenador personal. Los tableros gráficos de alta precisión, también conocidos como "digitalizadores", emplean una vasta gama de técnicas para producir la información requerida. Los sistemas más exactos pueden proporcionar una resolución de imagen de alrededor de 1/4 mm, suficiente para ser usada por ingenieros y delineantes.

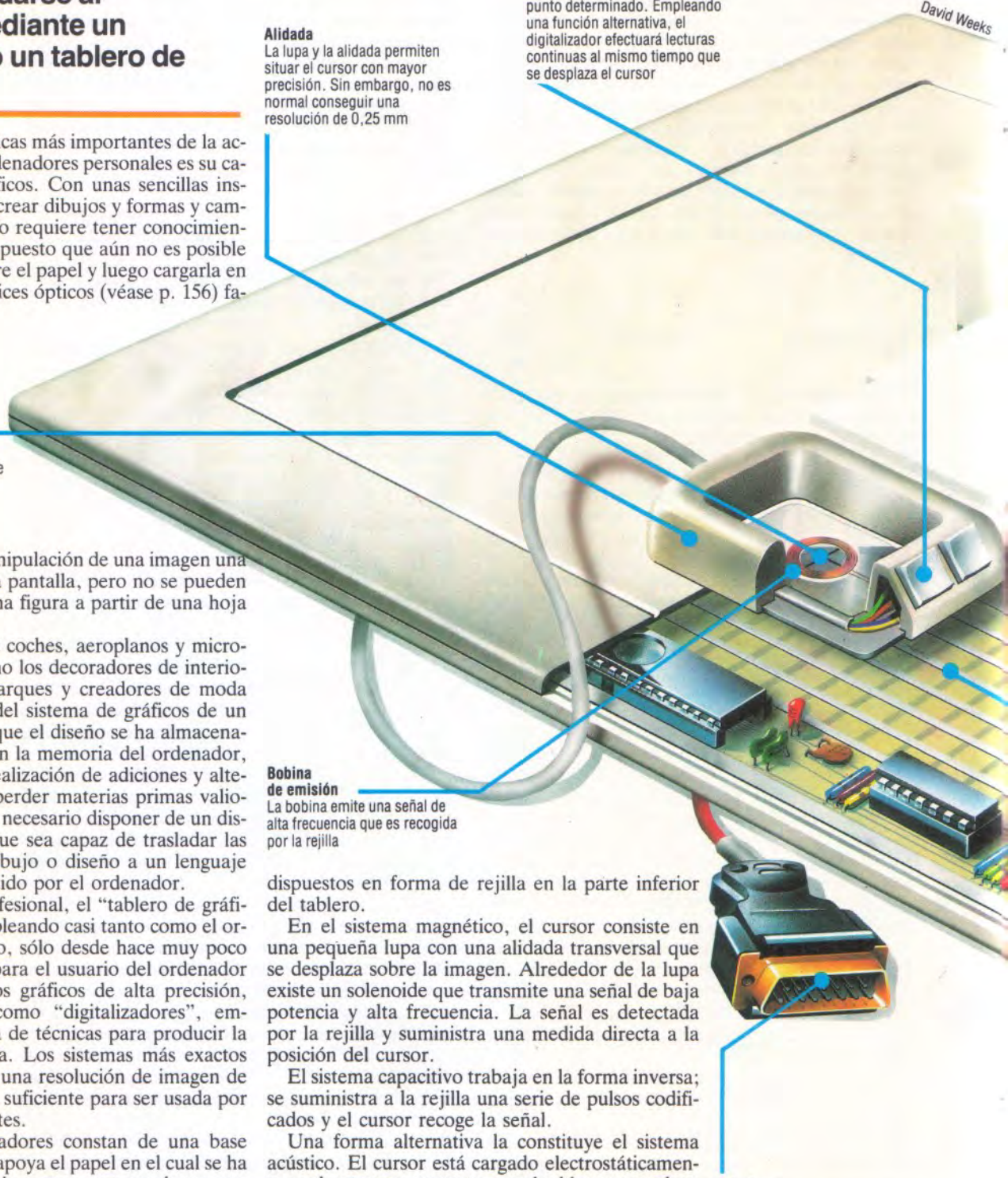
Todos los digitalizadores constan de una base plana sobre la que se apoya el papel en el cual se ha dibujado o pintado. Un cursor, que puede ser una pluma corriente o un dispositivo electrónico sofisticado, es luego desplazado sobre la imagen. La posición del cursor es detectada por el digitalizador y transmitida al ordenador en forma de un cambiante par de coordenadas.

Los dos sistemas más precisos —magnético y capacitivo— funcionan mediante una trama de cables dispuestos en forma de rejilla en la parte inferior del tablero.

En el sistema magnético, el cursor consiste en una pequeña lupa con una alidada transversal que se desplaza sobre la imagen. Alrededor de la lupa existe un solenoide que transmite una señal de baja potencia y alta frecuencia. La señal es detectada por la rejilla y suministra una medida directa a la posición del cursor.

El sistema capacitivo trabaja en la forma inversa; se suministra a la rejilla una serie de pulsos codificados y el cursor recoge la señal.

Una forma alternativa la constituye el sistema acústico. El cursor está cargado electrostáticamente, y al entrar en contacto con el tablero se produce una minúscula chispa. El tiempo que necesita la onda acústica creada por la chispa para alcanzar dos micrófonos proporciona una medida de la posición del cursor. Entre otras características, este sistema ofrece la posibilidad de digitalizar en tres dimensiones por medio de una señal que atraviese el objeto.



Botones entrada datos
La mayoría de cursores poseen más de un botón de desplazamiento, mediante el cual el operador puede indicar que es necesario registrar un punto determinado. Empleando una función alternativa, el digitalizador efectuará lecturas continuas al mismo tiempo que se desplaza el cursor

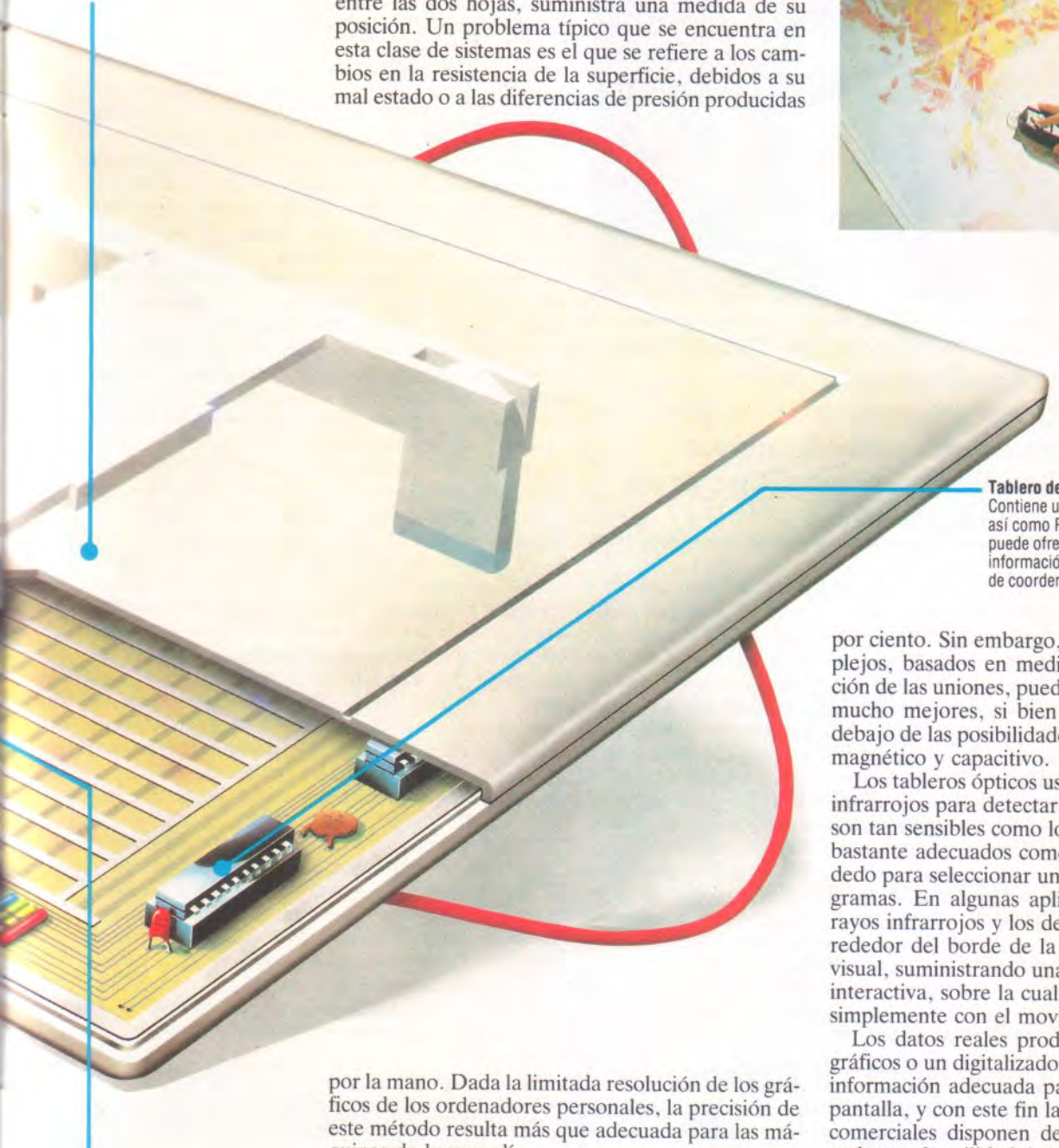
Alidada
La lupa y la alidada permiten situar el cursor con mayor precisión. Sin embargo, no es normal conseguir una resolución de 0,25 mm

David Weeks

Cursor
Este dispositivo es movido a mano para seguir los trazos de la figura que está siendo digitalizada

Bobina de emisión
La bobina emite una señal de alta frecuencia que es recogida por la rejilla

Interface
Normalmente los digitalizadores se acoplan al ordenador mediante una interface conectada en serie o en paralelo



Tablero base
En esta superficie se sitúa la imagen a digitalizar. En algunos sistemas, se aplica al tablero una carga electrostática para que el papel quede completamente plano. Es muy importante que la imagen no se desplace con relación al tablero

Tablero de procesamiento
Contiene un microprocesador, así como ROM y RAM. Por ello puede ofrecer al ordenador información en forma de pares de coordenadas X-Y

Rejilla receptora
Está adosada en la parte inferior del tablero, y puede recoger la señal emitida por la bobina. La anchura de la malla es considerablemente más amplia que la correspondiente a la alta resolución del digitalizador, debido a que el sistema de circuitos de procesamiento es capaz de interpolar la potencia relativa de la señal recogida por los cables de la rejilla

En el extremo inferior de la escala, desde el punto de vista de la precisión, se encuentra el tablero sensible a la presión: en él la figura es contorneada por un cursor. Dos hojas conductoras de electricidad están separadas por un aislante celular; a estas dos capas se las alimenta con dos señales diferentes de alta frecuencia. La señal detectada por el cursor cuando éste efectúa una conexión eléctrica entre las dos hojas, suministra una medida de su posición. Un problema típico que se encuentra en esta clase de sistemas es el que se refiere a los cambios en la resistencia de la superficie, debidos a su mal estado o a las diferencias de presión producidas por la mano. Dada la limitada resolución de los gráficos de los ordenadores personales, la precisión de este método resulta más que adecuada para las máquinas de hoy en día.

Los digitalizadores más baratos y sencillos son los pantógrafos, basados en el anticuado sistema de dibujo formado por dos brazos articulados. Los pantógrafos utilizan valores de coordenadas para proporcionar una medida directa de la posición del cursor. Unas resistencias variables, montadas en las dos uniones, suministran voltajes proporcionales a los ángulos del "hombro" y el "codo" del brazo articulado. La resolución del pantógrafo está limitada por la precisión tanto de las resistencias variables como de las articulaciones mecánicas; el valor normal de ella suele ser sólo de alrededor de un cinco por ciento. Sin embargo, los pantógrafos más complejos, basados en mediciones ópticas de la rotación de las uniones, pueden ofrecer unos resultados mucho mejores, si bien quedan todavía muy por debajo de las posibilidades que ofrecen los sistemas magnético y capacitivo.

Simon Lewis

Trazado de mapas
Uno de los usos profesionales más comunes para los digitalizadores es la recogida de datos de mapas y reconocimiento de terrenos. En la figura, el ordenador se utiliza para localizar campos petrolíferos a partir de datos geológicos digitalizados

Los tableros ópticos usan un entramado de haces infrarrojos para detectar la posición del cursor. No son tan sensibles como los otros sistemas, pero son bastante adecuados como para permitir utilizar un dedo para seleccionar un tema de un menú de programas. En algunas aplicaciones la fuente de los rayos infrarrojos y los detectores están situados alrededor del borde de la unidad de representación visual, suministrando una pantalla verdaderamente interactiva, sobre la cual se pueden dibujar figuras simplemente con el movimiento de un dedo.

Los datos reales producidos por un tablero de gráficos o un digitalizador deben ser convertidos en información adecuada para ser representada en la pantalla, y con este fin la mayoría de los productos comerciales disponen del software necesario. Sin embargo, la utilidad de los tableros gráficos va más allá de la mera introducción de datos. Una vez la información ha sido almacenada en el ordenador, el tablero puede usarse como herramienta de montaje, permitiendo añadir o cambiar colores y modificar formas. La superficie del tablero puede ser programada para actuar como un menú que selecciona opciones estándar de un programa, de forma que sólo sea necesario usar el teclado para seleccionar las funciones principales. Los sistemas de animación con ayuda de ordenador (véase p. 181) tienen un tablero de gráficos de alta calidad como componente principal de su dispositivo de entrada.

Gottfried Leibniz

1646
Nace el 1 de julio, en Leipzig

1661
Se matricula en la Universidad de Leipzig y se gradúa a los 17 años

1661-1670
Trabaja como abogado y diplomático. Publica un ensayo sobre "El arte de la combinación"

1672
Desarrolla en París el principio de la "razón suficiente"

1673
Presenta en la Royal Society la máquina calculadora

1675
Realiza estudios sobre la gravedad independientemente de Newton

1676
Aportación a la dinámica mediante el concepto de energía cinética

1678
Es nombrado bibliotecario y consejero del duque de Hannover

1679
Desarrolla la teoría del sistema binario

1683
Publica el panfleto "El más cristiano dios de la guerra", un ataque a Luis XIV

1690-1699
Su estudio genealógico de la Casa de Hannover se amplía a una Historia del mundo

1700
Organiza la Academia de las Ciencias de Berlín

1714
Se le designa para establecer el derecho de sucesión de Jorge I al trono vacante de Inglaterra

1716
Muere el 14 de noviembre, en Hannover

Cortesía Science Museum

Los científicos que trabajan en la creación de la quinta generación de ordenadores muestran cada vez mayor interés en las investigaciones de este pensador

Gottfried Wilhelm Leibniz fue la figura científica principal de su tiempo —el período conocido como el Siglo de las Luces—. Nació en la ciudad centroeuropea de Leipzig en 1646, y murió en Hannover en 1716. Durante sus setenta años de vida (el tipo de número exacto que se puede esperar de un matemático), formuló los principios del cálculo infinitesimal, realizó estudios sobre la dinámica y contribuyó con valiosas aportaciones a los campos de la geología, teología, historia, lingüística y filosofía. Y lo más importante para nosotros: desarrolló teorías básicas para la creación del ordenador.

Empezó a viajar a la edad de veinte años, después de que la Universidad de Leipzig rehusara concederle el doctorado en leyes debido a su juventud. Carente de recursos propios, Leibniz se vio obligado a desempeñar trabajos que interferían con sus investigaciones científicas. Con poco más de veinte años fue abogado y diplomático; más tarde fue bibliotecario y consejero de la familia real.

Los intereses de Leibniz eran muy amplios y su naturaleza cosmopolita le condujo a viajar frecuentemente por Europa, manteniendo conversaciones y contactos con todos los grandes pensadores de su tiempo. Era un prolífico escritor de cartas: mantuvo correspondencia con más de 600 personas.

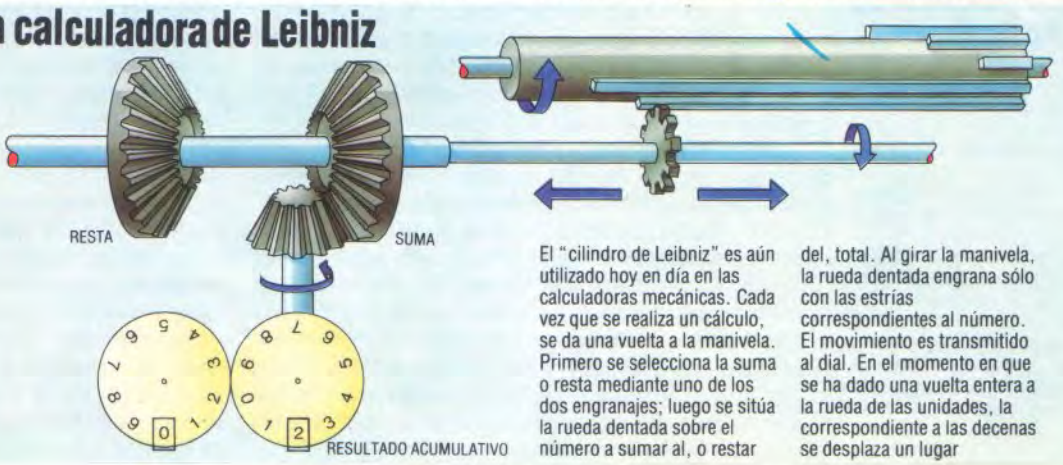
Su primera contribución importante a la filosofía se produjo en 1672, cuando formuló el principio de la "razón suficiente". Éste sostenía, simplemente, que tiene que haber una razón para todo, y "todo conduce al bien en el mejor de los mundos".

Centrando su interés en las matemáticas, empezó a trabajar en el perfeccionamiento de la máquina de sumar de Blaise Pascal, inventada en 1642 (véase p. 86). Leibniz intentó mejorarla de forma que fuera capaz de multiplicar y dividir. Lo logró mediante un dispositivo mecánico llamado *cilindro de Leibniz* (véase la ilustración en la parte inferior de la página). El mecanismo de Leibniz fue un adelanto decisivo para su tiempo. Antes, debido a la complejidad de multiplicar con números romanos, esta operación aritmética se enseñaba sólo en los centros de estudio de mayor categoría. Una máquina que pudiera multiplicar mecánicamente la volvía más accesible. Después de haber perfeccionado esta máquina, Leibniz centró sus esfuerzos en la creación de un método que permitiera convertir el sistema decimal en otro de base binaria.

La mayor ambición del filósofo y matemático alemán era idear un lenguaje universal que pudiera emplear la claridad y precisión de las matemáticas para resolver cualquier problema que se le pudiera presentar a la humanidad. Su lenguaje debería utilizar símbolos abstractos para representar los "átomos" fundamentales del entendimiento, con un conjunto de reglas para manejar estos símbolos. Su intento resultó fallido; pero sus ideas fueron recogidas a principios del siglo xx por Bertrand Russell, quien intentó explicar las matemáticas en términos de "lenguaje" lógico formal.

En los últimos años, se ha visto incrementado el interés en las investigaciones de Leibniz por parte de los científicos que trabajan en el proyecto a largo plazo de crear la quinta generación de ordenadores. Estas máquinas, según se cree, podrán resolver cualquier problema humano con la misma velocidad y fiabilidad con que los ordenadores actuales ejecutan cálculos matemáticos. Para ello será preciso un tipo de lenguaje totalmente nuevo.

La calculadora de Leibniz



El "cilindro de Leibniz" es aún utilizado hoy en día en las calculadoras mecánicas. Cada vez que se realiza un cálculo, se da una vuelta a la manivela. Primero se selecciona la suma o resta mediante uno de los dos engranajes; luego se sitúa la rueda dentada sobre el número a sumar al, o restar del, total. Al girar la manivela, la rueda dentada engrana sólo con las estrías correspondientes al número. El movimiento es transmitido al dial. En el momento en que se ha dado una vuelta entera a la rueda de las unidades, la correspondiente a las decenas se desplaza un lugar

Kevin Jones

Ordenador del futuro

El Lisa de Apple es el ordenador de gestión más innovador que existe en el mercado, y muchas de sus configuraciones aparecerán finalmente en los ordenadores personales

El Lisa, fabricado por Apple Computer, es un ordenador destinado exclusivamente a ser utilizado con fines de gestión empresarial. Vale dos millones de pesetas, sin incluir la impresora. El lector tal vez se pregunte por qué se le otorga tanta importancia en esta obra a una máquina tan cara. La razón por la que le hemos concedido tanta atención se explica por el hecho de que el Lisa está tan adelantado a su época, que es seguro que muchas de sus configuraciones finalmente se aplicarán a los futuros ordenadores personales. Se sabe que la Apple ya está trabajando con versiones a escala reducida del proyecto, y los fabricantes de la competencia se apresuran en imitar las características de este innovador ordenador.

No es el hardware del Lisa lo que es tan radicalmente nuevo, sino más bien el software estándar que viene con él. Desarrollar software sofisticado es, de hecho, una tendencia en la que en los momentos actuales coinciden todos los microordenadores. Ahora, diseñar y construir un nuevo tipo de microordenador exige menos horas de trabajo que escribir un programa sofisticado de gestión o de juego recreativo. El software se está convirtiendo en el elemento más importante de cualquier sistema informático, así como en el más caro. En la actualidad los usuarios de ordenadores personales se encuentran con que, en el transcurso de un año, han invertido en la adquisición de programas en cassette y en cartuchos la misma cantidad de dinero que destinaron a la compra de la máquina.

No obstante, primero analizaremos el hardware del Lisa, cuyo diseño fue dictado fundamentalmente por los requerimientos del software. El Lisa viene con un megabyte de RAM como memoria estándar (es decir, mil veces la de un ZX81). Tal enorme cantidad de memoria exige que el microprocesador dedique mucho tiempo a la función de "administración de memoria": desplazando los datos por la memoria y llevando el registro de dónde se encuentra cada elemento. El procesador es un Motorola 68000, que es un dispositivo de 16 bits (esto significa que es capaz de procesar 16 bits de datos simultáneamente, mientras que la mayoría de las CPU de los ordenadores personales manipulan 8). Desde el punto de vista de los ordenadores personales, se trata de un procesador muy rápido, con un conjunto de instrucciones muy avanzado. Para el almacenamiento permanente, el sistema Lisa incluye dos unidades de disco flexible y una unidad de disco rígido separada y con pocas características externas. El disco rígido es necesario tanto por su capacidad (cinco megabytes) como por su velocidad: el Lisa hace uso de una gran cantidad de programas que frecuentemente se necesita intercambiar entre la RAM y el disco.

Otra característica notable del hardware del Lisa es la visualización monocromática incorporada, que posee una resolución de 720 × 364 pixels. Ello permite una variedad de caracteres distintos para el texto, así como el tipo de gráficos que vemos en este artículo. El diseño del Lisa incorpora chips y sistemas de circuitos especiales destinados exclusivamente a accionar esta visualización y a mover rápidamente las imágenes.

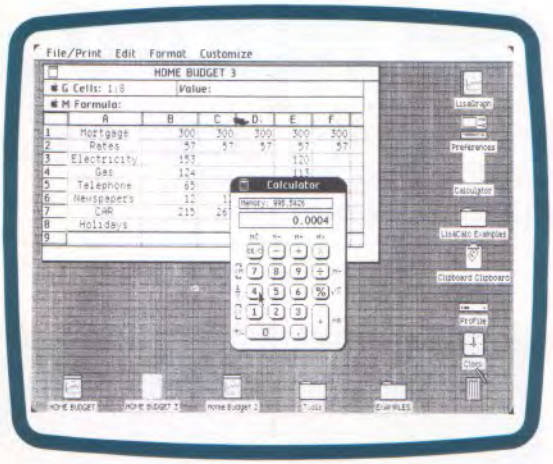
Ian McKinnell

Facilidades para el usuario
El Lisa de Apple se diseñó para que lo emplearan personas del campo empresarial que no tuvieran ninguna experiencia previa con ordenadores. La utilización de un "ratón" manual determina que se prescinda del teclado con muchísima mayor frecuencia que en el caso de otros sistemas

En conexión con una impresora adecuada (como una unidad matricial de gran velocidad y gran calidad), se puede reproducir en papel todo cuanto se visualiza en pantalla. Sin embargo, aunque la impresora no fuera compatible con el alto nivel de resolución de la pantalla, el Lisa produciría una imagen impresa de la mejor calidad.

El teclado del Lisa se puede separar de la unidad principal y tiene un buen trazado. No obstante, se utiliza con muchísima menos frecuencia que el de otras máquinas, porque el Lisa posee un "ratón". Un "ratón" representa una de las diversas maneras de dar entrada a información en la pantalla sin necesidad de valerse del teclado; otros métodos son la palanca de mando, el lápiz óptico y las unidades para reconocimiento de voz. En esencia, el "ratón" es una pequeña caja que se mueve a través de la superficie de su mesa o escritorio y que se conecta

La imagen perfecta
Toda función que realice con el Lisa se representa mediante un símbolo denominado "icono". Para activar la función, se mueve el "ratón" hasta que el cursor quede situado sobre el icono, y se pulsa el botón SELECT del "ratón". Entonces "se descubre" la aplicación, visualizándose en la pantalla con todo detalle



al ordenador mediante un cable colgante. El movimiento del "ratón" origina el desplazamiento de un "cursor" o indicador alrededor de la pantalla. De este modo se puede señalar en ésta la información o la orden que usted requiera y luego, pulsando el botón SELECT de la parte superior del ratón, seleccionar esa información o ejecutar esa orden. Sólo es necesario usar el teclado cuando se han de entrar nuevos datos en forma de texto o números.

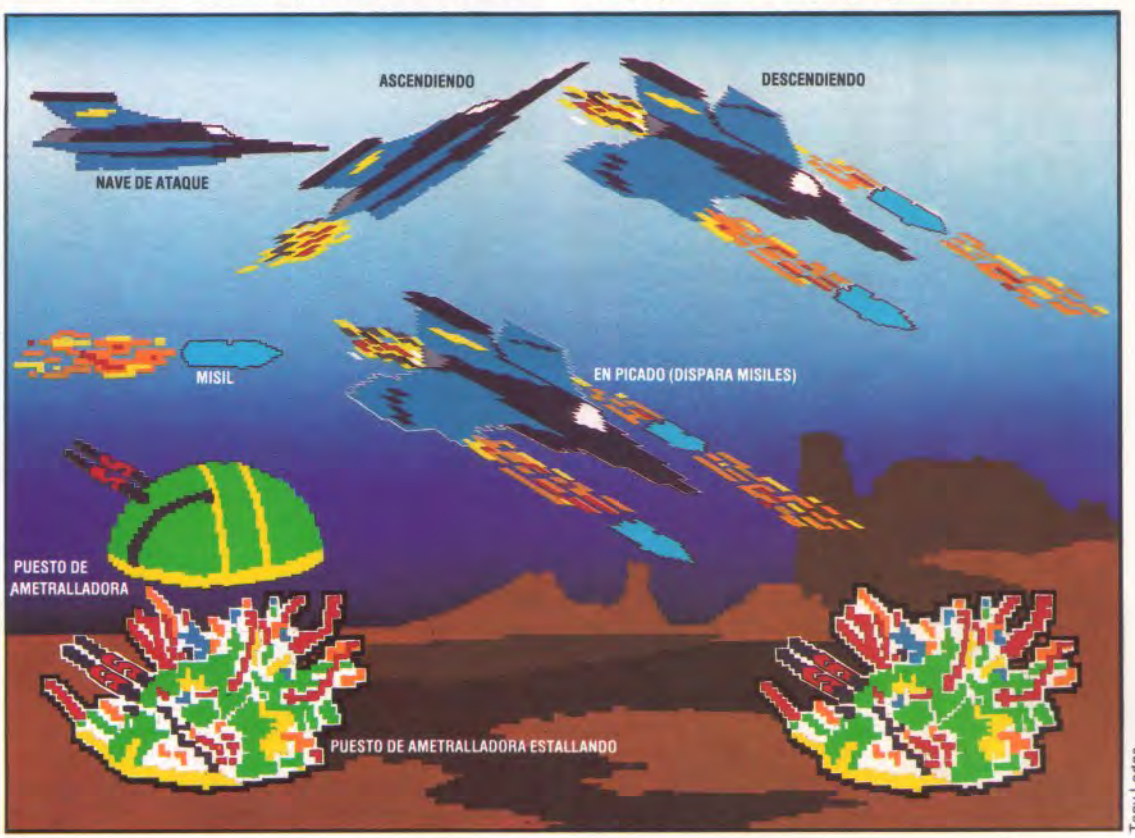
Llegados a este punto, ya estamos preparados para empezar a analizar el software del Lisa. De nuevo deseamos resaltar que, si bien sus aplicaciones se inscriben por completo dentro del campo de la gestión empresarial, los principios sobre los cuales funciona se introducirán con el tiempo en las configuraciones para ordenadores personales.

Cuando se conecta el Lisa por primera vez, la imagen que muestra la pantalla corresponde a la superficie de un escritorio con distintos objetos dispuestos sobre él. De hecho, casi todo lo que usted realice en el Lisa se representa visualmente en la forma en que lo llevaría a cabo si no contara con la ayuda de un ordenador. Ésta es la razón fundamental por la cual a los principiantes les resulta mucho más fácil llegar a familiarizarse con el Lisa que con el hardware y el software convencionales. Los objetos ordenados sobre el escritorio se denominan *iconos*; cada uno representa una función determinada, que por lo general aparece escrita debajo de él.

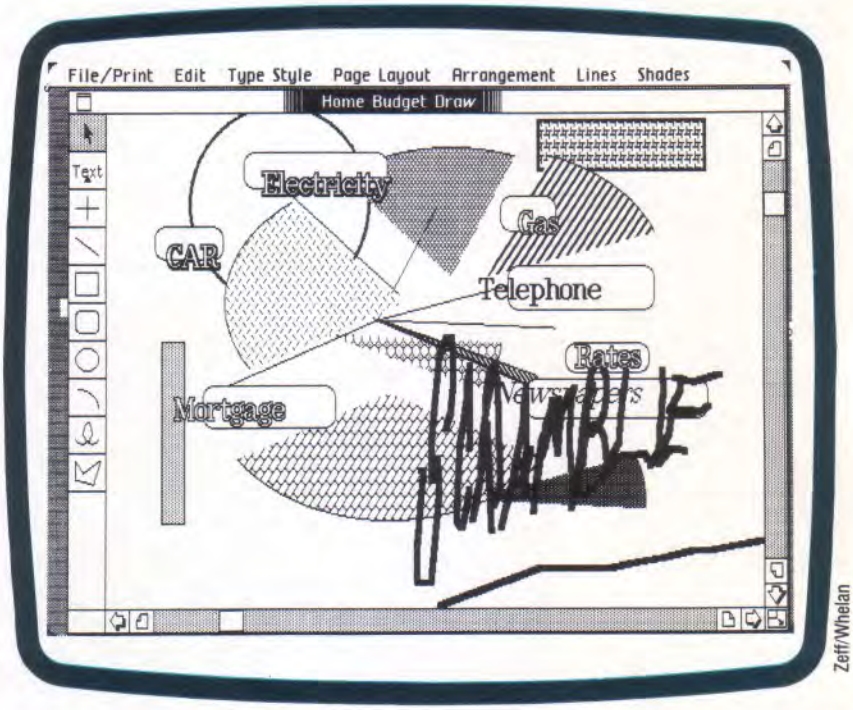
Tomemos como ejemplo el icono del reloj. Desplazando el cursor sobre la pequeña representación horaria mediante el "ratón" y pulsando el botón SELECT, se visualizará en la pantalla un reloj de tamaño mucho mayor, junto con la fecha del día. Si no desea que el gran reloj le desordene el escritorio, sencillamente lo puede "reducir" otra vez a su tamaño original. Del mismo modo, seleccionar el icono en forma de calculadora hará que aparezca una más grande, que se puede utilizar para resolver operaciones aritméticas sencillas. Si no encuentra satisfactoria la disposición de los iconos sobre el escritorio, con sólo pulsar el botón SELECT y desplazar el "ratón", puede hacerlos cambiar de lugar en la pantalla. Uno de los rasgos más graciosos del Lisa y que ilustra hasta qué punto este ordenador está modelado de acuerdo con nuestros hábitos de trabajo, es el icono que representa el cubo de la basura. Si ya no necesita una parte de la labor realizada, puede simplemente tirarla al cubo de la basura empleando el "ratón". Procediendo así es muy difícil que por accidente se borre alguna información importante. Incluso se puede examinar lo que contiene en ese instante el cubo y recuperarlo, ¡a menos que el ordenador haya sido desconectado!

En el Lisa la mayor parte del trabajo real se realiza utilizando uno de los siguientes seis sistemas aplicativos: LisaWrite, un procesador de textos; LisaCalc, una hoja electrónica; LisaGraph, un sis-

Programación orientada a un objeto
Para crear un juego recreativo al estilo del *Defender* empleando una programación convencional, sería necesario diseñar cierta cantidad de muestras de trazados en pantalla y luego, partiendo de cero, escribir un programa que controlara todo el juego. Utilizando la "programación orientada hacia un objeto" (lo cual por el momento es muy difícil de realizar en ordenadores personales porque no existen lenguajes de programación adecuados), el usuario, en cambio, se concentraría en cada elemento del juego individualmente.
Empezando por la nave de ataque, su definición enunciaría que ésta se desplaza siempre de izquierda a derecha; que cuando la palanca de mando se acciona hacia adelante o hacia atrás la nave se mueve, en consecuencia, hacia arriba o hacia abajo; y que cuando se pulsa el botón FIRE, se lanza un misil. Su segunda definición es la relativa al misil. Define su forma y enuncia que continúa en una dirección hasta que entra en contacto con otro objeto, en cuyo punto desaparece. El puesto de ametralladora estática se define mediante una forma sencilla, que se cambia por una imagen de explosión si un misil entra en contacto con ella.
¡Exprese estas tres definiciones en un lenguaje adecuado, colóquelas juntas en serie y el juego estará escrito más o menos para usted!



Tony Lodge



Zeff/Whelan

Pasando la información
Una de las configuraciones distintivas del Lisa permite pasar la información de una aplicación a otra mediante la opción COPY. Ésta almacena temporalmente la información en el icono de la tablilla con chips, disponiéndola para "pegarla" en otra ventana. Por ejemplo, podríamos partir del análisis de algunas cifras utilizando LisaCalc (la aplicación para hoja electrónica). Las cifras resultantes estarían en condiciones luego de ser copiadas en LisaGraph, que produciría un diagrama o gráfico de barras de las cifras automáticamente. Por último, la imagen completa se podría transferir a LisaDraw, que nos permitiría adornarla con etiquetas, flechas, diagramas u otros detalles adicionales. Luego se podría imprimir el resultado final

tema para trazado de gráficos; LisaList, un administrador de base de datos; LisaProject, un medio auxiliar para planificación de proyectos; y LisaDraw, una sofisticada herramienta para crear toda clase de imágenes gráficas. Los iconos para estas aplicaciones son simplemente blocs de papel. Por ejemplo, para efectuar un cálculo por hoja electrónica se coloca el cursor sobre el bloc de papel dividido en líneas y columnas denominado LisaCalc. En efecto, al pulsar el botón SELECT se "arrancará" una hoja de este papel, que se colocará después en algún lugar del escritorio y se le podrá colocar la etiqueta de "planes de venta", por ejemplo.

En realidad el usuario puede optar por tener simultáneamente sobre su escritorio diversas aplicaciones de hoja electrónica a la vez, volviendo a seleccionar el mismo icono. En un sistema por ordenador normal, habría de pasar por el proceso de cargar el programa de hoja electrónica y especificar luego el archivo de datos sobre el que desea trabajar. En el Lisa, sin embargo, el programa y los datos son inseparables. Éste es otro ejemplo de programación orientada hacia un objetivo, que hemos introducido al analizar el Pinball Construction Set (véase p. 241).

Otra característica importante del *entorno operativo* (así se lo denomina) del Lisa es su capacidad para "mirar por la ventana". Cuando se selecciona una aplicación, ésta aparece como una hoja grande de papel dispuesta sobre el escritorio. Las dimensiones de este trozo de papel también se pueden especificar, utilizando el "ratón". Si en un momento dado hay "abierta" a la vez más de una de estas aplicaciones, las hojas se ordenarán una sobre otra, visualizándose en la parte superior la hoja en la que está trabajando en ese momento, tal como ocurriría de estar trabajando en un escritorio. Podría darse el caso de que la aplicación sobre la que esté trabajando, por ejemplo el procesador de textos, requiriera más espacio que el disponible de acuerdo a las dimensiones de la hoja que usted hubiera especificado. En ese caso, la hoja actúa sólo como una ventana a la aplicación y se puede mover alrededor para visualizar cualquier parte del documento total.

Se puede trasladar la información desde una aplicación a otra, empleando otra vez los iconos, y éste es otro de los puntos fuertes del Lisa. Supongamos que usted está haciendo un análisis de sus cifras de ventas mensuales utilizando el LisaCalc. Valiéndose de la función COPY, que se selecciona a partir de un menú de funciones especiales listadas a lo largo de la parte superior de la pantalla, puede hacer una copia temporal de los resultados de la hoja electrónica, que se almacenan en el icono que representa una tablilla con chips. Luego, seleccionando un trozo de papel LisaGraph, se puede dar entrada a esos resultados en la sección INPUT DATA de la aplicación para gráficos, sencillamente seleccionando del menú la opción PASTE. Si se le requiriera, LisaGraph produciría luego un pulcro diagrama (o gráfico de barras y líneas), completamente etiquetado y sombreado. Ahora, utilizando nuevamente las opciones COPY y PASTE de la parte superior de la pantalla, se puede copiar esta imagen en un trozo de papel LisaDraw. Esta última aplicación permitiría entonces adornar el diagrama con algunas flechas, o modificar las etiquetas y los encabezamientos según una variedad de tipos diferentes. El resultado final se podría así imprimir y copiar para emplearlo, por ejemplo, como una ilustración para un reportaje o un artículo para alguna revista.

Como hemos dicho, tanto los principios de la programación orientada hacia un objetivo como los de los entornos operativos al estilo Lisa, pronto comenzarán a introducirse incluso en las máquinas más económicas, especialmente en la medida en que éstas se vuelvan más sofisticadas en términos de velocidad de procesamiento y dimensiones de la memoria RAM. Imagínese que en la pantalla de su ordenador personal tuviera múltiples ventanas: podría escribir un programa en una de ellas y observar en otra su salida. Luego podría llamar a otros medios auxiliares de programación con sólo señalar un icono con forma de caja para herramientas, y mover las subrutinas a través de su listado simplemente desplazando el "ratón".

Ventanas al mundo

El videotex es uno de los pocos campos de la informática en que se han fijado estándares internacionales. Ello permite a cualquier ordenador acceder a una red mundial de bases de datos

Los sistemas de videotex, como el servicio Prestel de la British Telecom, permiten tener acceso a una diversidad de bases de datos por medio de un televisor adaptado y un teléfono. Un sistema puede ser de naturaleza pública, abierto y accesible a todos, como el Prestel; puede estar abierto a todos pero tener un contenido más bien especializado, no comprensible a cualquier usuario (como el Fintel, una base de datos financieros que se actualizan de forma constante), o, finalmente, ser de naturaleza privada.

No obstante, los sistemas de videotex no se deben confundir con los sistemas de teletexto que están ahora disponibles a través de los canales de las emisoras de televisión, que utilizan juegos de caracteres y trazados de páginas similares. Los sistemas de teletexto, como el Ceefax de la BBC o el Oracle de la ITV (cadena privada de televisión), se emiten como señales subsidiarias superpuestas a los programas de televisión normales. Las señales que definen a una página de teletexto se transmiten entre los fotogramas de una emisión corriente de televisión. Un dispositivo decodificador las separa y crea la imagen de teletexto. El texto y los gráficos resultantes pueden desplazar a la imagen de emisión normal o se pueden superponer a ella de manera que aparezcan las dos juntas, como cuando se utiliza el teletexto para proporcionar subtítulos para no interferir el sonido.

Los sistemas de teletexto no son interactivos, es decir, el espectador no tiene forma alguna de modificar los contenidos de una página de información, ni siquiera de dar una respuesta. Sin embargo, el acceso a ellos es gratuito. Lo único que se necesita para acceder a un gran volumen de información útil es un televisor convenientemente adaptado.

Los sistemas de videotex, por el contrario, utilizan el televisor doméstico simplemente como un monitor para visualizar los datos recibidos a través de la red telefónica. Normalmente, los usuarios no pueden modificar la base de datos, pero pueden utilizar su teclado para dar entrada a preguntas en el sistema de respuestas. Todos los sistemas de videotex son activados por menú, es decir, siempre se visualiza la gama de opciones de que dispone el usuario a cualquier nivel. El usuario realiza su elección dando entrada a un número en el teclado numérico. Éste transfiere el control del submenú seleccionado; y el procedimiento se sigue repitiendo a través de los niveles jerárquicos hasta que el usuario llega a la página de información deseada.

Cada página se identifica mediante un número exclusivo y se puede llegar directamente a una página determinada dando entrada a este número. Éste es el método más rápido (y más barato) para aquellas personas que consultan con frecuencia una misma página dedicada a tratar cierto tema.

A modo de ejemplo, sigamos los pasos que daría un usuario hipotético para hacer la reserva de sus vacaciones de verano vía Prestel. El cuadro de *sign-on* (primera imagen que se ve cuando se conecta el sistema) le indica que pulse el símbolo # en el teclado para llegar al índice principal. A partir de entonces el usuario sigue las indicaciones del menú página a página. La primera parada es en el cuadro de Información General. La cuarta entrada de esta página está encabezada por la leyenda Vacaciones, transporte, viaje, de modo que digita 4.

En respuesta se le ofrecerá una opción entre Viaje en tren, Viaje en avión, Viaje en autocar, Otros transportes, Vacaciones y turismo y Coches y vehículos motorizados. También hay cuatro "salidas", cada una de las cuales lleva a un cuadro de información (una

Cortesia de Prestel

Información pública
Al servicio de videotex Prestel de la British Telecom, que consta de más de 250 000 páginas de información, se puede acceder desde todos los rincones de Gran Bretaña, incluso desde un teléfono público en ciertas condiciones. Todo lo que se necesita para consultar la base de datos es una línea telefónica y un receptor de televisión

Cortesia de Prestel

Estado de cuentas en pantalla
La Nottingham Building Society, en asociación con el Bank of Scotland y la British Telecom, han desarrollado un sistema que les permite a los suscriptores controlar muchas de sus operaciones financieras directamente desde sus propios hogares, utilizando un adaptador Prestel que les proporciona el consorcio. Los clientes pueden examinar el estado de sus cuentas bancarias y de la entidad financiera, pagar facturas, comunicarse con otros suscriptores e incluso comprar una serie de bienes y servicios

especie de desvío, como las notas a pie de página de un libro) desde los cuales se puede regresar al cuerpo principal del texto. Después de pulsar la tecla correspondiente a Vacaciones y turismo, el futuro viajero ha de decidir si reservará el viaje y el alojamiento por separado o si optará por un programa de vacaciones completo, y esa decisión determinará su ruta de salida de la página. Como método alternativo para seleccionar la página, la Prestel también ha publicado una guía, a partir de la cual se puede efectuar una selección inmediata. Por ejemplo, se le proporciona al usuario una referencia directa de la página que incluye una relación de líneas aéreas o una cadena de hoteles.

Concebido en los laboratorios de investigación de la British Telecom en 1971, el primer sistema de videotex entró en funcionamiento hacia 1976 de forma experimental y quedaría a disposición del público bajo la denominación de Prestel a fines de 1979. British Telecom originalmente había llamado Viewdata (videotex) a su sistema, pero al estipularse legalmente que videotex era un término genérico, se vio obligada a cambiarlo por Prestel.

El videotex representa un caso especial en el sentido de que desde su comienzo se convirtió en un estándar aceptado en todo el mundo. Diversos fabricantes de ordenadores y empresas de telecomunicaciones empezaron a producir sistemas propios que utilizaban los protocolos y las estructuras de información del Prestel. Consecuencia de ello ha sido la creación de una red internacional de bases de datos locales, a las que puede tener acceso cualquier suscriptor.

Los sistemas de videotex exigen una línea telefónica abierta durante todo el tiempo que esté en funcionamiento, y el usuario ha de hacer frente a este gasto además del que representa cualquier conexión con el servicio de videotex. Además, existe la posibilidad de una tercera cantidad a pagar por el usuario, determinada por la fuente de datos a que ha tenido acceso, pero este cargo queda supeditado a la voluntad del proveedor de la información. También hay que pagar una pequeña suscripción anual. En un intento por reducir los gastos del usuario, Prestel ha instalado cierto número de "concentradores locales" (líneas troncales telefónicas exclusivas para el sistema) que permiten que un suscriptor que llame desde, por ejemplo, Glasgow a un número de Londres pague la misma tarifa que si se tratase de una llamada local.

El hardware que se requiere para utilizar los servicios de videotex se divide en tres tipos principales. El más sofisticado, y también el más caro, es el terminal de videotex construido especialmente. La mayor parte de los usuarios comerciales lo prefieren. La segunda alternativa consiste en acoplarle un adaptador a un televisor doméstico. Existen diversos adaptadores de este tipo, que van desde un sencillo teclado numérico con dial manual como el de un teléfono, hasta un teclado estilo máquina de escribir, con dial totalmente automático. Pero en el corazón de todos estos dispositivos hay un microordenador exclusivo que decodifica las señales que entran y salen. El tercer método, popular entre los usuarios de microordenadores personales, consiste en comprar software para videotex para un micro estándar. Esta última alternativa viene resultando particularmente atractiva desde la introducción de un servicio conocido como Micronet 800, que les ofrece a los suscriptores la posibilidad de cargar programas para ordenador directamente a través de la línea telefónica. Más aún, estos adaptadores permitirán guardar en disco una página de Prestel, eliminando así la necesidad de una conexión telefónica abierta de manera constante.

Prestel también les permite a los suscriptores enviarse mensajes entre sí gracias a un servicio denominado Mailbox. El suscriptor, al encender su terminal Prestel o, en el caso de que la estuviera usando, al terminar una llamada, recibe la notificación de que hay un mensaje esperando. No es necesario, como se podría pensar, tener acceso a un teclado alfanumérico completo para enviar un mensaje. Por otra parte, se encuentran a disposición del usuario una serie de "modelos de mensaje" que se pueden completar valiéndose sólo de números.

A fines de 1983 había en Gran Bretaña alrededor de 35 000 suscriptores de Prestel, que disponían de más de 250 000 páginas de información; además, un cierto número de empresas y organizaciones utilizaban el videotex para sus propias consultas de base de datos interna.

Red de distribución

El servicio Micronet 800 que funciona en Gran Bretaña, les permite a los suscriptores adquirir software directamente del sistema informático central de Prestel. Para utilizar el sistema es necesario recibir las señales de videotex en un microordenador convencional (y no tan sólo un televisor normal adaptado) y almacenar luego el programa, ya sea en cinta de cassette o en disco, desde donde posteriormente se puede cargar y ejecutar siguiendo los procedimientos normales. El Micronet ofrece alrededor de cien programas libres de cargo, y ofrece muchos más de acuerdo a las tarifas comerciales. Los suscriptores de Micronet también tienen acceso automático al resto de la base de datos de Prestel

Cortesía de British Telecom

Agencias de viajes
Una de las aplicaciones comerciales más populares del Prestel se realiza en las agencias de viajes. Las líneas aéreas, en particular, poseen sistemas muy sofisticados de reserva y venta de billetes, si bien ambos funcionan en ordenadores separados. El Prestel les permite a los agentes acceder a la mayoría de estos sistemas y, también, vender billetes y reservar plazas directamente. Además, tal como se aprecia en el grabado de la izquierda, se puede obtener información acerca de las llegadas y salidas en los aeropuertos de Gran Bretaña, así como también de los cambios en los horarios

Juegos postales

No todos los juegos por ordenador requieren tiempos de reacción de fracciones de segundo: en los juegos postales, un movimiento puede durar seis semanas, participando docenas de jugadores

Aunque la mayoría de los nuevos compradores de ordenadores personales declaran que lo hacen con la intención de aprender a programar, no cabe la más mínima duda de que la aplicación más popular para este tipo de máquinas son en realidad los juegos. Tal como hemos comentado en numerosas ocasiones, los juegos por ordenador pueden contribuir tanto a la diversión como a la educación. El usuario no tiene por qué limitarse a los juegos recreativos como los "marcianitos". El ordenador ha planteado una nueva serie de conceptos relativos al juego, como los de aventuras (véase p. 161) y los educativos (véase p. 81), por no hablar de las versiones informatizadas de populares juegos de tablero.

Sin embargo, existe un tipo de juegos por ordenador que no hemos mencionado hasta ahora y del que es posible que usted ni siquiera haya oído hablar. Estableciendo un marcado contraste con los juegos recreativos, que exigen una sincronización y unos reflejos de fracciones de segundo, estos juegos se realizan a una velocidad claramente pedestre, ¡teniendo que esperar semanas para poder efectuar el movimiento siguiente! Y, a diferencia de la mayoría de los otros juegos por ordenador, que son sólo para un jugador, en éstos pueden participar varias docenas de jugadores a la vez.

Nos estamos refiriendo a los juegos postales, es decir, juegos en los que los participantes están diseminados por todo el país (o, en el caso de los juegos internacionales, a lo largo y a lo ancho del mundo) y en los que cada jugador especifica sus movimientos sobre el papel a intervalos predeterminados. Los movimientos se envían por correo a un coordinador de juego, quien los alimenta a un microordenador (la mayoría de los juegos postales no exigen que el jugador posea su propio ordenador personal). El coordinador envía luego una salida impresa del ordenador a cada jugador, que ilustra su propia posición y otras informaciones relacionadas con ella, como puede ser la posición de otros jugadores con los cuales haya entrado en contacto.

Estos juegos por lo general continúan durante muchos meses, cuando no indefinidamente, si bien los jugadores pueden incorporarse a ellos o abandonarlos en el momento que lo deseen. Se suele cobrar una tarifa de incorporación para cubrir los materiales iniciales y el reglamento del juego; después se cobra una tarifa de juego por cada movimiento (en Gran Bretaña por lo común es de 1 libra, o sea, unas 225 pesetas). ¡Estos juegos no son un pasatiempo barato! Un movimiento por semana se considera un juego rápido, mientras que los juegos internacionales pueden implicar una espera de seis semanas entre uno y otro turno.

Un típico juego postal puede contar con varias docenas de jugadores. Si hay más personas interesadas en suscribirse, el coordinador simplemente comienza un segundo juego paralelo al primero, utilizando los mismos programas pero distintos discos de datos en su ordenador.

Los juegos postales existían mucho tiempo antes del nacimiento del ordenador: se trataba de campeonatos postales de ajedrez. Incluso el popular juego de mesa *Diplomacy* (en el que los jugadores representan a siete naciones europeas e intentan dominar el continente estableciendo alianzas entre sí y rompiéndolas) se puede jugar por correo.

La introducción del ordenador para efectuar todo el trabajo de cálculo y administración ha supuesto el incremento del alcance de este tipo de juegos, junto con el de su ingeniosidad y su sofisticación. Algunos de ellos imaginan inmensas galaxias a través de las cuales los jugadores realizan maniobras con sus flotas espaciales; otros ofrecen territorios míticos y reinos guerreros. Son comunes las alianzas entre diez jugadores o más, lo que implica un gran intercambio de correspondencia o de llamadas telefónicas. Una medida de la calidad de estos juegos viene dada por el hecho de que el espectro de edades de los jugadores es mucho más amplio que en otros tipos de juegos por ordenador.

Guerras galácticas
Starlord ha sido el primer juego postal por ordenador desarrollado en Gran Bretaña. Lo coordina Mike Singleton utilizando un ordenador Commodore Pet 3032, una unidad de disco rígido de 7,5 megabytes y una impresora matricial en color Integrex. El objetivo de cada jugador consiste en descubrir la estrella Throne y convertirse en emperador. A cada movimiento el jugador recibe un mapa que muestra la zona inmediata a la posición de sus fuerzas y una lista de quienes controlan las estrellas más cercanas. Se precisa un disco de gran capacidad debido al alto número de programas que dirigen conjuntamente el juego y a la gran cantidad de información que se debe mantener para más de 700 jugadores. En las tiendas especializadas se vende una revista denominada *Flagship*, con amplia información acerca de los juegos postales

Cortesía de Starlord

Modelos de comportamiento

La simulación es una técnica informática que permite experimentar una situación que, de otra forma, sería peligrosa o muy cara.

Uno de los usos más importantes de los ordenadores se produce en el área de la simulación. Se trata de un método de planificación hacia adelante por el cual "se simula" en el ordenador un modelo de la situación a analizar. Un modelo proporciona una visión simplificada de la situación, reteniendo los aspectos significativos del problema y descartando los detalles irrelevantes.

Tomemos el ejemplo de dos vasos, uno de los cuales contiene vino blanco y el otro vino tinto. Si se le agrega al vaso de vino blanco una cucharada del vino tinto, se mezcla bien y después se vuelve a echar una cucharada de la mezcla en el vaso de vino tinto, ¿cuál de los vasos tendrá mayor impureza?

Existen muchas maneras de resolver el problema, pero una de las más sencillas consiste en utilizar un modelo. Por ejemplo, podríamos establecer uno en el que el volumen de vino de cada vaso fuera exactamente de una cucharada. Es fácil comprender que en este caso ambos vasos acabarán con el mismo grado de impureza (igual mezcla de vino tinto y vino blanco). Ampliar nuestro modelo para mayores cantidades de vino nos demostraría que en todos los casos los resultados serían los mismos.

Los modelos aparecen bajo tres formas principales. Un modelo pictórico (por ejemplo, una fotografía o un mapa) muestra acomodaciones y relaciones espaciales entre los elementos que componen esa imagen. Luego están los modelos cuyos componentes se comportan, los unos en relación a los otros, de manera similar a los elementos reales a los que representan en el problema: por ejemplo, los problemas que se resuelven mediante ordenadores analógicos (véase p. 238). El tercer tipo corresponde a los modelos simbólicos que para representar la situación utilizan símbolos abstractos y relaciones matemáticas. Éstos son los que emplean los ordenadores digitales en las simulaciones.

Existen cuatro situaciones principales entre las que podemos optar para resolver un problema mediante la simulación por ordenador. La primera es aquella situación con la cual experimentar podría resultar demasiado peligroso; por ejemplo, determinar qué nivel de radiactividad es inocuo en la zona aledaña a un reactor nuclear.

La segunda situación, de la cual constituye un buen ejemplo un modelo de la economía nacional, corresponde a aquella en la que sería casi imposible hallar una solución puramente matemática para la serie de ecuaciones que componen el problema. Es mejor establecer las ecuaciones en forma de un modelo para ordenador y observar en ellas los efectos de diferentes acciones y acontecimientos.

El tercer caso representa aquella situación en la que el problema a analizar implica tal inversión que todos los ajustes tienen que hacerse efectivos en la etapa de construcción del modelo, antes de que se asuma el compromiso de construir la versión final. En la planificación de una propuesta para un aeropuerto nuevo para la ciudad de Londres, por ejemplo, para los problemas de ingeniería y planificación se llevaron a cabo exhaustivos trabajos de simulación. Esta simulación investigó el ruido y otras consideraciones relativas al medio ambiente, así como el flujo de personas y de tráfico alrededor del emplazamiento propuesto.

Bajo control
Una utilización importante de la simulación se realiza en el campo de la educación, al entrenar a las personas a partir de modelos de sistemas reales. Un ejemplo lo constituye la simulación de una central nuclear, creada por Atari, en la cual el usuario ha de controlar los diversos sistemas de refrigeración para evitar el recalentamiento del reactor. La documentación explica las distintas funciones de control que poseen las centrales nucleares de este tipo, y también dispone de una modalidad de demostración que ejecutará el programa para usted

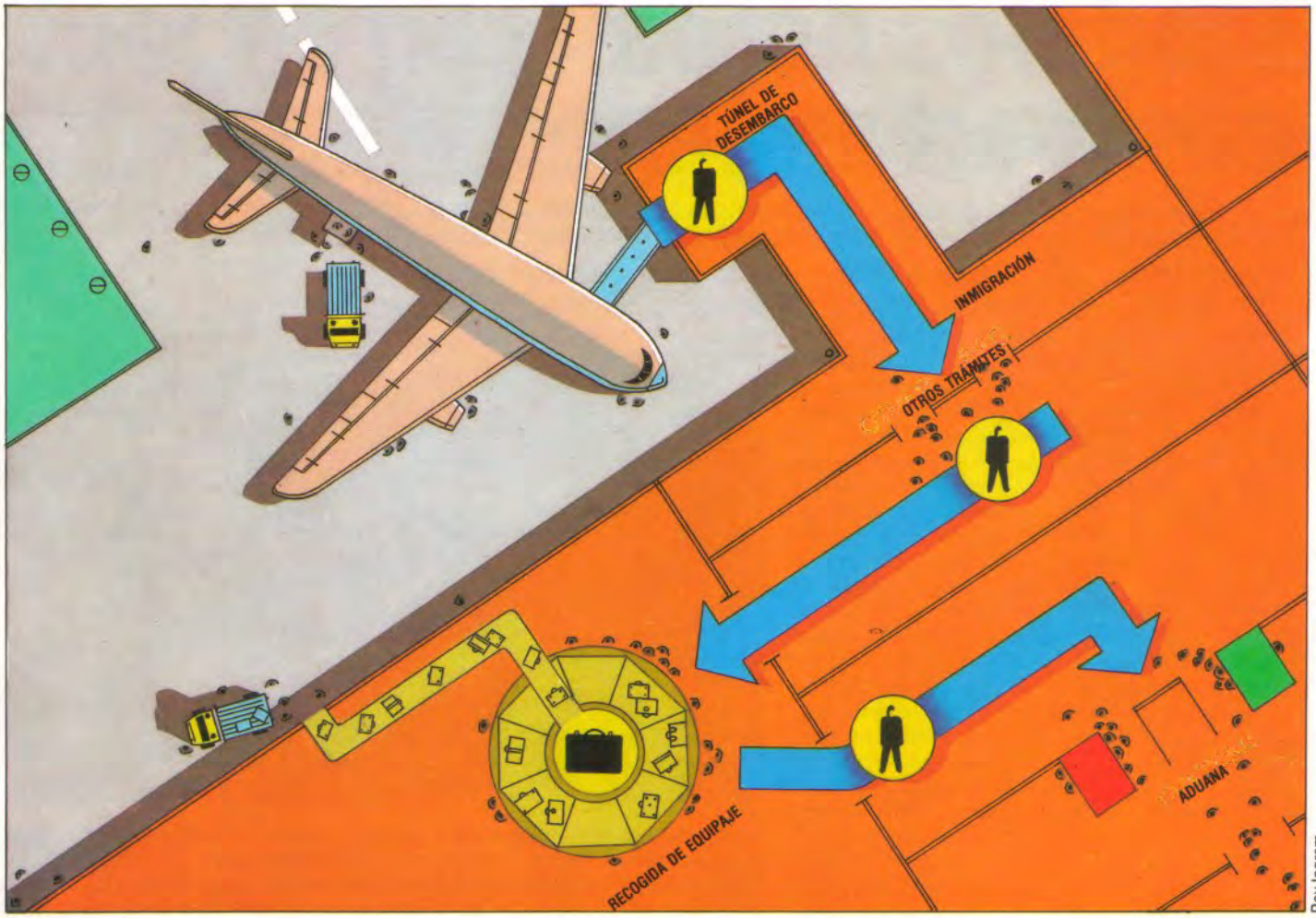
Software, cortesía de Pilot Software
Ian McKinnell

La otra situación en la que resulta de gran valor un modelo es aquella que se traduce en un problema totalmente teórico en el cual es imposible efectuar experimentos físicos. Así, los astrofísicos especulan acerca de cómo nacen las estrellas, y se emplean modelos para evaluar una teoría cosmológica (como la del *big-bang*) en relación a otra.

En toda simulación la primera tarea consiste en construir el modelo. Éste se realiza estudiando la situación y decidiendo cuáles son los elementos importantes y cómo se interrelacionan entre sí.

Los sistemas y sus modelos se dividen en dos familias: los sistemas cerrados o "deterministas" y los sistemas abiertos o "estocásticos". Cuando una familia confecciona el presupuesto para sus gastos, el dinero se puede repartir de muchas maneras, pero al final el balance debe cuadrar, lo que confiere al sistema un carácter determinista. No obstante, si la familia basara cada una de sus decisiones relativas a gastar una suma de dinero en el cara o cruz de una moneda lanzada al aire, y no en función de la cantidad de dinero disponible en su cuenta bancaria, se trataría de un sistema de carácter estocástico.

Con las simulaciones teóricas no se puede estar absolutamente seguro de que el modelo que uno escoja sea el correcto. La gente creía que la Tierra era el centro del universo hasta que Copérnico pro-



Roy Ingram

Diseño de una terminal aérea
La simulación por ordenador se empleó extensamente en el diseño de la nueva terminal 4 del aeropuerto de Heathrow, en Londres. Primero, el programa hubo de simular los patrones de las llegadas de aviones durante el día, con un número variable de pasajeros y equipaje. Luego simuló los "procesos" por los que habrían de pasar los viajeros. El modelo verificó que el sistema pudiera absorber el volumen de tráfico esperado y sugirió las medidas óptimas de planificación

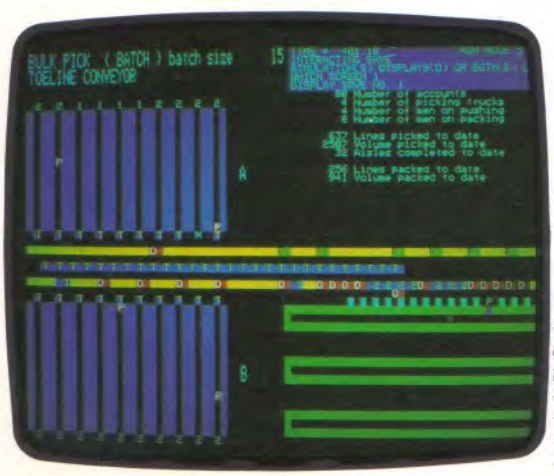
porcionó un modelo matemático mucho más sencillo, con el Sol en el centro. En la actualidad, al observar lejanas galaxias de estrellas, los astrónomos descubren que todas ellas se alejan de nosotros a velocidades que aumentan de continuo, lo que vuelve a sugerir que nosotros nos hallamos en el centro del universo. Pero si adoptáramos un modelo del universo centrado alrededor de la Tierra estaríamos engañados, tal como demuestra un modelo alternativo. Si se salpica un globo con manchas de tinta que representen las estrellas y después se infla, cada mancha se alejará de las otras y, sin embargo, ninguna de ellas está en el centro del globo.

No obstante, emplear modelos ofrece muchas ventajas: ayudan a formular mejores teorías, aceleran el análisis, permiten intentar modificaciones y, lo más importante de todo, son mucho más baratos que las cosas reales. Su utilización permite, asimismo, realizar saltos creativos en la teoría. El láser, por ejemplo, lo inventó alguien mientras continuaba trabajando en un aspecto de un modelo matemático que previamente se había pasado por alto.

BL Systems, subsidiaria de la British Leyland, comercializa en Gran Bretaña un sistema de simulación para aplicaciones múltiples que originalmente se desarrolló para el diseño de líneas de producción y depósitos automatizados en Cowley y Longbridge. El sistema se denomina *See why* (Vea por qué), y para mostrar los resultados utiliza visualizaciones gráficas en lugar de las listas de estadísticas tradicionales.

Un problema típico que puede afrontar un sistema de simulación es el de hacer cola. En un aeropuerto, por ejemplo, si repentinamente cambia el viento y sólo está disponible una de las pistas de aterrizaje, los aeroplanos han de hacer cola. Los aviones sólo llevan a bordo una reserva de combustible limitada, y para que cada avión aterrice se requiere un tiempo específico. Las reglas para hacer cola se programarán en el sistema. En estas simulaciones se emplean generadores de números aleatorios (véase p. 209) para crear acontecimientos inesperados, como llegadas de aviones al azar.

La simulación es un área aplicativa muy importante para los ordenadores digitales e incluso ha dado lugar a la creación de nuevos lenguajes que se han escrito especialmente para proyectos de simulación (por ejemplo, GASP, SIMSCRIPS y GPSS).



Cortesía de BL Systems

"See why" (Vea por qué)
BL Systems, subsidiaria de la British Leyland, desarrolló un paquete de modelación por microordenador para diseñar sus nuevas factorías y cadenas de producción. El paquete, denominado "See why", desde entonces se ha puesto a la venta en Gran Bretaña. Aunque el mayor énfasis del programa recae en los procesos de modelación, incorpora salida gráfica en forma de diagramas esquemáticos. Los números dispuestos al lado de cada una de las etapas del proceso indican cómo avanza el trabajo y los problemas que surgen

Los puntos sobre las íes

El "generador de caracteres" es la sección de la memoria que define la forma en que aparecen los caracteres en la pantalla. En algunos sistemas el usuario puede diseñar sus propios símbolos

Anteriormente hemos visto en nuestro curso de programación BASIC (véase p. 214) que todos los caracteres alfanuméricos (y los símbolos gráficos, en el caso de que su ordenador los posea) se almacenan en la memoria RAM en forma de códigos de ocho bits (por lo general ASCII), de modo que un carácter ocupa un byte.

Cuando se imprime información en la pantalla, los códigos para cada carácter se colocan en una zona reservada de la memoria denominada *RAM de video*. Si, por ejemplo, se imprimiera la letra A en el rincón superior izquierdo de la pantalla, el primer byte de RAM de video contendría el código 65 (ASCII correspondiente a A). Si debajo de la A se imprimiera una C, y el ordenador tuviera una pantalla de 40 columnas, entonces en la 41 localización de RAM de video se hallaría el valor 67, y así sucesivamente. ¿Cómo convierte el ordenador el valor 65 en el patrón de puntos que componen en la pantalla el carácter A? La respuesta viene dada por un dispositivo denominado *generador de caracteres*.

Un generador de caracteres es sencillamente un conjunto de patrones almacenados en la memoria como bits. En los ordenadores personales el generador de caracteres está almacenado en ROM, lo que permite una visualización inmediata de los caracteres cuando se enciende la máquina. El generador de caracteres puede estar incorporado en las unidades ROM que contienen el intérprete de BASIC y el sistema operativo, o puede estar en un chip de ROM propio. En este último caso, con frecuencia el usuario encontrará proveedores independientes que le ofrezcan unidades ROM de recambio capaces de crear un juego de caracteres extranjeros, o una gama de símbolos especializados para, por ejemplo, ingeniería o matemáticas. Sin embargo, cada vez es mayor el número de máquinas que permiten transferir el generador de caracteres a RAM, gracias a lo cual el programador puede diseñar sus propios caracteres y símbolos.

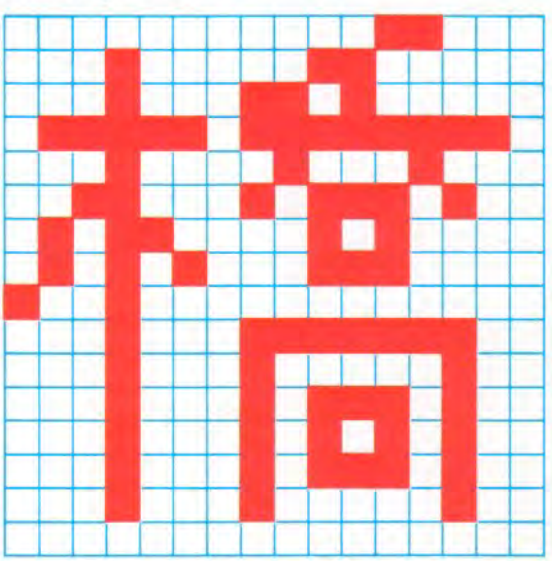
Todos los caracteres se construyen en una matriz de puntos, que en la mayoría de los ordenadores personales es de ocho por ocho, si bien con una matriz mayor se obtendría una mayor legibilidad y una gama más amplia de caracteres visualizables. Éstos se diseñan rellenando los cuadros de la cuadrícula. Se definen representando con un 1 los cuadrados ocupados y con un 0 los cuadrados en blanco, dando un total de 64 bits para cada carácter.

El primer byte del generador de caracteres representará el patrón de bits para la línea superior del primer carácter de la tabla. Si el ordenador sólo puede visualizar los caracteres ASCII con códigos de 0 a 127, el generador de caracteres requerirá entonces 128 × 8 bytes (1 Kbyte de memoria).

La dificultad para el ordenador reside en que cuando el barrido de la pantalla de televisión está generando la línea superior de la visualización, debe producir la línea superior de puntos para el carácter situado en el extremo superior izquierdo de la pantalla, seguido de la línea superior del carácter situado a su derecha, y así sucesivamente a través de la pantalla. Después, cuando el barrido empieza a pasar por segunda vez, debe hallar y visualizar la segunda línea para cada uno de los caracteres de la fila superior de la pantalla.

El sistema de circuitos de video consigue esto gracias a que posee dos contadores independientes. Uno lleva el registro de a qué localización de memoria de video corresponde el punto por el cual está pasando el barrido. El otro cuenta las líneas del barrido, empezando desde cero para la primera línea y llegando a siete para la octava, y volviendo luego a empezar desde cero en la novena, y así sucesivamente. De modo, entonces, que el ordenador busca el código ASCII o de visualización en la memoria de video, lo multiplica por ocho y suma el valor que en ese momento registra el contador de líneas. Esto le proporciona una dirección en el generador de caracteres para el patrón de ocho bits correspondiente a la fila correcta del carácter que se está barriendo en ese momento.

A modo de ejemplo, veamos cómo se generaría el carácter A, cuyo código ASCII es 65. Podemos calcular que la primera línea del carácter se almacenará en el byte número 520 (64 × 8 + 0); la segunda, en el byte 521 (65 × 8 + 1); la tercera línea, en el byte 522 (65 × 8 + 2); y así sucesivamente. Todo lo que falta es que el sistema de circuitos de video convierta esos ocho bits en una secuencia de voltajes que haga que el haz de electrones de barrido se encienda y se apague para visualizar el carácter en la pantalla.



Un carácter japonés
Los caracteres japoneses pueden ser sumamente complejos y la matriz normal de 8 × 8 no logra visualizarlos con suficiente detalle como para que resulten legibles. El carácter para "puente", que muestra la ilustración, apenas resulta legible en una matriz de 16 × 16. Utilizando una matriz de puntos de 24 × 24 se obtiene una mayor fidelidad

Tandy Color

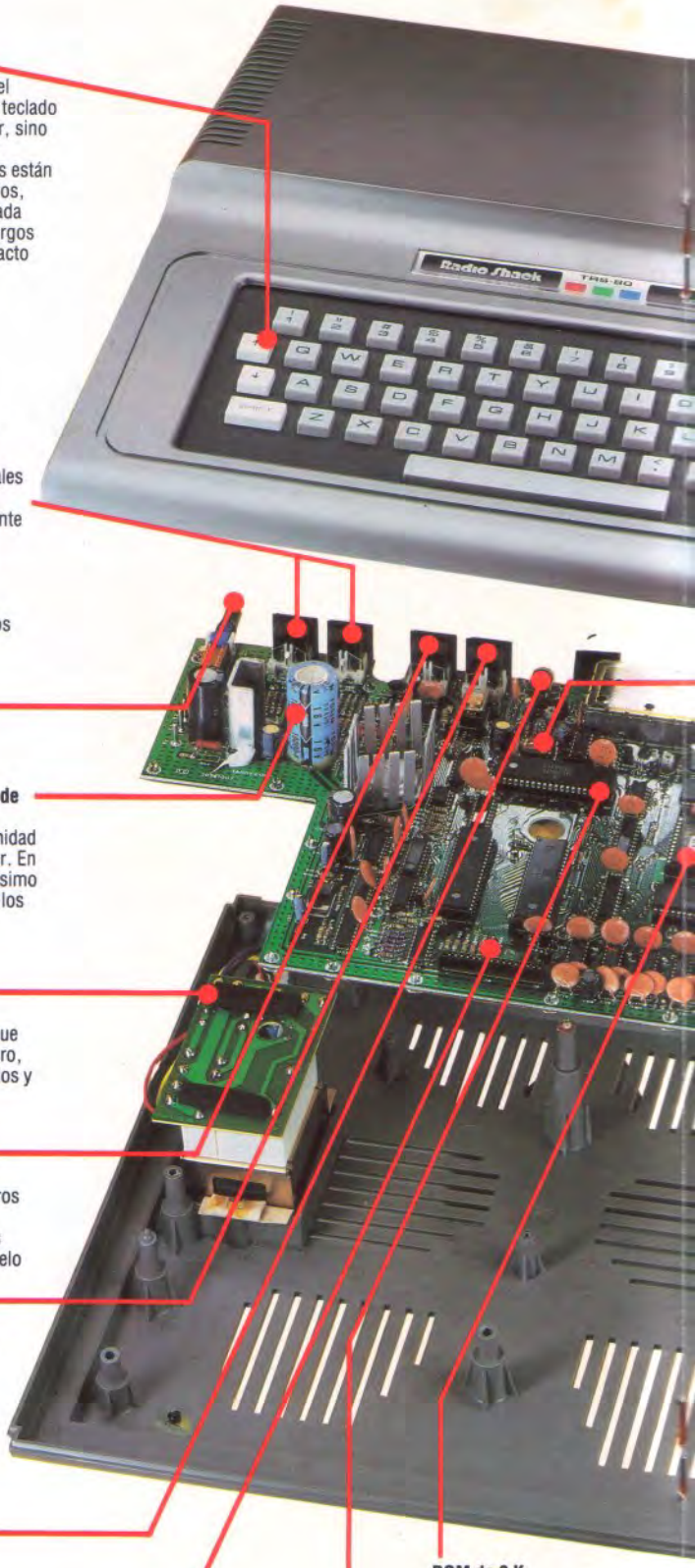
De diseño muy similar al Dragon 32, este ordenador tiene un buen apoyo de periféricos: desde digitalizadores a impresoras de chorro de tinta

A pesar de ser bastante distinto en cuanto a su aspecto, el ordenador Tandy Color guarda notable parecido con el Dragon 32 (véase p. 130) por la forma en que funciona. De hecho, son tan compatibles que muchos de los programas que se ejecutan en uno pueden ejecutarse en el otro.

Existen, asimismo, otras semejanzas. Ambos poseen el mismo tipo de conexión para cartucho y la misma CPU: una 6809E. Ésta es una unidad de procesamiento muy potente, poco utilizada por otros ordenadores personales. Pero en algunos aspectos esta CPU está descompensada respecto a los otros componentes del ordenador. Normalmente, la 6809 se emplea en máquinas diseñadas para uso profesional. Sin embargo, en el Tandy Color, que posee una velocidad de reloj relativamente lenta (895 KHz), su potencial se desperdicia bastante.

El ordenador viene con su propia versión de BASIC: el BASIC de Tandy Color. A pesar de la poca velocidad del reloj, el funcionamiento del ordenador es aceptablemente rápido gracias a la sofisticación de la CPU. Posee varias palabras órdenes especializadas, diseñadas para que el hardware resulte más sencillo de utilizar. Una de las configuraciones del Tandy Color es el controlador de video 6847, un dispositivo programable que produce un formato de pantalla de 16 líneas de 32 caracteres. Normalmente la visualización es de letras negras

Tablilla Tandy para gráficos
La tablilla para gráficos GT-116 se puede utilizar ya sea como digitalizador, para trazar dibujos e imágenes en el ordenador, o bien como dispositivo señalador, para seleccionar ítems de los menús visualizados en la pantalla del ordenador



El teclado Tandy
A diferencia del Dragon, el Tandy Color no posee un teclado estilo máquina de escribir, sino que, en cambio, tiene 53 botones cuadrados. Éstos están suficientemente espaciados, pero no ofrecen la adecuada "sensación táctil" para largos períodos de escritura al tacto

Conectores para palanca de mando
Son totalmente proporcionales e incluyen una línea de señalización, que generalmente está conectada a un botón. Estas interfaces se pueden utilizar para otras señales analógicas, como las provenientes de experimentos de laboratorio

Interruptor

Componentes de la fuente de alimentación
Acondicionan y dan uniformidad a la salida del transformador. En el proceso se genera muchísimo calor, que es eliminado por los grandes disipadores

Transformador
La fuente de alimentación eléctrica incorporada consigue que el montaje sea más pulcro, con menos cables secundarios y cajas separadas

Conexión RS232
Aquí se pueden conectar impresoras, modems u otros dispositivos en serie. La máquina no tiene previstas entradas o salidas en paralelo

Interface para cinta
Esta interface proporciona también control remoto del motor

Selección de canal
La salida de TV se puede sintonizar en dos canales diferentes. La selección se efectúa cambiando la posición de este interruptor

ROM de 8 K
Contiene el BASIC de Tandy Color y rutinas de bajo nivel

Conector para el teclado
A través de esta conexión, un cable plano de plástico laminado une el teclado con el ordenador

Controlador de video 6847
El estilo y el color de la visualización en pantalla se controlan programando este chip, ya sea en BASIC o en código de lenguaje máquina

sobre un fondo verde, si bien tanto el color del fondo como el del primer plano se pueden escoger de entre nueve colores y se puede alterar el formato. Se proporcionan facilidades para generación de sonido, pero al disponer de sólo un canal y carecer de control de envoltura y generación de "ruido blanco", esta máquina es menos flexible que otras.

El Tandy Color posee el mismo tipo de carga de cartuchos ROM que el Dragon, y la gama de software disponible en este formato es muy amplia. Existe la variedad normal de cartuchos de juegos: ajedrez, cartas, damas, prospecciones petrolíferas y pinball. El software disponible incluye cartuchos de música, aritmética y mecanografía. También existe una buena gama de programas de utilidad para presupuestos, archivos y redacción de cartas, así como algunos medios auxiliares para desarrollo de programas para BASIC o código de lenguaje máquina. Además, existe una ROM de diagnóstico muy fácil de manejar, diseñada para asegurar el correcto funcionamiento de la máquina.

En la parte posterior, una interface en serie RS232 permite la utilización de una gama de dispositivos externos, incluyendo impresoras en serie. Esta conexión se puede usar también para comunicarse con otros ordenadores mediante un modem (véase p. 216). Con un dispositivo de este tipo y con el cartucho de videotex se pueden buscar diversas bases de datos o enviar correo electrónico.

También en la parte posterior de la máquina se proporcionan dos conectores para palanca de mando, cada uno de los cuales posee dos ejes de movimiento y un botón de disparo. Esto parece plantearle un pequeño problema al BASIC de Tandy Color: cuando se pulsa el botón se genera un flujo de caracteres. Las palancas de mando son mucho mejores que otras, con movimiento variable en lugar del tipo de mecanismo más usual y sencillo de empujar y tirar, y se pueden emplear para leer otras entradas analógicas (de valor variable), como las de los experimentos físicos.

La ampliación de la máquina se efectúa a través de la conexión para cartucho y se pueden conectar hasta cuatro unidades de disco de 5 ¼ pulgadas, que proporcionan un estimable almacenamiento total de 626 Kbytes.

TANDY COLOR

DIMENSIONES
369 x 344 x 94 mm

VELOCIDAD DEL RELOJ
0,895 MHz

MEMORIA
4 Kbytes de RAM; 8 Kbytes de ROM, ampliables a 32 Kbytes

VISUALIZACION EN VIDEO
16 líneas de 32 caracteres. Nueve colores con selección independiente de primer plano y fondo. 127 caracteres predefinidos y 127 caracteres definibles por el usuario

INTERFACES
Conexión RS232, cassette, dos palancas de mando proporcionales, interface para TV en dos canales

LENGUAJE SUMINISTRADO
BASIC

OTROS LENGUAJES DISPONIBLES
Código de lenguaje máquina 6809 con paquetes Assembler

VIENE CON
Manuales de instalación y de BASIC, cable para TV

TECLADO
53 teclas individuales, dispuestas como en una máquina de escribir

DOCUMENTACION
La presentación del manual del usuario es desigual, pero éste es muy amplio y constituye una guía perfectamente adecuada para la máquina

Chris Stevens

Reloj de video
El tamaño y el número de caracteres en la pantalla se establecen por la frecuencia generada por este cristal, que también se utiliza para la producción real de la imagen

Interruptor de borrado
Pulsándolo limpiará la máquina y ésta empezará de nuevo

Carga de cartuchos
Se utiliza para conectar unidades de disco y software operativo, además de cartuchos corrientes de programas y juegos

RAM
Esta máquina posee 16 K de RAM dinámica en 8 chips, cada uno de los cuales abarca un bit del byte de 8 bits

CPU 6809E
Un chip avanzado con aritmética de 16 bits interna, que opera sobre un bus de datos de 8 bits y puede direccionar 64 K de memoria

Enchufe para ROM de 8 K de recambio
Se puede emplear para servicios adicionales o ampliar el BASIC

Las impresoras Tandy
El Tandy Color está bien apoyado por una gama de impresoras que comercializa la propia casa Tandy. La CGP-220, por ejemplo, utiliza el principio del chorro de tinta para producir siete colores, y su funcionamiento es silencioso. La CGP-115, más barata que la anterior, utiliza cuatro lápices esferográficos en miniatura para producir una impresión en color y trazar gráficos

Chris Stevens

Nuevas entradas

Para poder insertar una nueva entrada en una matriz, en primer lugar es necesario hallar un espacio vacío. La búsqueda binaria es una forma muy eficaz de lograrlo

En el capítulo anterior vimos que un archivo de datos se compone de registros, que a su vez se dividen en campos, a cada uno de los cuales se le puede dar acceso a los otros campos mediante un campo de clasificación. Ahora examinaremos algunas de las técnicas para buscar en estas listas.

Crear registros para nuestra agenda de direcciones no es difícil. Supongamos que existe una matriz en serie separada para cada uno de los campos de registros. Estas se podrían denominar NOMBRE$ (para nombre completo) CALLE$, CIUDAD$ y TELEFONO$ (luego hablaremos de por qué hemos utilizado aquí una variable alfanumérica para el campo del número de teléfono en vez de una variable numérica). En la lista de las ocho funciones deseables del programa de la agenda de direcciones, el número seis era la capacidad de agregar nuevas entradas. Si estas ocho opciones se presentaran en la pantalla al principio, cuando se ejecutara el programa, al seleccionar 6 lo llevaría a una rutina de entrada del tipo que presentamos como ejercicio.

Supongamos que en la agenda ya hay cierto número de entradas, pero que no recuerda cuántas. Es esencial que las nuevas entradas no se escriban sobre las entradas existentes; de este modo una de las tareas del programa podría consistir en buscar a través de los elementos de una de las matrices para hallar la primera que no contenga datos.

Buscar en una matriz para ver si un elemento está "ocupado" no es difícil. En BASIC las variables alfanuméricas se pueden comparar del mismo modo que se comparan las variables numéricas. IF A$ = "COMPUTER" THEN... es tan válido como IF A = 61 THEN..., al menos en la mayoría de las versiones de BASIC. Si cualquiera de las matrices de nuestra agenda de direcciones ya posee una entrada, ésta constará de al menos un carácter alfanumérico. Un elemento "vacío" no contendrá ningún carácter alfanumérico, de manera que todo lo que tenemos que hacer es buscar a través de los elementos, empezando desde el principio, hasta que hallemos uno que no contenga caracteres.

Si hay matrices para el nombre, la calle, la ciudad y el número de teléfono, tendremos cuatro matrices con un elemento en cada una para cada campo del registro. Puesto que todos estos campos "van juntos", el registro 15 tendrá sus datos de nombre en el elemento 15 de la matriz de nombre, sus datos de calle en el elemento 15 de la matriz de calle, los datos de ciudad en el elemento 15 de la matriz de ciudad, y los datos del número de teléfono en el elemento 15 de la matriz de número de teléfono. Por lo tanto, sólo necesitamos buscar en una de estas matrices para hallar un elemento vacío; no necesitamos revisar todas las matrices.

Si la variable POSICION representa el número del primer elemento libre de cualquiera de las matrices, un programa para localizar POSICION (en el supuesto de que no lo sepamos ya) podría ser tan sencillo como el siguiente:

```
PROCEDIMIENTO (hallar elemento libre)
BEGIN
  LOOP
    REPEAT UNTIL localizar elemento libre
    READ matriz (POSICION)
    POSICION = POSICION + 1
    IF matriz (POSICION) = " "
      THEN tomar nota POSICION
      ELSE no hacer nada
    ENDIF
  ENDLOOP
END
```

En BASIC, esto tendría un desarrollo sencillo:

```
1000 FOR L= 0 TO 1 STEP 0
1010 LET POSICION = POSICION + 1
1020 IF NOMBRE$ (POSICION) = " " THEN
     LET L = X
1030 NEXT L
1040 REM resto del programa
```

Observe que en la línea 1020 el valor de X es el que se requiere para terminar el bucle FOR...NEXT y este valor varía de máquina a máquina (véase el recuadro "Complementos al BASIC"). También es importante observar que éste es un fragmento de programa y se supone que NOMBRE$ ( ) está DIMensionada y que se ha inicializado POSICION. Para ejecutar este fragmento como si se tratara de un programa, debe DIMensionar NOMBRE$ ( ) e inicializar POSICION y X en algún momento antes de la línea 1000.

Aunque ya hemos utilizado anteriormente la técnica FOR X = 0 TO 1 STEP 0, éste es un buen momento para analizar con más detalle la forma en que funciona. Por lo general, un bucle FOR...NEXT en BASIC "sabe" de antemano cuántas veces se espera que se repita el fragmento de programa. Si desea repetir algo 30 veces, FOR X = 1 TO 30 lo conseguirá de forma precisa. No obstante, esta vez estamos simulando un bucle REPEAT...UNTIL (repetir hasta). Aunque las versiones corrientes de BASIC no disponen de REPEAT...UNTIL, simularlo mediante un FOR...NEXT es bastante fácil. En la medida en que fracase la condición de la línea 1020, L (el contador del bucle FOR...NEXT) permanece en el valor 0, sumándosele 0 a cada iteración (repetición del bucle), mientras que la línea 1010 hace que POSICION se incremente en 1 a cada iteración. Cuando la condición de la línea 1020 es verdadera (o sea, cuando se ha hallado un elemento de NOMBRE$( ) vacío), L se establece en el valor X, y el bucle FOR...NEXT acaba en la línea 1030. Con ello POSICION señala el primer elemento libre de NOMBRE$( ).

POSICION es un valor que posiblemente necesitemos establecer antes, cada vez que se emplea el programa de la agencia de direcciones, y es un valor que es probable sea necesario actualizar varias veces durante la utilización del programa. Por lo tanto será una de nuestras variables "globales" y será necesario que el establecimiento de su valor forme parte de una rutina de "inicialización". Esto se puede hacer cada vez que se ejecuta el programa, o se puede crear una "bandera" si el valor de POSICION ha cambiado o no desde la última vez que se ejecutara el programa. Este último enfoque no es difícil, pero en este punto viene a crear una complicación innecesaria. Nosotros dejaremos que las cosas sigan siendo sencillas y hallaremos el valor de POSICION como una de las primeras tareas cada vez que se ejecuta el programa.

Pasemos revista a las actividades que deseamos que haga la agenda de direcciones computerizada y veamos si podemos avanzar hacia una estrategia de programa total. Esta vez seremos un poco más rigurosos y daremos por sentado que cada una de las actividades se tratará como subrutina separada (el nombre de cada una se indicará entre asteriscos).

1.	Hallar registro (del nombre)	*ENCREG*
2.	Hallar nombres (de nombres incompletos)	*ENCNOMBRE*
3.	Hallar registro (de ciudad)	*ENCCIUDAD*
4.	Hallar registros (de inicial)	*ENCINICI*
5.	Listar registros (todos)	*LISTREGS*
6.	Agregar registro	*INCLREG*
7.	Modificar registro	*MODREG*
8.	Borrar registro	*BORREG*
9.	Salida del programa (guardarlo)	*SALPROG*

Ahora sabemos, en líneas generales, cuáles son las "entradas" y "salidas" deseadas del programa, de modo que ya podemos empezar a pensar en términos de un programa principal. Toda la pormenorización se puede efectuar a través del proceso de programación *top-down* (de arriba abajo), y se puede codificar en las diversas subrutinas. Sabemos que se deberán inicializar varias cosas, incluyendo el valor de POSICION. Sabemos que, como será un programa activado por menú, se nos presentará una serie de opciones cada vez que se ejecute el programa. Sabemos, asimismo, que independientemente de cuál sea nuestra respuesta a las opciones presentadas, desearemos que se ejecute al menos una de ellas. De manera que ya puede tomar forma el cuerpo del programa principal:

```
PROGRAMA PRINCIPAL
EMPEZAR
   INICIALIZACION (procedimiento)
   PRESENTACION (procedimiento)
   ELECCION (procedimiento)
   EJECUCION (procedimiento)
FIN
```

En BASIC, esto tendría el aspecto siguiente (con los números de línea reemplazados por los nombres de las subrutinas):

```
10 REM PROGRAMA AGENDA DE MI COMPUTER
20 GOSUB *INICIALIZACION*
30 GOSUB *PRESENTACION*
40 GOSUB *ELECCION*
50 GOSUB *EJECUCION*
60 END
```

La subrutina o procedimiento *PRESENTACION* visualizaría en la pantalla durante algunos segundos un saludo, seguido del menú. El saludo podría ser:

```
*BIEN VENIDO A LA*
*AGENDA COMPUTERIZADA*
*DE MI COMPUTER*
(PULSE LA BARRA ESPACIADORA CUANDO ESTE
LISTO PARA CONTINUAR)
```

En respuesta a la invitación a pulsar la barra espaciadora, el programa se bifurcará hacia la subrutina *ELECCION* y se le ofrecerá al usuario una pantalla como ésta:

```
*DESEA USTED*
1. HALLAR UN REGISTRO (de un nombre)
2. HALLAR NOMBRES (de parte de un nombre)
3. HALLAR REGISTROS (de una ciudad)
4. HALLAR REGISTROS (de una inicial)
5. LISTAR TODOS LOS REGISTROS
6. AGREGAR UN REGISTRO
7. MODIFICAR UN REGISTRO
8. BORRAR UN REGISTRO
9. SALIR Y GUARDAR
*ELIJA DE 1 A 9*
*SEGUIDO DE RETORNO*
```

En este punto, el programa se bifurcará hacia la subrutina apropiada, según el número al que se dé entrada. Ahora la estructura del programa está empezando a tomar forma. Es necesario que todas las opciones, a excepción de la número 9 (SALIR y GUARDAR), terminen con una instrucción para retornar a la subrutina *ELECCION*. Pero no hemos tenido en cuenta muchos detalles relativos a la organización interna de los datos. De ellos nos ocuparemos más adelante.

Supongamos que estamos ejecutando el programa, que éste ya posee todos los registros que necesitamos y que deseamos buscar un registro completo dando entrada solamente a un nombre. Para ello se requiere la opción 1: HALLAR UN REGISTRO (*ENCREG*). Antes de que intentemos diseñar esta parte del programa, consideremos algunos de los problemas que plantean las rutinas de búsqueda computerizada.

La búsqueda

Los libros de texto acerca de las técnicas de programación tienden a tratar conjuntamente la búsqueda y la clasificación. Los lectores recordarán que nosotros ya hemos abordado el tema de la clasificación en un programa diseñado para ordenar nombres alfabéticamente (véase p. 134). Tanto la clasificación como la búsqueda plantean puntos interesantes acerca de cómo se organizan los datos en un ordenador o en cualquier otro sistema de información.

Si una agenda de direcciones "manual" consistiera en una agenda de notas sin índice alfabético, y si las entradas se fuesen agregando a medida que fuera necesario, sin clasificarlas por orden alfabético, tendríamos una estructura de datos de las que se denominan "pilas". Una pila es un conjunto de datos agrupados por el orden en que llegan. Es obvio que una pila es la forma menos eficaz de organizar los datos. Cada vez que se desee hallar la dirección y el número de teléfono de alguien habrá que mirar a través de toda la agenda de direcciones. Lo mismo suele ocurrir con los sistemas informáticos, aunque existen ocasiones en las que los crite-

rios en virtud de los cuales se accede a los datos son tan impredecibles que una pila puede ser una estructura de datos tan buena como cualquier otra.

Una estructura de datos más organizada, y cuya utilización resulta mucho más sencilla tanto para las personas como para los ordenadores, es la que se consigue cuando la información se organiza de acuerdo con un sistema simple y reconocido. Una guía telefónica constituye un buen ejemplo de un conjunto de datos (nombres, direcciones y números de teléfono) donde el campo del nombre está ordenado de acuerdo con las sencillas reglas de la sucesión alfabética. En sí mismos, los números en el fondo están dispuestos al azar, pero los nombres (que son más "significativos") están ordenados según unas reglas muy fáciles de seguir.

De acuerdo con la organización interna de los datos de nuestra agenda de direcciones computerizada, los datos están dispuestos en una pila, almacenándose un registro en el elemento X de la matriz de "nombre", en el elemento X de la matriz de "calle", etc., y almacenándose el registro siguiente en el elemento X +1 de la matriz de "nombre", en el elemento X +1 de la matriz de "calle", y así sucesivamente. Hallar un dato determinado (JUAN FLORES, por ejemplo) implicaría, por lo tanto, mirar el primer elemento de la matriz de "nombre" y ver si corresponde a JUAN FLORES, mirar el segundo elemento y ver si se refiere a JUAN FLORES, y así sucesivamente hasta que se logre localizar el campo o bien, por el contrario, descubrir que no hay ninguna entrada para JUAN FLORES.

Si el dato que deseamos buscar ya se hubiese ordenado en una estructura reconocible, veríamos cuánto se simplificaría la búsqueda. Supongamos que posee una base de datos sobre equipos de fútbol y que uno de los campos de los registros es el de los resultados de una semana determinada. Una base de datos eficaz le permitiría hallar qué equipo o equipos marcaron 11 goles durante esa semana. Ésta sería la matriz que retendría los marcadores de los equipos para la semana en cuestión:

```
1,6,2,2,1,9,0,0,2,1,4,11,4,2,12,5,2,1,0,1
```

Debería resultar obvio que los marcadores están dispuestos por orden de los equipos y no por el orden de los marcadores. Son veinte los equipos que participaron, y sólo uno consigue marcar 11 goles esa semana. Éste fue el 12.° equipo al que se dio entrada en la matriz. Con unos datos no estructurados como éstos, la única forma de hallar la información que se desea consiste en mirar el primer elemento y ver si es 11; de no serlo, mirar el elemento siguiente para comprobar si es 11, y así sucesivamente hasta localizar un 11 o descubrir que no hay ningún elemento cuyo valor sea igual a 11.

Si analizáramos estos datos, veríamos que había un total de 20 marcadores, cuyos valores oscilaban entre 0 y 12. Este ejemplo es relativamente trivial, e incluso aunque tuviéramos que buscar a través de cada dato no nos llevaría mucho tiempo descubrir que el 11 estaba en el 12.° elemento de la matriz. Pero, ¿y si en una gran matriz hubiera miles de elementos? Buscar entre una cantidad de datos muy numerosa retrasaría de manera considerable un programa.

La solución consiste en ordenar primero los datos, de modo que se pueda efectuar la búsqueda con mucha más rapidez. A continuación ofrecemos nuevamente la matriz de marcadores, dispuestos en orden numérico:

```
0,0,0,1,1,1,1,1,2,2,2,2,2,4,4,5,6,9,11,12
```

Si sabemos que el número de equipos es 20, entonces la forma más rápida de hallar la posición del marcador que deseamos consiste en dividir la matriz en dos partes y buscar sólo en la que probablemente contendrá el número que deseamos. Recuerde que es probable que examinar grandes cantidades de datos lleve mucho más tiempo que operaciones aritméticas tan sencillas como dividir un número por dos. El algoritmo para localizar el marcador tendría ahora la siguiente configuración:

```
Hallar la matriz que contenga los marcadores
Leer el número que deseamos buscar
Hallar la longitud de la matriz
Hallar el punto medio de la matriz
Hacer un bucle hasta localizar el número
  Si el dato situado en el punto medio es igual al
  número que estamos buscando, entonces se ha
  localizado el número
  Si no lo es, ver si el número buscado es mayor
  o menor que el situado en el punto medio
  Si el número buscado es mayor que el número
  situado en el punto medio, hallar el punto
  medio de la parte superior de la matriz
  Si el número requerido es menor que el número situado en el punto medio, hallar el punto
  medio de la parte inferior de la matriz
  (Repetir este procedimiento hasta localizar el
  número)
```

A esto le podríamos dar la siguiente forma:

```
EMPEZAR
  Hallar la matriz de marcadores
  INPUT NUMERO (a buscar)
  LOOP hasta localizar el número
    IF NUMERO = (punto medio)
      THEN apuntar posición punto medio
      ELSE
        IF NUMERO > (punto medio)
          THEN hallar punto medio de mitad
            superior
          ELSE hallar punto medio de mitad
            inferior
        ENDIF
      ENDIF
  ENDLOOP
  IF NUMERO está localizado
    THEN PRINT posición de punto medio
    ELSE PRINT "NUMERO NO HALLADO"
  ENDIF
END
```

Si piensa en este programa escrito en seudolenguaje, verá que finalmente no puede fracasar en localizar el número que se está buscando si existe en la matriz. Desarrollemos este seudolenguaje hasta llegar a un programa de trabajo. Este proceso de búsqueda mediante subdivisión repetida se denomina *búsqueda binaria*.

Le ofrecemos, para que pruebe con él, un programa en BASIC fundamentado en el seudolenguaje anterior. Crea una matriz y lee los marcadores desde una sentencia de datos. Luego se prepara para buscar el marcador. Si lo encuentra, imprime el elemento de la matriz en el que se encontró el número.

```
10 REM UN PROGRAMA PARA LOCALIZAR UN
   NUMERO EN UNA MATRIZ
20 DIM MARCADORES (20)
30 FOR Z = 1 TO 20
40 READ MARCADORES (Z)
50 NEXT Z
60 DATA 0,0,0,1,1,1,1,1,2,2,2,2,2,4,4,5,6,9,11,12
70 LET L = 20
80 LET BTM = 1
90 LET TP = L
100 INPUT "DE ENTRADA AL MARCADOR";N
110 FOR Z = 0 TO 1 STEP 0
120 LET L = TP - BTM
130 LET MD = BTM + INT(L/2)
140 IF N = MARCADORES(MD) THEN LET Z = X
150 IF N > MARCADORES(MD) THEN LET
    BTM = MD
160 IF N < MARCADORES(MD) THEN LET
    TP = MD
170 NEXT Z
180 PRINT "EL MARCADOR ESTABA EN
    EL ELEMENTO N.º ";MD
190 END
```

Nuevamente, observe que se habrá de inicializar X según las exigencias de su máquina (véase el recuadro "Complementos al BASIC").

Si los datos retenidos en un archivo o en una matriz son bastante regulares, como en el caso de una guía telefónica, donde los nombres están distribuidos con razonable uniformidad a través del alfabeto, entonces la búsqueda binaria es un procedimiento eficaz para hallar una entrada determinada. No obstante, no es de ningún modo la forma más eficaz, y hay algoritmos alternativos que pueden encontrar los datos utilizando menos iteraciones. Uno de ellos es la técnica *hashing*, en que el programa hace una conjetura sobre la localización de la entrada, y la va refinando hasta hallarla. Pero esto rebasa los límites de este curso, y el proceso de búsqueda binaria basta para nuestras necesidades.

Ejercicios

Si ejecuta este programa, verá que funciona siempre y cuando dé entrada a un marcador que exista en la matriz. Si diera entrada a un marcador como 3, que no está en la matriz, el programa no conseguiría llegar hasta el final y no aparecería ningún mensaje de error. Si digitara 12, que sí se halla en la matriz, el programa fracasaría en localizarlo. El programa también da por sentado que todos los números de la matriz clasificada serán diferentes, pero, como puede observar a partir de la sentencia de datos, varios números se dan más de una vez. El programa no detecta esto ni informa de todas las localizaciones donde se produce el número.

Su tarea consiste en:

1. Analizar el programa y descubrir por qué éste no puede localizar un marcador de 12
2. Modificar una línea del programa para rectificar este defecto
3. Descubrir por qué el programa no puede manipular números que no existan en la variable e idear una estrategia para superar el problema

En la página 235 le ofrecíamos una serie de ejercicios de revisión para ayudarlo a valorar su nivel de aprovechamiento en nuestro curso de programación BASIC. Hallará las soluciones en la página 280.

Complementos al BASIC

SPECTRUM

El siguiente es el listado del Spectrum para el primer fragmento de programa en BASIC:

```
100 DIM N$(6,4)
110 LET POSICION = 0
120 LET N$(1) = "JUAN"
130 LET N$(2) = "INES"
140 LET N$(4) = "JOSE"
1000 FOR L = 0 TO 1 STEP 0
1010 LET POSICION = POSICION + 1
1020 IF N$(POSICION) = " "
     THEN LET L = 2
1030 NEXT L
1040 PRINT "EL N.º DEL 1.er ELEMENTO
     LIBRE ES "; POSICION
1050 STOP
```

Observe que de la línea 100 a la 140, así como la línea 1040, transforman el fragmento de programa (líneas 1000 a 1030) en un programa de demostración de trabajo. Se pueden modificar los valores y el formato de estas líneas extras para investigar el funcionamiento del fragmento de programa.

El segundo programa para el Spectrum es:

```
10 REM UN PROGRAMA PARA
   LOCALIZAR UN NUMERO EN UNA
   MATRIZ
20 DIM M(20)
30 FOR Z = 1 TO 20
40 READ M(Z)
50 NEXT Z
60 DATA 0,0,0,1,1,1,1,1,2,2,2,2,2,
   4,4,5,6,9,11,12
70 LET L = 20
80 LET BTM = 1
90 LET TP = L
100 INPUT "DE ENTRADA AL
    MARCADOR";N
110 FOR Z = 0 TO 1 STEP 0
120 LET L = TP-BTM
130 LET MD = BTM + INT(L/2)
140 IF N = M(MD) THEN LET Z = 2
150 IF N > M(MD) THEN LET
    BTM = MD
160 IF N < M(MD) THEN LET TP = MD
170 NEXT Z
180 PRINT "EL MARCADOR ESTABA EN
    EL ELEMENTO N.º ";MD
190 STOP
```

VARIABLES

Para lo referente a las variaciones de los nombres de las variables, véase "Complementos al BASIC" anterior (p. 257).

STEP 0

En ambos programas en el texto principal existe una referencia a "...THEN LET Z = X". Los valores para la variable X son:

Oric-1	reemplazar X por 1
Dragon 32	reemplazar X por 1
Lynx	reemplazar X por 2
BBC Micro	reemplazar X por 2
Commodore 64 y Vic-20	reemplazar X por 1

LYNX

Ambos programas del texto principal funcionarán bien en el BBC, el Dragon 32, el Lynx, el Oric-1, el Commodore 64 y el Vic-20, siempre que se cumplan las indicaciones incluidas en el "Complementos al BASIC" relativo a los nombres de las variables y a STEP 0.

Los listados del Spectrum se apartan de la norma en la sentencia DIM de la línea 100 transcrita arriba y en la condición de la línea 1020; exceptuando esto, se podrían utilizar como guía para la implementación en otras máquinas.

La versión de la línea 100 para el Lynx es la siguiente:

```
100 DIM N$(4)(6)
```

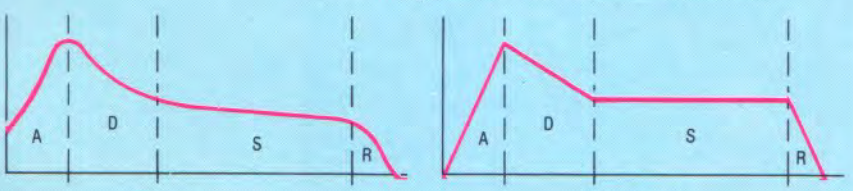
Sonidos fieles

Continuamos con el tema de la música por ordenador

Como parte del estudio acerca de la generación de sonido con un microordenador, veremos ahora algunas de las configuraciones más avanzadas.

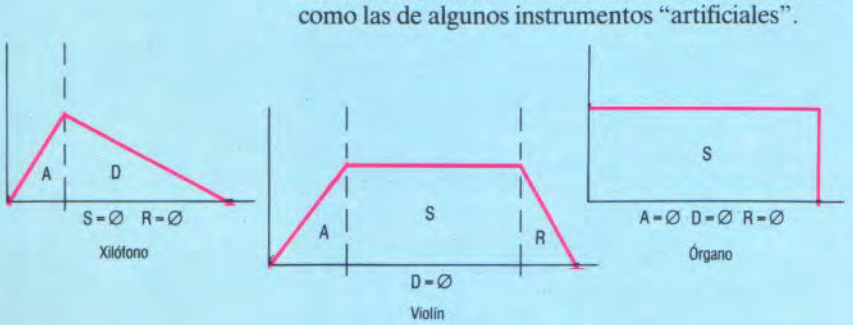
Generadores de envoltura

La *envoltura* de un sonido es el patrón de sus cambios de volumen, desde que se ejecuta hasta que se extingue. El diagrama ilustra envolventes similares a las de una nota de piano y de otros instrumentos. Las envolventes se dividen en cuatro elementos, que se suelen denominar ADSR (*Attack, Decay, Sustain, Release*: arranque, demora, sostenido y liberación). En el caso del piano, el volumen se eleva rápidamente hasta su nivel más alto una vez se pulsa la tecla (arranque), luego disminuye más lentamente (demora) hasta el nivel de volumen más constante (sostenido) mientras la tecla se mantiene oprimida, y por último cae rápidamente a cero (liberación) cuando se suelta la tecla. El sostenido es un nivel de volumen, mientras que el arranque, la demora y la liberación son intervalos de tiempo. El dispositivo capaz de controlar estos cuatro aspectos de una envoltura se denomina *generador ADSR*. Todo dispositivo capaz de encender y apagar un sonido es un generador de envoltura de clases.



La mayoría de los ordenadores no posee otra sofisticación más que un generador *beep*, que no es más que un interruptor para encender el sonido a un volumen constante durante un tiempo específico. En este caso los elementos A, D y R son iguales a cero.

El Oric-1 proporciona cierto control de envoltura, pero en este sentido los ordenadores más versátiles son el BBC Micro y el Commodore 64; ambos poseen generadores ADSR, capaces de imitar las envolventes de los instrumentos más comunes, así como las de algunos instrumentos "artificiales".



...Dibujos atrayentes

Animación sencilla utilizando las órdenes POKE y PRINT

Habiendo analizado los principios básicos de los gráficos por ordenador, ahora vamos a ver cómo se puede obtener una animación sencilla. Primero, sin embargo, será necesario detallar las formas en que se pueden hacer aparecer los caracteres en la pantalla y cómo se pueden controlar sus posiciones empleando un programa en lenguaje BASIC. El programador dispone de dos métodos principales: las órdenes POKE y PRINT, con sus correspondientes funciones.

La orden POKE coloca un número en cualquier posición de memoria especificada. En el interior de todo ordenador existe una serie especial de posiciones de memoria, cada una de las cuales está relacionada con una posición específica del carácter en la pantalla. En una pantalla estándar de 25 filas por 40 columnas, se reservan 1 000 posiciones con esta finalidad. Cada posición retiene un número que corresponde a un carácter determinado de ese juego de caracteres de la máquina. Éste puede ser el código ASCII del carácter (véase p. 214) o un código diseñado por el fabricante de la máquina. Además de estas posiciones codificadas de caracteres, por lo general existe otro juego de posiciones que retienen información relativa al color de un carácter visualizado en cualquier lugar de la pantalla.

En el Commodore 64, por ejemplo, existen muy pocas órdenes de gráficos en BASIC para ayudar al programador, y a menudo se utiliza la orden POKE para crear visualizaciones en pantalla. Las direcciones de las posiciones que retienen códigos de caracteres van de 1024 a 2023, y las posiciones que retienen la información de color poseen direcciones entre 55296 y 56295. Para el Commodore, el carácter A posee un código de pantalla 1 y el color negro se representa por un código de color 0; en consecuencia, las órdenes que se requieren para colocar una letra A negra en el ángulo superior izquierdo de la pantalla son:

```
10 POKE 1024,1
20 POKE 55296,0
30 END
```

Se pueden realizar modificaciones sencillas para visualizar una fila de letras A negras en la línea superior de la pantalla:

```
10 FOR X = 0 TO 39
20 POKE 1024 + X,1
30 POKE 55296 + X,0
40 NEXT X
50 END
```

Las letras A las produce el bucle FOR...NEXT, que cada vez incrementa en uno las direcciones de la posición del carácter y del color. Si incorporáramos una orden para borrar una A antigua cada vez que se creara una nueva, entonces parecería que la A se

moviera a través de la pantalla: ¡una forma no muy refinada de animación!

El código de caracteres del Commodore para un espacio es 32. Todo lo que se necesita es que el programa lo coloque en la posición adecuada, un lugar detrás de donde se va a visualizar la nueva A. Inserte esta línea en el programa anterior:

```
35 POKE 1024 + X,32
```

O, como alternativa:

```
15 POKE 1024 + (X – 1),32
```

Utilizar este sistema para producir gráficos es un proceso laborioso. Probablemente lo más adecuado sea crear visualizaciones de fondo estático empleando sentencias READ y DATA para dar entrada a los códigos de los caracteres antes de almacenarlos (POKE) en la posición correcta.

La mayoría de los ordenadores personales posee muchas órdenes para gráficos como parte de su juego de instrucciones en BASIC estándar, que permiten que el usuario cree notables visualizaciones, plenas de color, utilizando sólo unas pocas sentencias simples. A menudo se incluyen órdenes para construir patrones geométricos de alta resolución. Con estas instrucciones el usuario puede trazar puntos en la pantalla y unirlos entre sí por medio de líneas rectas; puede dibujar cuadrados, arcos y círculos y colorear los interiores de las formas dibujadas.

Un método muy directo suele ser el de volver a trazar formas antiguas con el mismo color que el fondo, lo que equivale a borrarlas. Trazar rápidamente, borrar y volver a trazar en una nueva posición es, de hecho, la base de la animación gráfica sencilla. El realismo de la acción depende en gran medida de la velocidad con que se pueda llevar a cabo el proceso. Los sprites son mucho más efectivos porque no requieren borrar el trazado a medida que se desplazan hacia una posición nueva, y ello aumenta notablemente la velocidad a la cual parecen moverse. En realidad, el desarrollo de los sprites significa que por primera vez es posible escribir juegos recreativos en BASIC, para lo cual antes era imprescindible utilizar el código de lenguaje máquina.

Los principios de los gráficos móviles se pueden utilizar en conjunción con programas sencillos en BASIC. Muchos ordenadores personales poseen órdenes que le permiten al usuario imprimir (PRINT) en posiciones específicas de la pantalla, como PRINT AT en el Spectrum y PRINT @ en el Dragon.

Cómo se crea una estrella

He aquí un breve programa para el Dragon utilizando PRINT. En él, la variable X es la posición en pantalla donde se ha de imprimir la estrella. Observe que la línea 40 borra la estrella antigua mientras se imprime la nueva.

```
10 CLS: REM LIMPIAR PANTALLA
20 FOR X = 160 TO 191
30 PRINT @ X,"*";
40 PRINT @ X – 1," ";
50 NEXT X
60 END
```

Forma de onda

La forma de onda es la "forma" repetitiva de la señal producida por un oscilador (véase p. 247) y le proporciona al sonido su carácter. Dos instrumentos distintos que ejecuten notas a la misma altura no suenan igual, y ello se debe en parte a que las formas de onda son diferentes. Las formas de onda más comunes son la cuadrada (o pulso), la triangular y la aserrada (véase ilustración).

La mayoría de los ordenadores personales proporciona sólo una forma de onda, por lo general del tipo de pulso. Ésta es la razón por la cual muchos de ellos poseen ese sonido sintético inconfundible y áspero.

El Commodore 64 es el ordenador más interesante desde el punto de vista musical, fundamentalmente porque el usuario puede seleccionar cualquiera de las tres formas de onda básicas en tres osciladores distintos. Las formas de onda se pueden modificar utilizando filtros, que alteran el tono de manera muy parecida a la de los controles *bass/treble* de un equipo de alta fidelidad y tienen la cualidad de suavizar el sonido. Aún más útil resulta la capacidad de cambiar estos ajustes de filtro durante la duración de una nota. Esto permite que se puedan imitar sonidos naturales con más exactitud y producir sonidos artificiales más atractivos.

Ruido

El ruido es un tipo de sonido complejo efectuado por vibraciones al azar. El oído no puede captar un patrón repetitivo, por lo cual no oye ninguna altura específica. Imaginemos algunos sonidos cotidianos como la lluvia, el viento y los truenos. Estos ruidos no suenan igual porque son una combinación de ruido puro (impredecible) con algunos tonos dominantes. La mayoría de los ordenadores que poseen capacidad de ruido le permitirán, por tanto, modular el ruido de alguna manera o mezclarlo con notas puras. Los posibles efectos van desde un "viento susurrante" hasta violentas explosiones.

Salida

La salida por lo general se realiza a través del altavoz del televisor. Si éste es el caso, el usuario puede conectar el televisor a su equipo de alta fidelidad mediante una grabadora de video. No obstante, algunos ordenadores sólo pueden dar salida al sonido a través de un pequeño altavoz incorporado. Con estas máquinas es imposible obtener un sonido de buena calidad sin modificar su hardware o comprar un accesorio externo. Puede darse el caso de que el ordenador que usted utiliza posea una salida apta para conectarla directamente a su equipo de alta fidelidad, por lo cual vale la pena el esfuerzo que implica producir formas complejas de sonido.

Cristal líquido

Las visualizaciones en cristal líquido, muy usadas en relojes y calculadoras, comienzan a aparecer en los ordenadores

Las visualizaciones en cristal líquido (LCD: *Liquid Crystal Display*) existen desde finales de 1973, cuando aparecieron por primera vez en las calculadoras. Posteriormente se emplearon en la fabricación de relojes digitales y contribuyeron en gran medida a la popularidad de esa clase de relojes. Ahora las LCD se están haciendo un lugar en la industria de los microordenadores. Se utilizan en máquinas portátiles de tamaño A-4, como el Epson HX-20 y el Tandy TRS80 Modelo 100, así como en el Sharp PC5000, de gran capacidad.

Para comprender qué son los cristales líquidos, debemos señalar, en primer lugar, que toda materia experimenta variaciones en su manera de ser a causa de la temperatura y la presión, que influyen en la mayor o menor cohesión entre sus moléculas. De este modo, pasa, sucesivamente, por tres estados: sólido (o cristalino), líquido y gaseoso. Sólo en el estado sólido se puede hallar una alineación regular de las moléculas de una sustancia. La única excepción la constituyen un pequeño número de sustancias, en las que la alineación regular se mantiene parcialmente en el estado líquido. Estas sustancias, cuya naturaleza es un secreto celosamente guardado, se conocen como *cristales líquidos*.

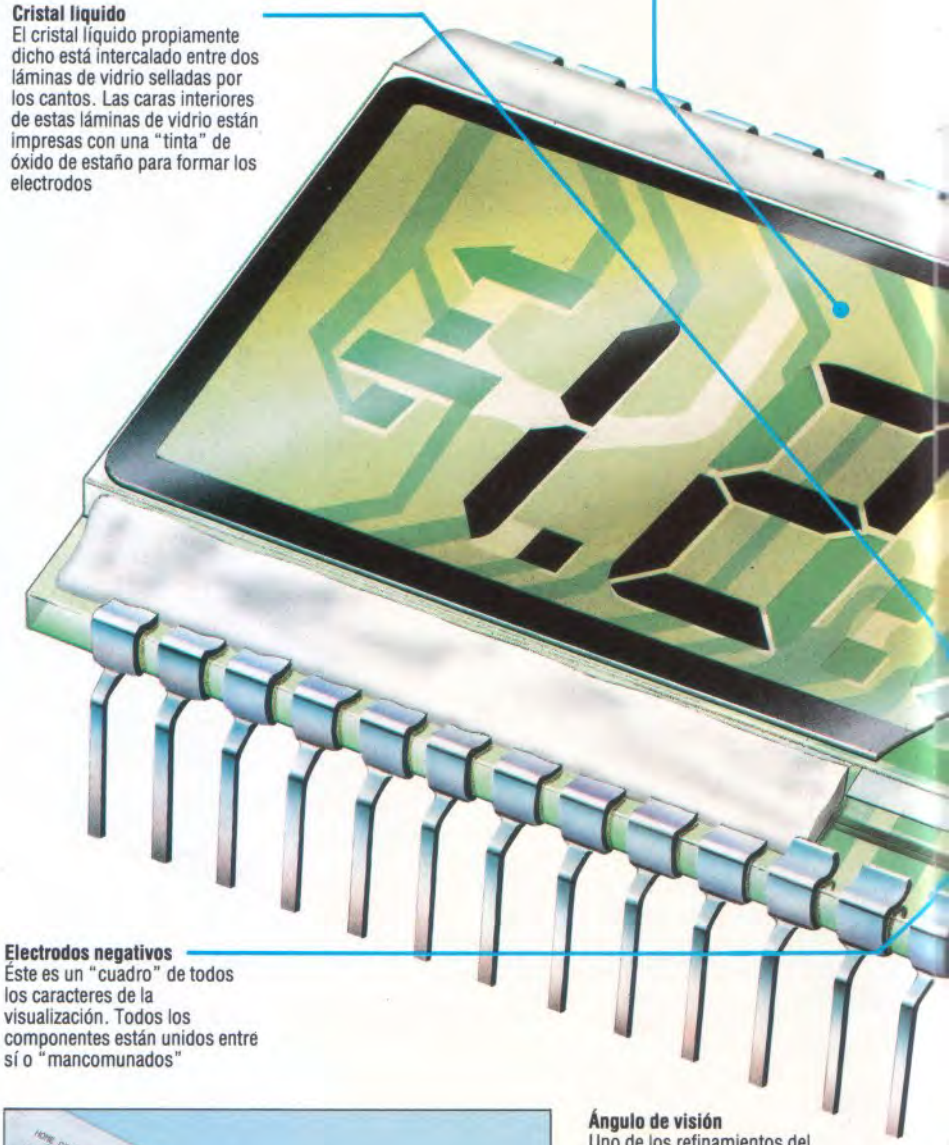
Hasta mediada la década de los setenta, las visualizaciones de las calculadoras y los relojes estaban compuestas por LED (*Light Emitting Diodes*: diodos emisores de luz) en forma de barra, dispuestos de manera que formaban una versión más bien angular de una letra o un número. Pero las visualizaciones por LED poseen varios inconvenientes: requieren considerables cantidades de energía y su tamaño es relativamente grande.

En la búsqueda de métodos alternativos para la visualización de la información, se descubrió que se podía alterar la alineación de las moléculas de los cristales líquidos mediante una corriente eléctrica; y, más aún, que esta alteración era puramente local. Una vez establecido este principio, fue posible construir un soporte para visualizar la información. El primer paso consistió en componer electrodos con la forma de un carácter, en las caras interiores de dos láminas de vidrio. Entre éstas se intercaló una capa muy delgada de cristal líquido y se aplicó un voltaje. A la luz normal pareció que no sucedía nada, pero cuando se aplicaron filtros polarizantes (véase diagrama) en las partes posterior y anterior y se montó toda la estructura contra un fondo reflector, se produjo el efecto deseado: un carácter claramente definido con un fondo neutro.

El proceso en virtud del cual se define este carácter requiere que la luz pase a través del primer filtro y que, de este modo, se polarice verticalmente. Luego se desvía 90°, y a causa de este hecho queda concentrada en el filtro posterior. De esta manera, el área del cristal líquido a la cual se le ha aplicado un voltaje aparece como un área oscura. En la ac-

Filtro de polarización vertical
El filtro de polarización de la parte anterior de la visualización rechaza toda la luz excepto las ondas luminosas que oscilan verticalmente. Incorpora un filtro ultravioleta para prolongar la vida del cristal líquido

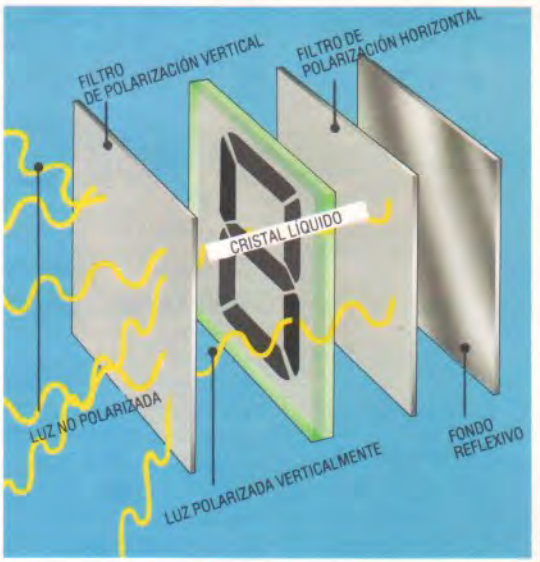
Cristal líquido
El cristal líquido propiamente dicho está intercalado entre dos láminas de vidrio selladas por los cantos. Las caras interiores de estas láminas de vidrio están impresas con una "tinta" de óxido de estaño para formar los electrodos



Electrodos negativos
Éste es un "cuadro" de todos los caracteres de la visualización. Todos los componentes están unidos entre sí o "mancomunados"

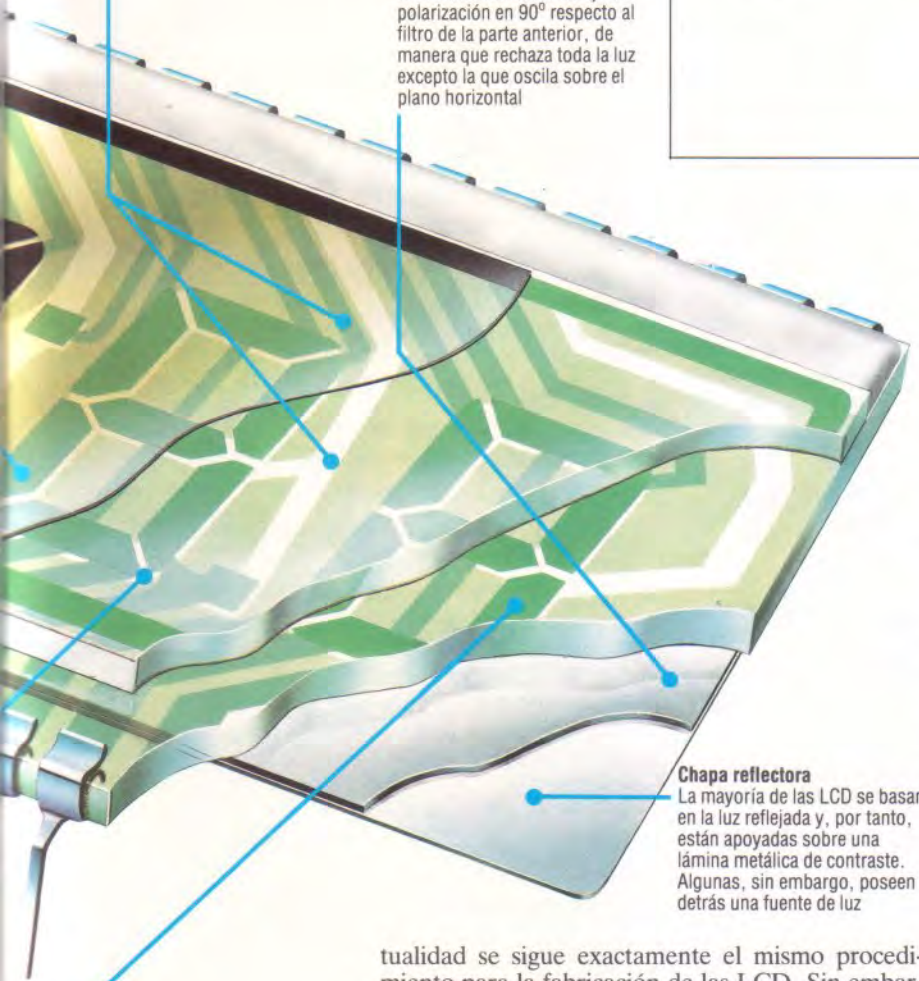
Ángulo de visión
Uno de los refinamientos del Epson HX-20, un ordenador portátil notablemente compacto y sofisticado, es el ajustador del ángulo visual. Los cristales líquidos están compuestos por moléculas largas y delgadas que poseen polos magnéticos situados en el medio de sus lados largos. La aplicación de un voltaje eléctrico a través de su longitud hace que intenten enroscarse por los extremos, mientras su estabilidad natural intenta mantenerlas en su lugar. Cuanta más corriente se les aplique, más se enroscarán

Polarización
El filtro de la parte anterior de una visualización en cristal líquido sólo permite que pasen a través de él los rayos de luz que tengan la oscilación electromagnética orientada verticalmente. El cristal líquido gira luego la "polarización" en 90°, permitiendo que pase la luz a través del filtro posterior (horizontal) y vuelva a ser reflejada. Cuando se aplica energía a un segmento de la LCD, la polarización deja de rotar, y el resultado es una imagen negra



Pasos de conexión
Cada electrodo está conectado a su circuito activador por medio de un depósito casi invisible de tinta conductora, situado en la superficie ya sea del vidrio anterior o del posterior

Filtro de polarización horizontal
El filtro de la parte posterior de la visualización tiene su eje de polarización en 90° respecto al filtro de la parte anterior, de manera que rechaza toda la luz excepto la que oscila sobre el plano horizontal



Chapa reflectora
La mayoría de las LCD se basan en la luz reflejada y, por tanto, están apoyadas sobre una lámina metálica de contraste. Algunas, sin embargo, poseen detrás una fuente de luz

Electrodos positivos
En esta representación de los caracteres, los circuitos activadores pueden direccionar individualmente a cada uno de los componentes, que están separados

tualidad se sigue exactamente el mismo procedimiento para la fabricación de las LCD. Sin embargo, los electrodos están impresos en tinta traslúcida, incolora, sobre la superficie del vidrio, y al secarse quedan casi invisibles.

A causa de que es necesario producir una variedad de caracteres a partir de la misma matriz, aún se utiliza el método original (una combinación de barras cortas que forman caracteres angulares), si bien cada vez están siendo más comunes las LCD matriciales. Debido a que se puede direccionar individualmente cada punto de la matriz, las capacidades para gráficos son muy avanzadas: se pueden producir tanto formas continuas como caracteres para gráficos convencionales.

Teóricamente es posible emplear LCD matriciales direccionadas para que actúen como las pantallas de los receptores de emisiones de televisión, así como monitores para ser utilizados con ordenadores. El principal inconveniente para ello es la carencia de variabilidad del contraste. Éste fue uno de los problemas que se consideraron mientras se desarrollaban los diversos televisores de pantalla plana que están empezando ahora a salir al mercado. Es necesario reducir el tamaño de los elementos individuales de la matriz hasta un nivel aproximado al de un pixel de un tubo de rayos catódicos, con el fin de conseguir una resolución aceptable. Otra posibilidad requeriría la utilización de varias LCD intercaladas entre sí y funcionando de manera simultánea. Este método se emplea habitualmente para la visualización de información compleja que necesita un espacio más amplio.

El tiempo de respuesta de una LCD de gran calidad a la temperatura normal de funcionamiento (20 °C) es de 70 milisegundos aproximadamente para la subida de neutro a negro, y 80 milisegundos más para la bajada a neutralidad otra vez. Este total de 150 milisegundos significa una clara desventaja en comparación con la respuesta de 0,00025 milisegundos del tubo de rayos catódicos. No obstante, la LCD cuenta con ciertas ventajas. Ya hemos analizado su tamaño compacto, pero quizá la mayor ventaja sea su escaso consumo energético. Una LCD típica consume apenas 10 microvatios por centímetro cuadrado de visualización.

Al variar el voltaje que pasa a través de un cristal líquido se produce un efecto muy interesante. La inclinación de las moléculas aumenta y disminuye de acuerdo con el que se aplique. La firma Epson, por ejemplo, utiliza muy bien este efecto en el ordenador portátil HX-20, confiriéndole a la visualización un ajuste del ángulo de visión.

Cuando la luz del sol es intensa se hace evidente una ventaja adicional. El contraste de una visualización por tubo de rayos catódicos disminuye sustancialmente, pero como los cristales líquidos se perciben por la *ausencia* de luz reflejada por ellos, las LCD adquieren mayor contraste y, por lo tanto, son más legibles cuanta más luz exterior haya.

A pesar de que sólo cuenta con diez años de existencia, la tecnología de las visualizaciones en cristal líquido ya ha hecho incursiones significativas en mercados tradicionalmente reservados a los tubos de rayos catódicos. En el futuro, y en la medida en que aumenten las exigencias de microminiaturización, se espera que esta tecnología continúe desarrollándose.

Respuestas a los ejercicios

He aquí algunas soluciones a los problemas de la página 235, si bien usted puede haber encontrado otros métodos alternativos

En la página 235 le propusimos nueve problemas, concebidos para comprobar su habilidad en la utilización de las sentencias y funciones más usadas en BASIC. He aquí las soluciones que sugerimos.

Si ha seguido el curso de programación BASIC desde el principio, quizá haya identificado los ejercicios con problemas con los que ya nos habíamos encontrado antes y que habíamos resuelto. Sus soluciones pueden ser distintas a las nuestras. Es muy raro que exista sólo una manera de resolver un problema; el procedimiento adoptado por usted puede ser tan bueno como el nuestro o incluso mejor.

Si encuentra que las soluciones son tan enigmáticas como las preguntas, vuelva a leer el curso desde la página 149 y vaya estudiando las soluciones a medida que avance. Si ha logrado resolver los ejercicios 2, 4, 6 y 8, ha comprendido la mayoría de las lecciones de nuestro curso.

```
100 REM EJERCICIO DE REVISION 1
200 INPUT "DIGITE CUALQUIER NUMERO";A
300 INPUT "DIGITE OTRO NUMERO";B
400 LET C = A + B
500 PRINT "SU SUMA ES";C
```

```
100 REM EJERCICIO DE REVISION 2
200 LET A$ = "PRIMERA PALABRA"
300 LET B$ = "SEGUNDA PALABRA"
400 LET C$ = A$ + B$
500 PRINT C$
```

```
100 REM EJERCICIO DE REVISION 3
200 INPUT "DIGITE CUALQUIER PALABRA";P$
300 LET L = LEN(P$)
400 PRINT "LA PALABRA QUE HA DIGITADO
    TIENE";L;"CARACTERES"
```

```
100 REM EJERCICIO DE REVISION 4
200 PRINT "PULSE CUALQUIER TECLA"
300 FOR C = 0 TO 1 STEP 0
400 LET A$ = INKEY$
500 IF A$ <> " " THEN LET C = 2
600 NEXT C
700 PRINT "EL VALOR ASCII DE";A$;
    "ES";ASC(A$)
```

Véase "Complementos al BASIC", de pp. 175 y 215. En el Spectrum, reemplazar la línea 700 por:

```
700 PRINT "EL VALOR ASCII DE";A$;
    "ES";CODE(A$)
```

```
100 REM EJERCICIO DE REVISION 5
200 INPUT "DIGITE UNA PALABRA";P$
300 LET L$ = RIGHT$(P$,1)
400 PRINT "LA ULTIMA LETRA DE LA
    PALABRA ERA ";L$
```

Véase el recuadro "Complementos al BASIC" de p. 149. En el Spectrum este ejercicio se lee:

```
100 REM EJERCICIO DE REVISION 5
200 INPUT "DIGITE UNA PALABRA";P$
250 LET N = LEN(P$)
300 LET L$ = P$(N)
400 PRINT "LA ULTIMA LETRA DE LA
    PALABRA ERA";L$
```

```
100 REM EJERCICIO DE REVISION 6
200 PRINT "DIGITE UN NOMBRE EN LA FORMA:"
300 PRINT "NOMBRE Y PRIMER APELLIDO"
400 PRINT "P. EJ. ROSA TORRES"
500 INPUT "NOMBRE";N$
600 LET S = 0:LET L = LEN(N$)
700 FOR P = 1 TO L
800 IF MID$(N$,P,1) = " " THEN LET S = P
900 NEXT P
950 PRINT "EL ESPACIO ERA EL";S;"º CARACTER"
```

Véase "Complementos al BASIC", p. 149. En el Spectrum, reemplazar la línea 800 anterior por:

```
800 IF N$(P) = " " THEN LET S = P
```

```
100 REM EJERCICIO DE REVISION 7
150 LET X$ = "º"
200 PRINT "DIGITE UN NOMBRE EN LA FORMA:"
300 PRINT "NOMBRE Y PRIMER APELLIDO"
400 PRINT "P. EJ. ROSA TORRES"
500 INPUT "NOMBRE";N$
600 LET S = 0: LET L = LEN(N$)
700 FOR P = 1 TO L
800 IF MID$(N$,P,1) = " " THEN LET S = P
900 NEXT P
925 IF S = 3 THEN LET X$ = "ER"
950 PRINT "EL ESPACIO ERA EL";S;X$;
    "CARACTER"
```

```
100 REM EJERCICIO DE REVISION 8
200 INPUT "DIGITE UNA ORACION";O$
300 LET C = 1
400 FOR P = 1 TO LEN (O$)
500 IF MID$(O$,P,1) = " " THEN LET C = C + 1
600 NEXT P
700 PRINT "LA ORACION QUE DIGITO
    TENIA";C;"PALABRAS"
```

Véase "Complementos al BASIC", p. 149. En el Spectrum, reemplazar la línea 500 anterior por:

```
500 IF O$(P) = " " THEN LET C = C + 1
```

```
100 REM EJERCICIO DE REVISION 9
200 FOR C = 128 TO 255
300 PRINT "CARACTER N.º";C;"=";CHR$(C)
400 REM BREVE DEMORA AQUI
500 FOR D = 1 TO 500
600 NEXT D
700 REM FIN DE LA DEMORA
800 NEXT C
```

Su fiel servidor

En la actualidad los robots industriales pueden reconocer objetos y aprender nuevas tareas imitando las acciones humanas

La palabra "robot" (del checo *robota*: trabajo) fue acuñada en 1920 por el escritor checo Karel Čapek en su obra teatral *R.U.R. (Rossum's universal robots*: *Los robots universales de Rossum)* para denominar a un androide creado por un científico y capaz de llevar a cabo trabajos realizados tradicionalmente por un hombre. El término fue acogido con gran entusiasmo por los escritores de ciencia-ficción. A pesar de los numerosos relatos novelescos que narran los poderes de los robots, éstos no son más que una extensión electromecánica del ordenador, con todas las limitaciones e imperfecciones propias de estas máquinas.

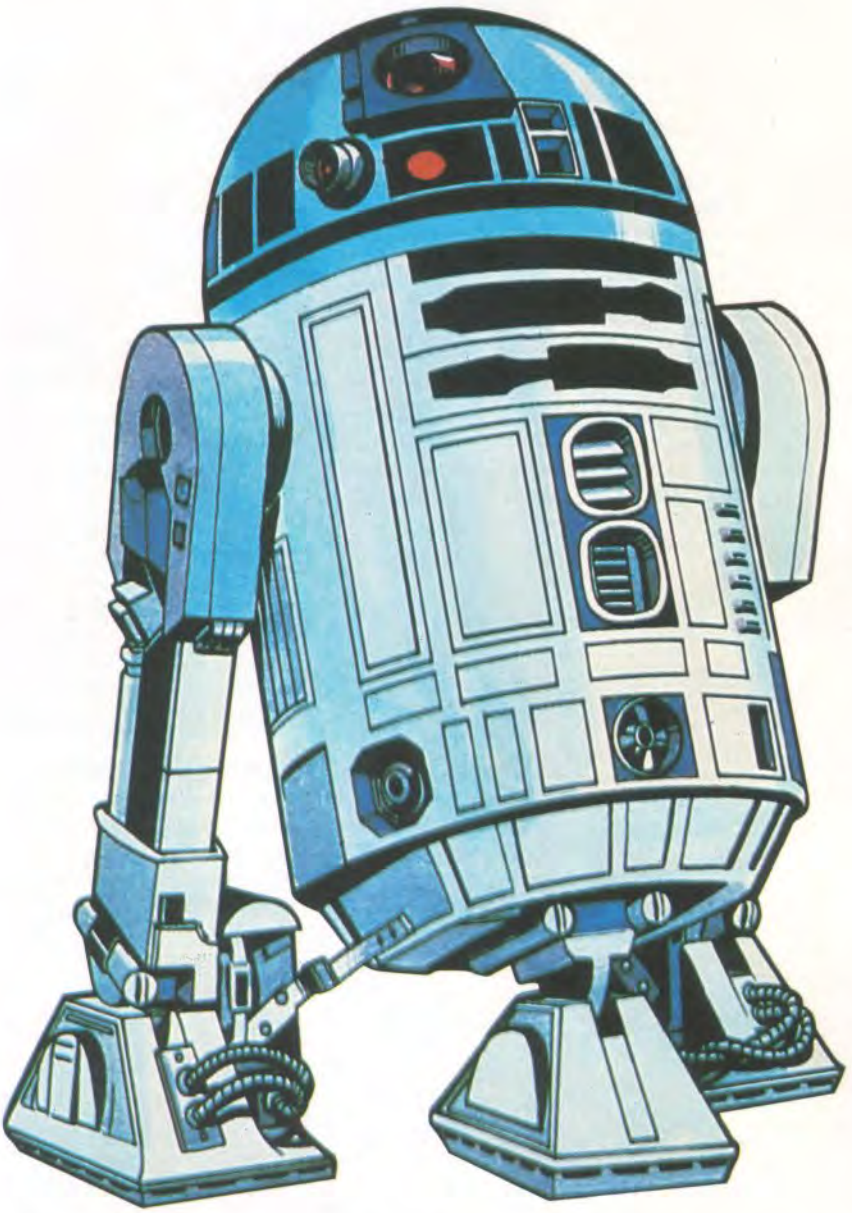
Sus orígenes se remontan a los talleres de maquinaria de los años cincuenta, en los que se aplicó por primera vez la teoría del control numérico a las máquinas-herramienta. Estos primeros esfuerzos fueron, previsiblemente, muy elementales: máquinas controladas por cinta de papel de cinco agujeros (del tipo de las que utilizan las máquinas de télex) que, en el mejor de los casos, sólo podían mover una herramienta fija de un punto a otro alrededor del objeto sobre el cual estaban trabajando.

El siguiente paso de su desarrollo fue conseguir que pudieran cambiar de herramienta en el transcurso de la faena. Esto se logró mediante la utilización de un "carrusel" o soporte de herramientas rotatorio; todas ellas tenían fijaciones idénticas, que se podían seleccionar y ajustar al soporte de herramientas bajo el control del programa.

Incluso con esta refinación, una máquina determinada sólo era capaz de realizar un único tipo de tarea: un torno seguía siendo un torno, aun cuando pudiera hacer todos los trabajos de tornería requeridos para un proceso determinado. Por esa misma época se estaban desarrollando manos y brazos accionados por control remoto para trabajar en medios peligrosos: bajo el océano, por ejemplo, o en laboratorios donde se manipulaban elementos radiactivos. Estos dispositivos manipuladores eran meras extensiones de las manos del operario, pero enseguida se comenzaron a utilizar ordenadores para controlarlos directamente. Los robots que se han desarrollado después son, aplicando un término más preciso, "brazos robot", pues consisten en un soporte de herramientas montado sobre un brazo extensible o articulado.

Si deseamos comprender cómo se programan los robots, primero debemos considerarlos en relación al espacio en el cual operan. La mayoría de los robots industriales ocupan una posición fija, de modo que el espacio será una esfera aplanada en su parte inferior, y podemos pensar en la cuestión del control del robot como un simple ejercicio de geometría tridimensional. El centro del esferoide será la articulación de los "hombros" del robot, y el radio será la longitud del brazo extendido, medido desde el "hombro" hasta la punta de los "dedos": la uña o soporte de la herramienta. Cualquier punto dentro de este espacio se puede expresar como tres coordenadas: por ejemplo, como distancias norte-sur, este-oeste y arriba-abajo, desde una posición cero o "punto de referencia". En este caso las coordenadas se denominan "cartesianas", en honor del filósofo y matemático francés René Descartes (1596-1650). Alternativamente, la posición se puede expresar por coordenadas esféricas. En lenguaje llano esto equivaldría a decir: "a una distancia de dos metros en dirección nordeste y treinta grados sobre la horizontal". En este caso el punto de referencia es el "hombro" del robot.

No obstante, el problema de programar al robot



Héroe del cine
R2D2, el cautivador robot de *La guerra de las galaxias*, en realidad estaba gobernado por un operador humano. Su diseño, sin embargo, reflejaba el aspecto que, según suele creerse, debe tener un robot

Un robot a pilas
El Hero-1 es un robot a pilas totalmente autocontenido que combina algunas de las funciones de una tortuga con la capacidad de manipulación de un brazo robot. Constituye un sistema por ordenador notablemente flexible, con configuraciones tan avanzadas como sintetizador de voz, sensores de nivel de luz, entrada auditiva y (debido a que es móvil) un enfocador ultrasónico de alto alcance que también actúa como detector de movimiento

Ian McKinnel

implica darle una serie de instrucciones acerca del movimiento desde un lugar hacia otro, de modo que existe aún un tercer procedimiento de determinar la posición del soporte de herramienta. Denominado *posicionamiento punto a punto*, este método requiere que el punto de referencia se mueva con el soporte de herramienta.

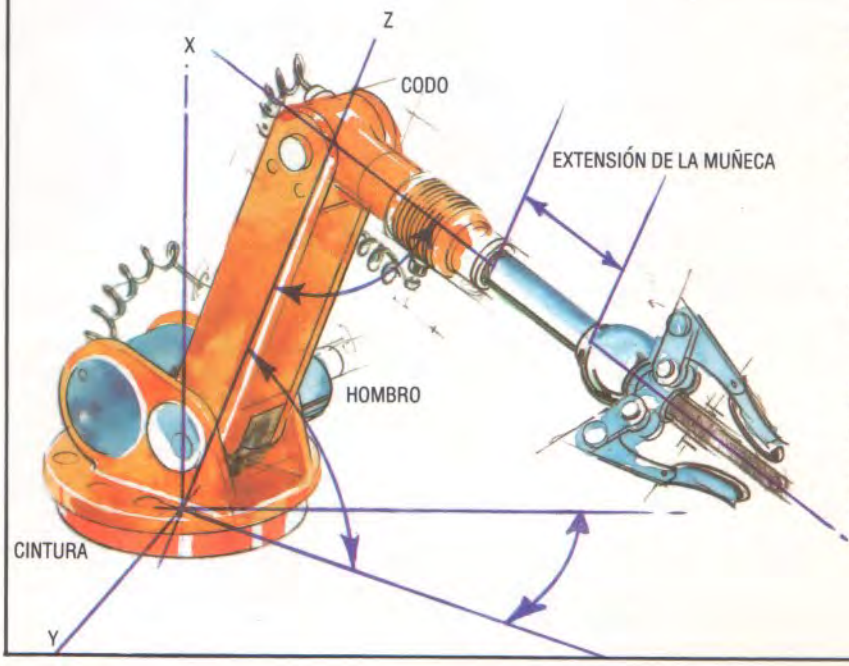
Por lo general, el margen de precisión de los robots industriales es de un milímetro. Incluso los modelos más sencillos (que se pueden utilizar con cualquier ordenador personal que posea una salida en paralelo de ocho bits) tienen un margen de precisión de dos milímetros.

Existen dos métodos generalmente aceptados para accionar los brazos robot. Para aquellos de poca carga útil, son suficientes los motores paso a paso (motores eléctricos que se mueven en un margen preestablecido cada vez que se les aplica corriente, como los que se emplean en las unidades de disco para colocar en posición la cabeza de lectura-escritura). Pero para los brazos robot que se utilizan en una cadena de producción, donde se necesita maniobrar pesos mucho mayores, es mucho más común emplear arietes hidráulicos para mover las diversas partes del brazo alrededor de su fulcro (los puntos alrededor de los cuales rotan). Resulta bastante sencillo medir el volumen de fluido hidráulico que pasa por los arietes y deducir del mismo el movimiento al otro extremo, de acuerdo con las exigencias operacionales de precisión.

Los robots industriales contienen un miniordenador construido especialmente (o, en los modelos más recientes, un microordenador de gran capacidad) y cuya exclusiva función es controlar el brazo y ejecutar un lenguaje de programación diseñado con esa finalidad. Dado que no se precisa otra exigencia que indicar coordenadas e impartir órdenes simples como CLOSE GRIPPER (cerrar uña) u OPEN GRIPPER (abrir uña), el lenguaje de programación no contiene instrucciones para manipular textos. A las instrucciones del programa se les da entrada a través de un teclado numérico agrandado acoplado al ordenador mediante un largo "cordón umbilical", de modo que el operador se pueda mover alrededor del brazo robot mientras da entrada a las instrucciones. Las versiones más avanzadas de estos *paneles colgantes*, como se denominan, incluyen una palanca de mando de precisión.

Otro procedimiento de programación, conocido como *Follow me* (Sígame), es especialmente útil para tareas que no precisen una colocación exacta de la herramienta, como cuando se pinta con una pistola pulverizadora. En este caso, el brazo del robot posee un dispositivo que permite al operador empuñar el soporte de herramienta, desplazarlo de manera exacta alrededor de la posición de trabajo y dar entrada a estos movimientos directamente en la memoria del ordenador. El robot, entonces, los repetirá cada vez que el programa se ejecute.

Movimiento angular



Uno de los aspectos más difíciles que implica la programación de un brazo robot es el de realizar la conversión de la geometría, traduciéndola en movimiento. Estamos acostumbrados a especificar las posiciones utilizando ejes cartesianos o coordenadas x, y, z. Lo que necesita el robot son ángulos para la articulación del "codo", la articulación del "hombro", la rotación de la "cintura" y la distancia a la cual se debe extender la "muñeca". En los sistemas más simples los programadores han de dar los valores para estos cuatro ángulos. Los robots más sofisticados pueden realizar todas las conversiones de las coordenadas cartesianas

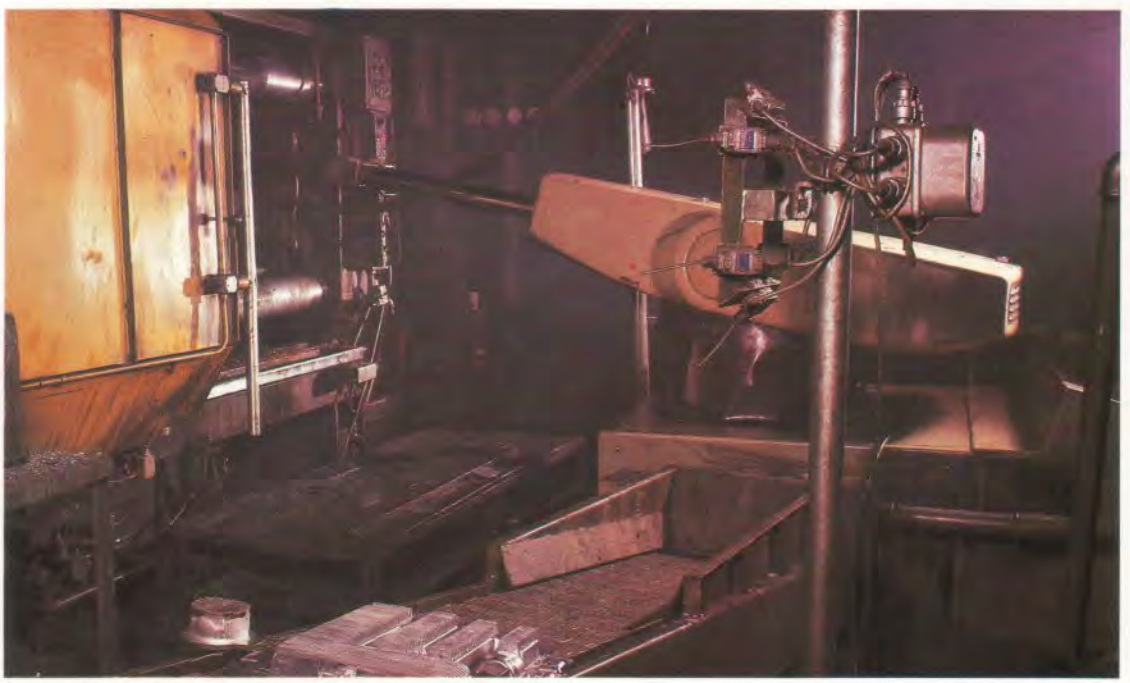
Bob Freeman

En todos estos métodos, la posición que se define es la del soporte de herramientas. El operador no necesita preocuparse de las posiciones relativas de las secciones individuales del robot: el lenguaje de programación incorporado en el ordenador de control del robot calcula cuáles deberían ser. También efectúa toda la optimización necesaria, asegurándose de que la herramienta se desplace de un lugar a otro por el camino más corto posible. La orientación del soporte de herramienta se controla de manera automática, manteniendo posiciones relativas tanto horizontales como verticales, a menos que se lo instruya en otro sentido. La velocidad del movimiento de punto a punto también es automática: el soporte de herramienta se desengancha despacio, se mueve rápidamente hasta acercarse al punto de destino y después vuelve a avanzar despacio para reengancharse a la pieza en el nuevo lugar.

Los robots de los que hemos hablado hasta ahora sólo son capaces de una "obediencia ciega", repitiendo la misma tarea exactamente en la misma localización, sin inmutarse por las influencias externas. Se los utiliza fundamentalmente en la industria de la ingeniería, en especial para la producción de vehículos motorizados. Ya hace mucho tiempo que ésta se ha organizado en cadenas de producción, en las cuales el componente o el vehículo parcialmente completo está siempre localizado con precisión en el espacio y el tiempo. Esto es de vital importancia para el buen funcionamiento de un proceso de fabricación por robot, porque si el componente no estuviera situado de forma correcta, el robot no adaptaría su movimiento en la debida forma. Con el objeto de superar este inconveniente, se pueden acoplar diversos sensores al soporte de herramienta. El más sencillo de estos sensores puede ser un microinterruptor convencional. En el programa de control se pueden incorporar planes de contingencia —por ejemplo, una orden WAIT (esperar)— para que se ejecuten en el caso de que el interruptor no haga contacto con la pieza; pero la aplicación de planes más sofisticados requeriría la intervención humana.

Una alternativa a la detección por presión implicaría la utilización de un sensor de luz. Si se colocara una fuente luminosa de modo que la pieza hiciera que ésta quedara oscurecida para el sensor del soporte de herramientas, éste se podría detener antes de que llegara al punto de colisión, se podría poner en modalidad WAIT hasta que la pieza estuviera bien colocada, y después permitir que continuara. Por supuesto, esto tampoco sería absolutamente seguro, y en las situaciones para las que se requiere una fiabilidad total se puede instalar un sistema de reconocimiento de imagen basado en cámaras de televisión CCD (*Charge-Coupled Device*: dispositivo de carga acoplada). Estas cámaras centran la imagen de manera precisa en un microchip de procesamiento por matriz (un chip dividido en cien o más fotosensores individuales, cada uno de los cuales puede registrar no sólo el blanco o el negro sino también una gama de tonos intermedios). Cada sensor individual puede requerir un byte de memoria para definir el contraste en la escala del gris. Inicialmente cada objeto se "fotografía" cierto número de veces y un programa de aprendizaje calcula el promedio de los valores. Al tiempo de ejecución, la cámara CCD toma del objeto una imagen, que luego se compara con la imagen de referencia de la memoria. Si las dos imágenes concuerdan, entonces la operación puede seguir adelante. Por este método se puede verificar que la pieza sea la correcta y su disposición la adecuada.

Este sistema de tratamiento de la imagen también se emplea para la selección de componentes en una "bolsa mezclada". Esta aplicación de "recogida y colocación", como se denomina, está siendo cada vez más utilizada para los robots pequeños como complemento de una cadena de producción regular. Además de en el proceso de producción propiamente dicho, los robots industriales se emplean en las etapas de prueba y control de calidad, a menudo a pares para posibilitar un mayor grado de flexibilidad en la colocación del producto en su posición correcta.



Jornada en la fábrica
Los brazos robot, como el que en la fotografía vemos trabajando en un taller de fundición, están tomando cada vez más bajo su cargo las tareas sucias, peligrosas y repetitivas de la industria. La limpieza de las piezas fundidas previa a su introducción en las máquinas constituye un buen ejemplo. La fundición, recién moldeada, es excesivamente caliente para las manos humanas y, por lo tanto, se colocaría a un lado hasta que se enfriara. Sin embargo, el robot no es sensible al calor, de modo que puede manipularla de inmediato y despacharla a la operación siguiente

Haciendo sonar el Vic

Una mirada a la generación de sonido del Vic-20...

El Vic-20 fue uno de los primeros ordenadores personales que salieron al mercado. Como consecuencia de ello, sus configuraciones pueden parecer poco satisfactorias en comparación con ordenadores más recientes. Además, Commodore no facilita de manera especial la creación de sonido ni de programas de música, ya que el BASIC del Vic-20, así como el BASIC del Commodore 64, no poseen órdenes relacionadas específicamente con el sonido. Todo el control de sonido se consigue mediante una serie de órdenes POKE en posiciones de memoria. Este principio se aplica también al Commodore 64, y las técnicas que aquí describimos para el Vic-20 le serán útiles al usuario del citado ordenador. El grado de control de sonido disponible se limita a volumen (equivalente a envoltura de A = D = R = 0), frecuencia en tres osciladores y un generador de ruidos. La salida sólo es posible a través del altavoz del televisor. Además, debido a las imprecisiones inherentes a la forma en que el Vic-20 selecciona las frecuencias, es imposible obtener la altura correcta para todas las notas de la escala musical.

Con sólo estas capacidades, el Vic-20 se muestra muy limitado si se quiere crear música; aunque, pensando mucho, con paciencia y con un poco de conocimiento de programación en BASIC, estas limitadas configuraciones se pueden utilizar para crear "melodías" de dos y tres acordes.

Control de sonido

El Vic-20 viene con tres osciladores de ondas cuadradas y un generador de ruidos. Cada oscilador abarca aproximadamente tres octavas de sonido, con la frecuencia compensada del siguiente modo:

Osc.1	Osc.2	Osc.3	Escala de frec. (Hz)	Octava
•			(65,41-123,47)	1
•	•		(130,81-246,94)	2
•	•	•	(261,63-493,88)	3
	•	•	(523,25-987,77)	4
		•	(1046,5-1975,53)	5

Esta distribución le permite al usuario cubrir cinco octavas en total con al menos un oscilador disponible en cada octava. La octava 3, que empieza en *do mayor* y contiene la referencia estándar *la* en 440 Hz, está disponible en los tres osciladores.

El control de los osciladores se ejerce modificando los contenidos de cinco posiciones de memoria de la siguiente manera:

Posición de memoria	Oscilador
POKE 36874,X	1
POKE 36875,X	2
POKE 36876,X	3
POKE 36877,X	ruido

En cada caso X es un número entero entre 135 y 241 (0 apaga ese oscilador), que se refiere a una tabla de valores de notas equivalentes reseñada en el folleto que acompaña al Vic-20. Para que se pueda escuchar la frecuencia seleccionada se debe establecer el nivel de volumen del siguiente modo:

```
POKE 36878,V
```

donde V se puede establecer entre 0 (apagado) y 15 (alto) afectando a los tres osciladores y al ruido. Por ejemplo:

```
POKE 36874,219:POKE 36875,219:POKE
  36876,219:POKE 36878,7
```

Esto hace tocar la referencia *la* a 440 Hz en el oscilador 1, *la* una octava más alta en el oscilador 2 y *la* una octava aún más arriba en el oscilador 3, todas ellas a un volumen medio de 7. ¡No se olvide de colocar POKE cada posición en 0 para apagarlas!

Notas y pausas

Sin una duración para cada nota y sin las pausas adecuadas entre ellas, en una secuencia de notas éstas se confundirían entre sí. Para facilitar estos períodos de "espera", se puede emplear uno de dos métodos para lograr que el ordenador "marque el compás" entre un POKE y otro. El primer procedimiento es el de los bucles FOR...NEXT, donde la pausa se sincroniza mediante un bucle vacío largo como:

```
10 POKE 36878,7
20 POKE 36876,203
30 FOR P = 1 TO 200
40 NEXT P
50 POKE 36878,0
60 POKE 36876,0
```

Esta secuencia de órdenes hace que se toque la nota *re #* para 200 operaciones FOR...NEXT. Sin embargo, la exactitud de este método depende de una cuidadosa sincronización externa del bucle. Una forma más sencilla y más elegante de establecer las duraciones y las pausas consiste en utilizar el reloj incorporado del Vic-20, que cuenta en sesentavos de segundo (*jiffys*) y al que se puede hacer referencia dentro de un programa empleando la variable TI. Ésta es extremadamente útil, ya que una orden se puede construir como para que "espere" durante un lapso medido con suma precisión, de este modo:

```
10 POKE 36878,7
20 POKE 36876,203:RE = TI
30 IF TI-RE < 15 THEN 30
40 POKE 36878,0
50 POKE 36876,0
```

Estas órdenes tocan la misma nota que antes pero durante un período de 15 *jiffys* (la cuarta parte de un segundo). Cuando se enciende el sonido, RE se establece en el valor de TI. La línea 30 cuenta 15 *jiffys* antes de proseguir a la línea 40. Se pueden construir melodías utilizando el mismo principio para hacer una pausa antes de tocar una nota diferente, y así sucesivamente. La próxima vez que nos ocupemos del Vic-20 en nuestra serie "Sonido y luz", estudiaremos cómo tocar melodías.

Iluminando el Dragon

... y a la capacidad para gráficos del Dragon 32

El ordenador Dragon 32 incorpora un dialecto particular de BASIC que se conoce como *Microsoft Extended Colour Basic*. Además del Dragon, existen en el mercado otros varios ordenadores que poseen también esta versión de BASIC, entre los que destaca la gama de ordenadores en color Tandy. El BASIC Microsoft es sencillo de utilizar y dispone de una buena gama de órdenes para trazar líneas, círculos y otras formas geométricas. Una vez dibujadas, estas formas se pueden colorear, obteniendo con un escaso esfuerzo de programación unas visualizaciones en pantalla notables.

El Dragon 32 posee siete niveles de resolución, proporcionándole al usuario la capacidad de trabajar con la pantalla dividida en 512 puntos individuales en el nivel más bajo, y en hasta 49 152 puntos en el nivel más alto. Hay ocho colores disponibles, pero al trabajar con alta resolución la elección se puede limitar a cuatro o incluso a dos colores.

Modalidades de resolución

La pantalla normal de 16 filas por 32 columnas de caracteres conforma el nivel de resolución más bajo, y la orden PRINT@ permite colocar al carácter en cualquiera de las 512 posiciones de pantalla. Además del juego de caracteres normal, existen 16 caracteres para gráficos de baja resolución disponibles en ocho colores.

La siguiente modalidad de resolución divide a la pantalla en 32 filas y 64 columnas. En esta modalidad, el tamaño de cada cuadrado es, por lo tanto, equivalente a la cuarta parte del de un carácter normal. Los puntos de este tamaño se pueden trazar en la pantalla mediante la orden SET y se pueden borrar mediante la orden RESET.

Las dos modalidades anteriores se pueden visualizar al mismo tiempo y se llaman pantallas de texto de baja resolución. Existen asimismo cinco niveles de pantallas de alta resolución, pero éstos no se pueden visualizar simultáneamente o en las pantallas de bajo nivel. Las cinco modalidades de alta resolución ofrecen opciones basadas en el estándar de resolución y el número de colores disponibles, y se seleccionan utilizando la orden PMODE.

PMODE	Resolución	Colores disponibles
0	128*96	2
1	128*96	4
2	128*192	2
3	128*192	4
4	256*192	2

Existe, por supuesto, una interrelación entre resolución, color y la cantidad de memoria necesaria para almacenar la información en pantalla, y esto se debe tener en cuenta al escribir programas largos en BASIC que también utilicen visualizaciones de alta resolución.

Aunque sólo disponga de un número limitado de colores en alta resolución, el Dragon posee la capacidad de seleccionar uno de dos juegos de colores. Esto se consigue mediante la orden SCREEN. Por ejemplo, SCREEN 1,0 selecciona una pantalla de alta resolución y el juego de colores 0; SCREEN 1,1 selecciona también una pantalla de alta resolución pero con un juego de colores alternativo.

PAINT
Esta orden resulta muy útil para ayudar al programador a crear imágenes interesantes. La utilización de PAINT hace que el ordenador comience a colorear la pantalla desde un punto determinado hasta llegar a una línea limítrofe. Esto significa que se puede rellenar fácilmente círculos, triángulos y cualquier forma cerrada.

DRAW
DRAW imita el movimiento del lápiz sobre la pantalla, permitiéndole al usuario trazar líneas en una de cuatro direcciones. Con la orden DRAW también se puede hacer girar o agrandar la imagen terminada.

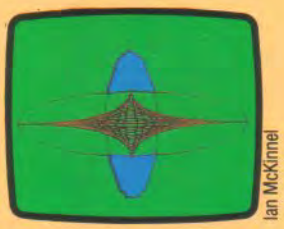
GET y PUT
GET instruye al ordenador para que almacene en su memoria una visualización en pantalla, y PUT hace que dicha visualización se vuelva a imprimir en la pantalla.

PSET y PRESET
Estas órdenes son los equivalentes para alta resolución de SET y RESET que ya hemos analizado antes y que encienden o apagan un punto determinado de la pantalla. También se puede determinar el color del punto.

LINE
La orden LINE hace que, en alta resolución, dos puntos especificados se unan entre sí por una línea recta.

CIRCLE
CIRCLE permite dibujar círculos de alta resolución con un centro y un radio determinados. También cabe dibujar porciones de un círculo completo para formar arcos y se puede condensar la forma circular para producir elipses.

El Dragon 32 es un ordenador que posee muchas órdenes avanzadas para ayudar a la programación de gráficos. Es más adecuado para aplicaciones que impliquen visualizaciones estáticas que para aquellas que requieran una acción de movimiento rápido. Especialmente las órdenes de la modalidad de alta resolución hacen del Dragon 32 un ordenador ideal para los niños de mentalidad emprendedora. El principal inconveniente es su incapacidad para visualizar simultáneamente en la pantalla texto y gráficos de alta resolución. Esto significa que no se puede utilizar para visualizar datos estadísticos en forma de diagramas de barras o cuadros.



Ian McKinnel

Orden de color
Esta visualización constituye un ejemplo típico de los efectos que se pueden obtener en un Dragon utilizando algunas de sus órdenes de alto nivel

Alta resolución
He aquí un breve programa para el Dragon 32 que demuestra algunas de sus capacidades de alta resolución. El programa emplea PMODE 3; ésta no es la modalidad más alta, pero admite cierta utilización de color

```
10 PCLS:PMODE3,1
20 SCREEN 1,0
30 COLOR 0,1
40 FOR X = 0 TO 127 STEP 10
50 LINE(X,85)-(127,85-X/3),
   PSET
60 LINE(X,85)-(127,85+X/3),
   PSET
70 LINE (255-X,85)-(127,85-
   X/3), PSET
80 LINE (255-X,85)-(127,85+
   X/3),PSET
90 NEXT X
100 CIRCLE(127,85),128,4,0.3
110 CIRCLE(127,85),30,4,3
120 PAINT(130,30),3,4
130 PAINT(130,130),3,4
140 GOTO 140
150 END
```

Ordenando la baraja

En la mayoría de los programas resulta esencial su capacidad para clasificar la información, y existen muchas formas de hacerlo

Clasificación burbuja
Este diagrama ilustra la "clasificación burbuja" para una baraja reducida de 9 naipes (D corresponde a la carta Diez). La parte ordenada de la baraja va creciendo a cada pasada desde el extremo derecho. El 1 y el 2 debajo de la baraja indican los dos naipes que se están comparando en cada momento

```
Empezar clasificación
2 8 9 3 D 5 K 6 7  Empezar pasada 1
1 2
8 2 9 3 D 5 K 6 7
  1 2
8 9 2 3 D 5 K 6 7
    1 2
8 9 3 2 D 5 K 6 7
      1 2
8 9 3 D 2 5 K 6 7
        1 2
8 9 3 D 5 2 K 6 7
          1 2
8 9 3 D 5 K 2 6 7
            1 2
8 9 3 D 5 K 6 2 7
              1 2
8 9 3 D 5 K 6 7 2  Fin pasada 1
9 8 D 5 K 6 7 3 2  Fin pasada 2
9 D 8 K 6 7 5 3 2  Fin pasada 3
D 9 K 8 7 6 5 3 2  Fin pasada 4
D K 9 8 7 6 5 3 2  Fin pasada 5
K D 9 8 7 6 5 3 2  Fin pasada 6
Fin clasificación
```

Clasificación por inserción
Con la "clasificación por inserción", la parte ordenada de la lista va creciendo desde el extremo izquierdo. Los naipes se desplazan directamente hasta su posición correcta en la lista a medida que son revisados

```
Empezar clasificación
2 8 9 3 D 5 K 6 7
2 1
8 2 9 3 D 5 K 6 7
2   1
9 8 2 3 D 5 K 6 7
    2 1
9 8 3 2 D 5 K 6 7
2     1
D 9 8 3 2 5 K 6 7
      2   1
D 9 8 5 3 2 K 6 7
2         1
K D 9 8 5 3 2 6 7
          2     1
K D 9 8 6 5 3 2 7
            2       1
K D 9 8 7 6 5 3 2
Fin clasificación
```

La clasificación es una de las operaciones por ordenador que se utilizan más ampliamente, pero es una tarea para la cual estas máquinas, por sus propias normas de funcionamiento, son sumamente ineficaces. De acuerdo con la investigación operativa, se invierte en la clasificación entre el 30 y el 40 % del tiempo informático, y si sumáramos las otras tareas que van unidas a ella —intercalación de datos y búsqueda de ítems específicos—, es posible que la cifra se elevara a más del 50 %.

Es probable que los programadores destinen tanto tiempo a inventar algoritmos de clasificación (métodos generales para resolver problemas) como el que invierten los ordenadores en efectuar la clasificación real. Los métodos de clasificación avanzados son muy difíciles de analizar, pero nos resultará bastante fácil comprender los procedimientos más sencillos que utilizan los ordenadores para clasificar los datos, tomando como ejemplo la clasificación de una baraja de naipes.

Coloque sobre una mesa 13 naipes del mismo palo. Póngalos en línea, sin seguir ningún orden determinado, pero sin que ni el As ni el Dos queden en el extremo derecho de la línea. Los naipes se han de clasificar por orden descendente (Rey, Dama, Valet...As), comenzando por la izquierda. Para nosotros ésta es una tarea casi trivial y requiere tan poco esfuerzo mental que es difícil describir exactamente cómo lo haríamos. No obstante, si se especificara que sólo se puede mover una carta cada vez, que no se puede colocar un naipe encima de otro y que las cartas deben cubrir la menor superficie posible de la mesa, la tarea ya sería mucho menos trivial y resultaría difícil determinar un método eficaz. En esta analogía los naipes son datos, la superficie máxima cubierta corresponde a la memoria del ordenador requerida y usted es el programa. ¿Cómo resolvería el problema?

1) Coloque una moneda debajo del naipe situado más a la izquierda, para que actúe como indicador de posición y para que le recuerde en qué punto de la clasificación se encuentra. Compare el naipe marcado con el naipe situado a su derecha. ¿Están en orden descendente? Si no lo están, invierta sus posiciones, dejando la moneda en el mismo sitio, y obedeciendo la regla de mover sólo una carta cada vez y de no colocar un naipe encima de otro. Observe lo que debe hacer para invertirlos.

2) Cuando los dos naipes estén en orden, desplace la moneda un lugar hacia la derecha y repita el paso 1. Ahora se encuentra en un bucle que terminará cuando coloque la moneda en el lugar situado más a la derecha. Alcanzar esta posición se conoce como efectuar una "pasada" a través de las cartas.

3) Finalizada la primera pasada, eche una mirada a los naipes. El As, que es la carta más baja de la baraja, se ha abierto camino hasta el extremo derecho de la línea y, por lo tanto, se halla en el sitio correcto. Si hace otra pasada a través de los naipes, siguiendo el procedimiento detallado en los pasos 1 y 2, el Dos se desplazará hasta el lugar que le corresponde. Esto se repetirá, de una pasada a otra, hasta que toda la baraja se encuentre en orden descendente.

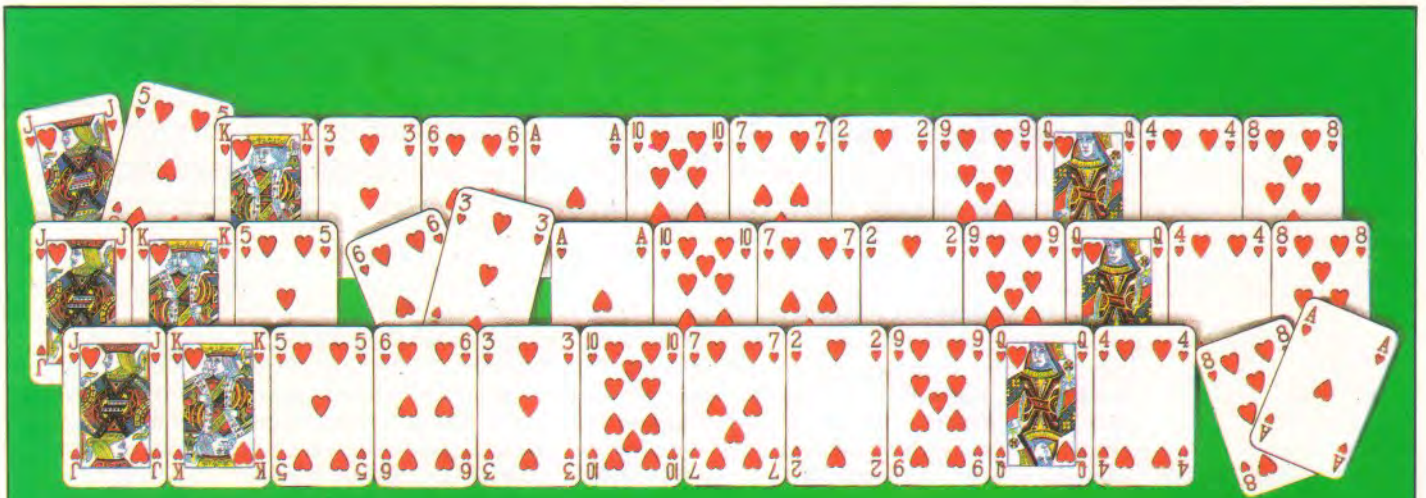
Es posible que haya observado diversos inconvenientes en este método. Es muy tedioso; no económico, ya que la sola acción de intercambiar las posiciones de dos naipes requiere efectuar tres operaciones diferentes; y, sobre todo, muchas de las comparaciones realizadas entre cartas distintas son innecesarias. Por ejemplo, al cabo de una pasada el As está en el lugar que le corresponde, de modo que no tiene ningún sentido desplazar la moneda a la posición 13 (donde, en todo caso, no existe comparación posible). En la segunda pasada, dado que el naipe situado a la derecha está en el lugar correcto, no hay necesidad de desplazar el señalador hasta la posición 12. En general, cada pasada terminará un lugar a la izquierda respecto al punto donde finalizó la pasada anterior.

Saber dónde se debe detener es otro problema. Un ordenador seguiría comparando naipes indefinidamente a menos que le dijéramos que parara. La única regla segura es detenerse después de una pasada sin inversiones. En otras palabras, si usted ha pasado a través de los datos sin tener que modificar su orden, se deduce que están ordenados.

El método de clasificación que hemos expuesto se denomina *clasificación burbuja*. Entre sus ventajas figuran: empleo de técnicas de programación simples, poca utilización de memoria extra y razonable eficacia con pequeñas cantidades de datos ordenados parcialmente. Éstos son los criterios en virtud de los cuales se debe juzgar un algoritmo de clasificación, si bien cuando los datos a clasificar son muchos, se ha de sacrificar la velocidad para economizar memoria, simplemente porque puede que la memoria del ordenador no tenga capacidad tanto para los datos en bruto como para una copia clasificada. Por este motivo, ignoraremos los algoritmos que requieran tomar datos de una matriz y desplazarlos hasta la posición clasificada de una segunda matriz. El segundo método de clasificación simple se base más directamente en la manera en que nosotros clasificaríamos las cartas.

1) Vuelva a colocar los naipes mezclados y coloque una moneda de una peseta debajo del segundo naipe empezando por la izquierda. Cualquiera que sea la carta bajo la cual esté colocada la moneda a cada pasada, la llamaremos "naipe de la peseta".

2) Empuje el naipe de la peseta fuera de la línea, dejando un lugar vacío, y coloque una moneda de cinco pesetas debajo del naipe situado inmediatamente a la izquierda. A este naipe lo llamaremos "naipe de las cinco pesetas".



Gran slam

La "clasificación burbuja" se puede ilustrar mediante una baraja de naipes que han de clasificarse de modo que el Rey (K) acabe estando en el extremo izquierdo y el As en el derecho. Primero se comparan los dos naipes situados más a la izquierda y, como se descubre que no están en orden, se invierten. Luego se comparan el segundo y el tercer naipe y, nuevamente, se invierten. En la quinta comparación, este método de clasificación ha recogido al As y en todas las comparaciones siguientes éste se va desplazando por inversión, de izquierda a derecha, hasta que al final de la primera "pasada" se ha abierto su camino, "burbujeando", hasta el extremo derecho. Sin embargo, podrían necesitarse 12 pasadas antes de que los naipes estuvieran en orden

Tony Lodge

3) Compare el naipe de la peseta con el naipe de las cinco pesetas. Si están en orden, vuelva a empujar a su lugar el naipe de la peseta y comience el paso 4. Si no están en orden, empuje entonces el naipe de las cinco pesetas hasta el lugar vacío y desplace la moneda de cinco pesetas un lugar hacia la izquierda para señalar un naipe de cinco pesetas nuevo. (Si la carta de las cinco pesetas está situada en el extremo izquierdo, esto no se aplica, de modo que coloque el naipe de la peseta en el lugar vacío y continúe por el paso 4.)

Compare este naipe de cinco pesetas con el naipe de la peseta (el desplazado). Ahora repita el paso 3 hasta hallar la posición correcta para el naipe de la peseta.

4) Desplace la peseta un puesto hacia la derecha y repita los pasos 2 y 3. Cuando ya no pueda desplazar la peseta hacia la derecha, los naipes estarán todos en orden.

Este método se denomina *clasificación por inserción* y se asemeja mucho a la forma en que solemos ordenar una baraja de naipes. Aunque es un poco más difícil de programar que una clasificación burbuja, es un método mucho más eficaz. Más adelante analizaremos algunos algoritmos más complejos para clasificación de datos.

```
9  REM ********************************
10 REM * ALGORITMOS DE CLASIFICACION *
11 REM ********************************
100 INPUT "CUANTOS ITEMS A CLASIFICAR";LT
150 IF LT <3 THEN LET LT = 3
200 LET LT = INT(LT)
250 DIM R(LT) ,C(LT)
300 LET Z = 0:LET Q = 0:LET P = 0
350 LET I = 1:LET D = 0: LET II = 2:LET TH = 2
400 INPUT "CUANTAS CONDICIONES";N
450 FOR CT = I TO N
500 GOSUB 4000
550 FOR SR = I TO TH
600 GOSUB 5000
650 PRINT:PRINT:PRINT:PRINT
700 PRINT "CONDICION #";CT+SR/10
750 INPUT "PULSE RETURN PARA COMENZAR
    CLASIFICACION";A$
800 PRINT "LA LISTA NO CLASIFICADA ES"
850 GOSUB 3000
900 ON SR GOSUB 6000,7000
950 PRINT "LA LISTA CLASIFICADA ES"
1000 GOSUB 3000
1050 NEXT SR
1100 NEXT CT
1150 END
2999 REM ***********************************
3000 REM * IMPRIMIR LA LISTA *
3001 REM ***********************************
3100 FOR K = I TO LT
3200 PRINT R(K);
3300 NEXT K
3400 PRINT
3500 RETURN
3999 REM ***********************************
4000 REM * GENERADOR ALEATORIO *
4001 REM ***********************************
4100 RANDOMIZE
4200 FOR K = I TO LT
4300 LET C(K) = INT(100*RND)
4400 NEXT K
4500 RETURN
4999 REM ***********************************
5000 REM * GENERADOR ALEATORIO *
5001 REM ***********************************
5100 FOR K = I TO LT
5200 LET R(K) = C(K)
5300 NEXT K
5400 PRINT:PRINT
5500 RETURN
5999 REM ************************
6000 REM * BURBUJA *
6001 REM ************************
6050 PRINT "CLASIFICACION BURBUJA - ADELANTE !!!!! "
6100 FOR P = LT - I TO I STEP - I
6150 LET F = -I
6200 FOR Q = I TO P
6250 LET Z = Q+I
6300 IF R(Q)<R(Z) THEN LET D = R(Q) :
     LET R(Q) = R(Z): LET R(Z) = D: LET F = Q
6350 NEXT Q
6400 IF F = -I THEN LET P = I
6450 NEXT P
6500 PRINT "CLASIFICACION BURBUJA - STOP !!!!! "
6550 RETURN
6999 REM ************************
7000 REM * INSERCION *
7001 REM ************************
7050 PRINT "CLASIFICACION POR INSERCION
     - ADELANTE !!!!! "
7100 FOR P = II TO LT
7200 LET D = R(P)
7300 FOR Q = P TO II STEP - I
7400 LET R(Q) = R(Q - I)
7500 IF D< = R(Q) THEN LET R(Q) = D:LET Q = II
7600 NEXT Q
7700 IF D> R(I) THEN LET R(I) = D
7800 NEXT P
7850 PRINT "CLASIFICACION POR INSERCION - STOP !!!!! "
7900 RETURN
```

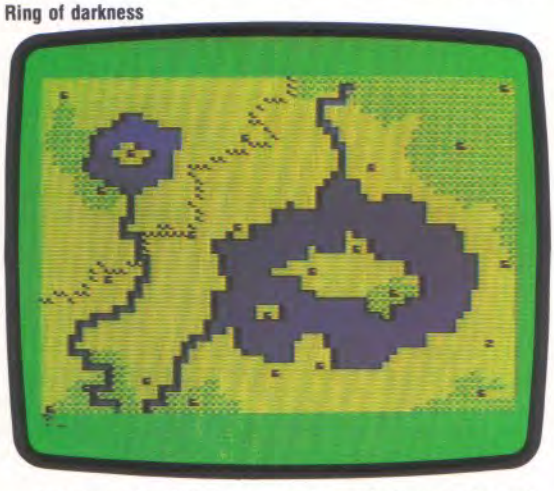
Clasificación a gran velocidad
Este programa en BASIC muestra la diferencia, en cuanto a eficacia, entre una "clasificación burbuja" y una "clasificación por inserción". El código se ha escrito teniendo siempre presente la velocidad, de modo que no hemos documentado la operación de las rutinas. El listado se debería poder ejecutar en la mayoría de las máquinas, pero lea en p. 215 los "complementos" referentes a ON...GOSUB y en p. 175 los relativos a RND y RANDOMIZE

Juegos de laberinto

A la gente siempre le han atraído los laberintos, y los juegos de este tipo creados por ordenador conservan toda su fascinación

Los laberintos siempre han sido una fuente de diversión y atracción tanto para los jóvenes como para los mayores, trátese de laberintos tan grandes como para perderse en ellos o tan pequeños que quepan en la palma de la mano. De hecho, el laberinto se ha convertido en la base de una gran variedad de juegos por ordenador, que abarcan desde una panorámica aérea muy sencilla y en dos dimensiones hasta laberintos tridimensionales sumamente complejos. Los de esta última clase en realidad simulan el aspecto de un laberinto visto desde su interior, de modo que el jugador se imagina que se halla dentro de uno verdadero. Para ayudar al jugador a que se haga una composición de lugar, o incluso para confundirlo aún más, algunas de estas imágenes de laberintos tridimensionales se combinan con vislumbres de una panorámica aérea de él.

Ring of darkness



Ring of darkness (Círculo de oscuridad)
A pesar de que este juego para el Dragon está catalogado como "de aventuras", contiene un laberinto tridimensional como uno de sus elementos principales. Fosos y escaleras le permiten al jugador moverse hacia arriba y hacia abajo

Siren city (Ciudad de sirenas)
Este juego para el Commodore 64 es un desarrollo del juego de "panorámica aérea" tradicional. Un coche de policía patrulla por una ciudad, con sus calles y sus edificios

Way out (Salida)
Con "Way out" se puede obtener en el Spectrum una realista imagen en tres dimensiones. Basta accionar minimamente la palanca de mando para que la panorámica cambie

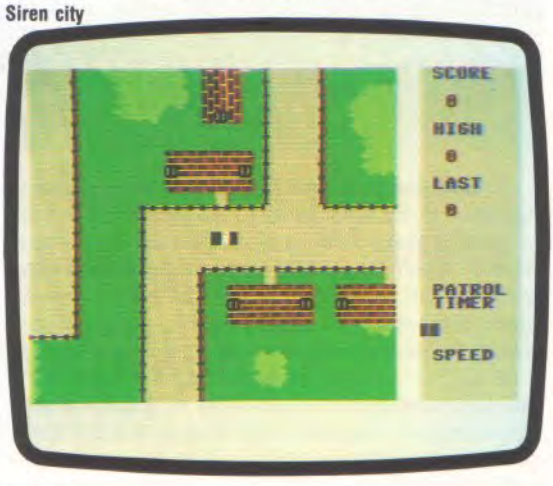
Así como los laberintos se han ido haciendo cada vez más sofisticados en cuanto a sus efectos visuales y sonoros, igualmente se les ha ido permitiendo a los programadores dejar volar su imaginación. Un jugador que quiera dar un relajante paseo a través de un laberinto habrá de evitar aquellos que esconden monstruos devoradores de hombres. Un ejemplo de esta clase de juegos es el *3D Glooper* (disponible para el Commodore 64), en el cual el jugador busca la salida del laberinto siguiendo unas baldosas especiales y puede ser atacado en cualquier momento por unos monstruos que ocupan toda la pantalla. La inminente aparición de una de estas criaturas se anuncia mediante un ruido cada vez más cercano de mandíbulas batientes.

Atic Atac (Spectrum) es un juego de persecución totalmente animado en el cual el jugador puede asumir el papel de cualquiera de tres personajes distintos. El laberinto es una serie de fosos, escalinatas y grandes mazmorras a varios niveles que se deben recorrer contra reloj. Las mazmorras están habitadas por diversas criaturas y objetos ilustrados gráficamente.

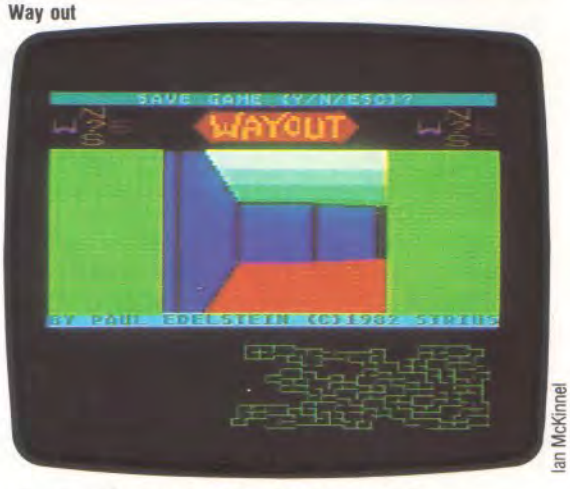
Un programa que simula con notable fidelidad lo que uno siente realmente al viajar a través de un laberinto es *Way out* (Salida). Las imágenes están en una perspectiva tridimensional, y cuando el jugador mueve la palanca de mando unas fracciones hacia la izquierda o hacia la derecha, la escena se desplaza proporcionalmente en esa dirección.

Consideremos ahora algunas de las técnicas de programación básicas para construir laberintos.

Siren city



Way out



Ian McKinnel

Construyendo laberintos

La forma corriente de almacenar la información relativa a un laberinto consiste en utilizar una matriz bidimensional: L$(FILA,COLUMNA), por ejemplo. Cada celda de la matriz definiría las características de esa celda del laberinto. Se podría utilizar, por ejemplo, una serie de cuatro caracteres para representar sur, oeste, norte y este. "Cero" indicaría la ausencia de pared y "uno" la presencia de una muralla. En consecuencia, si L$(5,6) contuviera la variable "1011", indicaría que la celda de la fila 5, columna 6, está limitada por paredes por el sur, el

norte y el este. Para ahorrar espacio de memoria, la matriz podría ser numérica en vez de alfanumérica, y el número de cuatro dígitos podría considerarse como un número binario. En nuestro ejemplo anterior, las celdas que incluyen norte, este y sur contendrían el número 11 (1011).

Todas las celdas comenzarían con cuatro paredes. Mediante la generación al azar de la entrada desde cualquier lugar del perímetro, se podría escoger aleatoriamente, de entre cualquiera de las tres celdas adyacentes, la siguiente celda. Una vez escogida ésta, la secuencia continúa, seleccionando al azar una celda de entre cualquiera de las tres adyacentes, haciendo caso omiso de la celda de la cual proviniera usted.

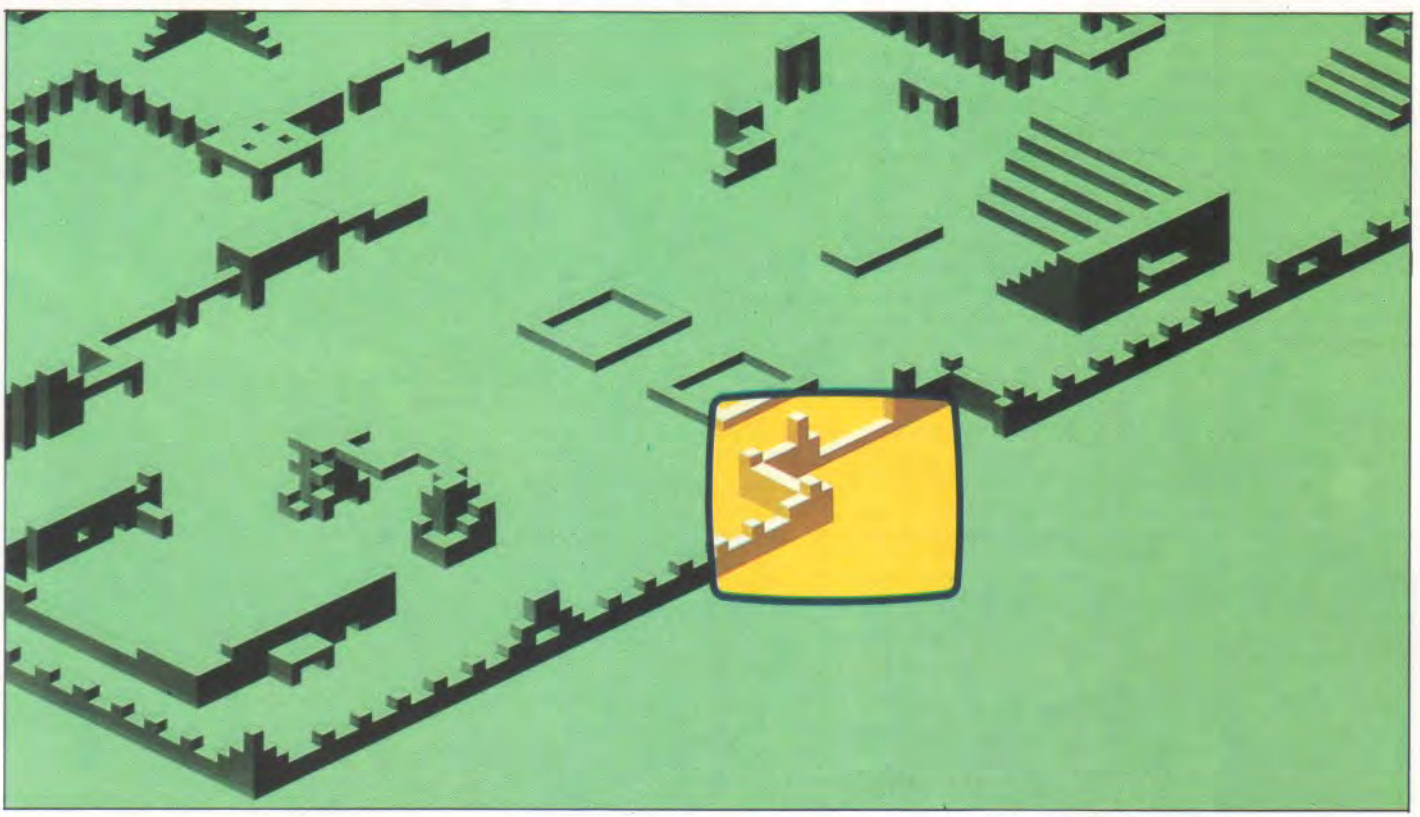
A medida que se escoge una nueva dirección, se quita la "pared" adecuada de la celda que se está por abandonar y de aquella a la que se va a entrar. Se deben efectuar verificaciones para asegurarse de que no se sale de los límites del laberinto (a menos que una celda determinada del perímetro sea el punto de salida) ni se crean circuitos cerrados (se debe poder acceder a todas las partes del laberinto desde cualquier punto).

Cuando uno se encuentra con una celda que posee menos de cuatro paredes (es decir, una celda que ya ha sido visitada), el programa debe escoger otra de las celdas adyacentes restantes. Si se han visitado todas las celdas adyacentes, el programa debe "retroceder un paso" hasta la celda visitada previamente y tomar una nueva bifurcación.

Existen 16 formas posibles de construir una celda: sin paredes, con paredes en todos los lados, con una sola pared (cuatro posibilidades), con paredes a los lados opuestos (dos posibilidades), con dos paredes adyacentes (cuatro posibilidades), y con tres paredes adyacentes (cuatro posibilidades).

Utilizando el número binario adecuado (0-15) para cada una de estas formas, se puede "hacer rotar" el número a izquierda o derecha para obtener la panorámica adecuada que observa el jugador. Por ejemplo, una pared norte se representaría por 2 (0010), una pared sur por 8 (1000), una pared este por 1 (0001) y una pared oeste por 4 (0100). Si el jugador que mirara al norte desde una celda con "una sola pared oeste" (4) girara para mirar al oeste, su panorámica estaría ahora limitada por una pared norte (porque mirar hacia adelante en una visualización tridimensional siempre es hacia el "norte"). Si el jugador se volviera hacia su izquierda (el oeste), mover el patrón de bits un lugar hacia la derecha proporcionaría la descripción que deseamos, o sea el binario pared oeste 0100 (decimal 4) se convierte en el binario 0010 (decimal 2: ¡una pared norte!). Los bits se mueven en la dirección opuesta al girar hacia la derecha, dos veces en una media vuelta. Es necesario, por supuesto, incluir un sistema para "recuperar" los bits que durante este proceso se pierden del extremo izquierdo o del derecho del medio byte, puesto que, de lo contrario, cada vez que el jugador girara dentro de una celda las características de identificación de ésta se modificarían. Una celda definida originalmente como 0011, por ejemplo (con paredes al norte y al este), ha de convertirse en 0110 si el jugador gira hacia la derecha, y en 1100 si da media vuelta.

En código de lenguaje máquina existen instrucciones especiales para hacer rotar los números binarios a derecha e izquierda. En BASIC, un número binario de cuatro bits expresado como decimal en la escala 0-15, se puede hacer girar hacia la izquierda multiplicando el número por dos y restando luego 15 si el resultado fuera mayor que 15. Para rotar hacia la derecha: dividir por dos si se trata de un número par, sumar 15 y dividir luego por dos en el caso de que fuera un número impar.



Ant attack (Ataque de las hormigas)
Cuando se ejecuta este juego en el Spectrum, la pantalla del ordenador actúa como una ventana abierta a un gran campo de juego parecido a un laberinto. A medida que se desarrolla el juego, la pantalla se desplaza, revelando detalles del escenario

Aquarius

Aunque fabricado por una empresa famosa por sus juguetes, el Aquarius es un ordenador eficaz y en toda regla

Con un procesador Z80 y su teclado estilo botón, el Mattel Aquarius es un microordenador en la línea del Spectrum. Sin embargo, en muchos aspectos es una máquina mucho más flexible, en gran medida porque sus diseñadores han sabido aprovechar muy bien su bus de ampliación incorporado.

A través de este bus se pueden conectar numerosos módulos de ampliación, desde pequeños paquetes de RAM de 4 Kbytes hasta un chasis de ampliación grande. Quizá el más útil de todos estos módulos sea el "chasis de ampliación pequeño", que posee dos ranuras para memoria extra o paquetes de programas, así como dos canales extra de sonido y dos controladores manuales. Conectando en una ranura un paquete de RAM de 16 Kbytes y, en la otra, un paquete de ROM patentado, como el Finplan, se puede obtener un sistema muy versátil.

La RAM de 4 Kbytes incorporada en la máquina no es muy generosa, pero con una ampliación de hasta 64 Kbytes, susceptible de lograr con el chasis de ampliación grande, se consigue una máquina tan potente como cualquier ordenador personal.

No obstante, el teclado y la visualización del Aquarius carecen de la calidad de las máquinas más grandes. El primero no posee barra espaciadora y la respuesta de las teclas no es ni muy rápida ni muy sensible, por lo cual no es apto para la escritura al tacto. La pantalla de 24 líneas de 40 caracteres, si bien es más grande que la de otros ordenadores, no es adecuada para su uso en pequeñas empresas.

La visualización dispone de 16 colores que se pueden utilizar tanto para el texto como para el fondo. Aunque carece de caracteres definibles por el usuario, posee 256 símbolos visualizables, que incluyen letras mayúsculas y minúsculas y una selección de símbolos gráficos. También se puede emplear como una pantalla de alta resolución de 320 × 192 pixels. La salida de la visualización es al televisor y no dispone de salida para monitor. La calidad es uniforme, con una notable tendencia hacia tonalidades azuladas y caracteres ligeramente borrosos, pero la imagen es continua y brillante, con una buena gama de color.

Miniampliador
Este dispositivo incorpora dos conexiones para cartucho, permitiendo conectar simultáneamente un cartucho de programas y un paquete de memoria. También incorpora los dos "controladores manuales" y tres canales de sonido adicionales

Chris Stevens

La impresora Aquarius
Esta impresora utiliza un mecanismo de impresión térmico y, en consecuencia, requiere un papel térmico especial. Puede imprimir a una velocidad de 80 caracteres por segundo, a través de una anchura total de 40 columnas

Esta máquina dispone de sonido, aunque carece de los sofisticados controles de envoltura y forma de onda que poseen otras. Lleva incorporado un BASIC Microsoft estándar, pero está previsto incluir un BASIC ampliado y un LOGO de Aquarius.

Uno de los accesorios más interesantes proyectados para el Aquarius es el sistema BSR X-10, que puede controlar una gama de aparatos domésticos. Este sistema permite controlar hasta 255 dispositivos eléctricos distintos en respuesta a las señales generadas por una unidad central. No se requiere ningún tendido de cables adicional, porque estas señales son en forma de impulsos enviados a través de la red eléctrica de la casa. Los impulsos no son lo suficientemente potentes como para que produzcan diferencias en la red de la corriente, pero un detector X-10 enchufado en cualquier toma de corriente puede captar el código y alterar la corriente suministrada a su aparato de acuerdo a la orden enviada.

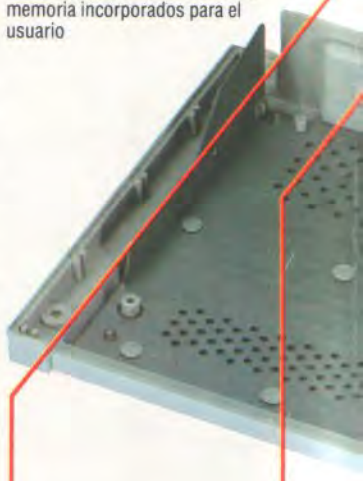
La unidad controlada la programa el Aquarius por ciclos semanales, y durante esta operación no se puede usar el ordenador para otros fines. Siempre y cuando el programa preestablecido sea satisfactorio, el ordenador queda libre para emplearlo normalmente en cualquier otro momento.

El teclado del Aquarius
El teclado es uno de los puntos más débiles del Aquarius. A pesar de que se lo promociona como un teclado QWERTY "estándar", apenas si merece esa calificación. No tiene barra espaciadora, posee una sola tecla SHIFT, RETURN está en una posición poco convencional y el interlineado no es exactamente igual al de una máquina de escribir

Conector para RF
Salida compatible para televisor; no dispone de salida para monitor

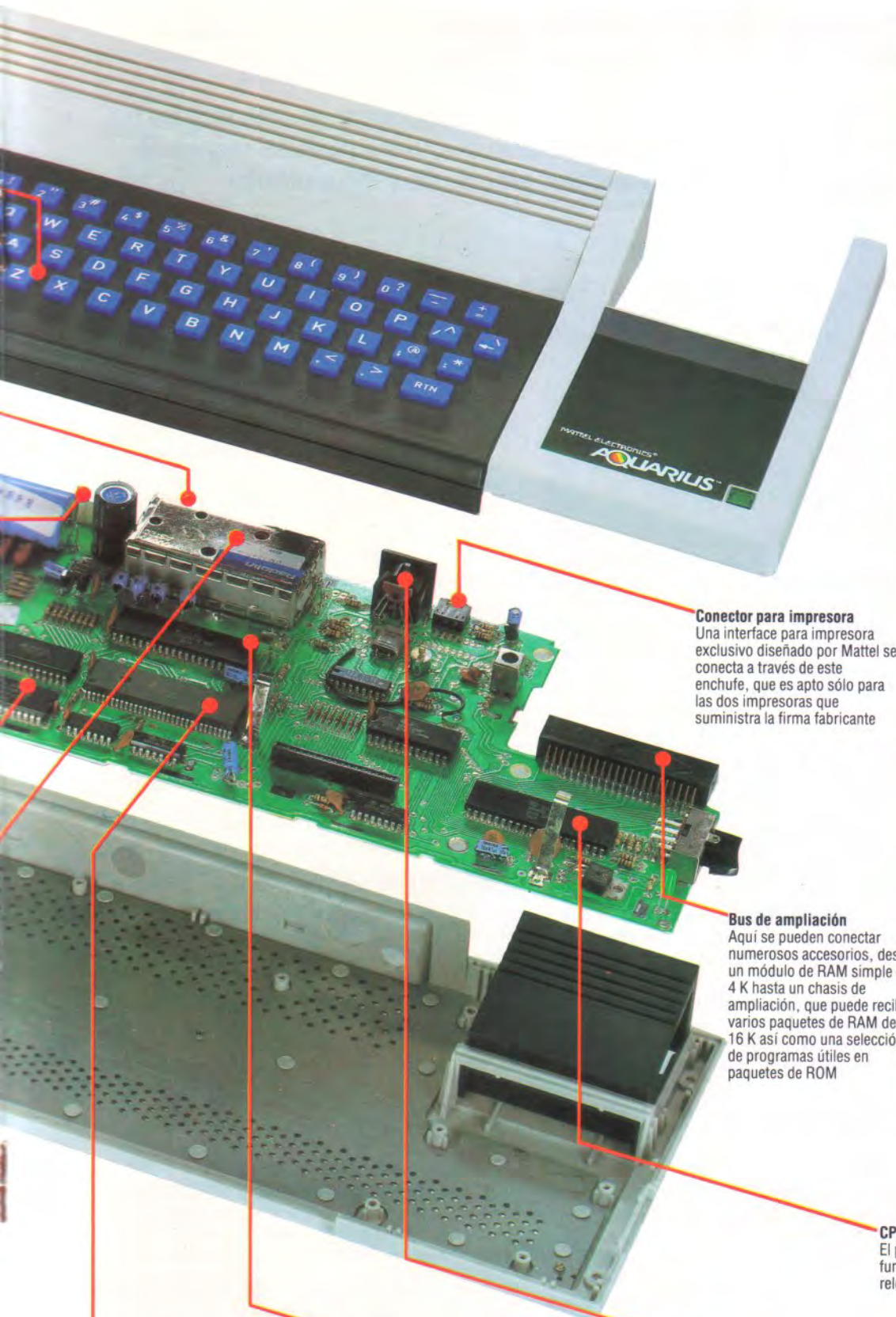
Conector para fuente de energía eléctrica
Aquí se aplica la energía proveniente de un pequeño transformador

RAM
Estos chips contienen los 4 K de memoria incorporados para el usuario



ROM
Estos chips retienen el BASIC Microsoft estándar de 8 K. El resto del espacio de la ROM está ocupado por las ampliaciones que se han agregado para manipular los gráficos y el sonido

Modulador
La señal de visualización en pantalla se convierte en una señal de TV estándar y aparece por el canal 36



AQUARIUS

DIMENSIONES
345 x 150 x 55 mm

VELOCIDAD DEL RELOJ
3,5 MHz

MEMORIA
10 Kbytes de ROM, más 4 Kbytes de RAM, ampliables a 64 Kbytes

VISUALIZACION EN VIDEO
24 líneas de 40 caracteres, 16 colores con determinación de fondo y de primer plano independiente; 256 caracteres predefinidos, pero ningún carácter definible por el usuario

INTERFACES
Cassette, impresora, bus de ampliación

LENGUAJE SUMINISTRADO
BASIC Microsoft

OTROS LENGUAJES DISPONIBLES
Mattel tiene previsto incluir un BASIC Microsoft ampliado y un LOGO de Aquarius. Vendrán en forma de paquete de ROM

VIENE CON
Manual de instalación y manual de BASIC, cable para TV

TECLADO
49 teclas estilo botón. El mando de borrado (RESET) está protegido para evitar que se le pulse accidentalmente

DOCUMENTACION
La documentación es especialmente adecuada para principiantes, con un juego muy útil de tarjetas que describen cada una de las funciones principales de la máquina y del BASIC incorporado. Carece de detalles técnicos, pero, en general, es apropiada para el mercado hacia el que está destinado el Aquarius

Conector para impresora
Una interface para impresora exclusivo diseñado por Mattel se conecta a través de este enchufe, que es apto sólo para las dos impresoras que suministra la firma fabricante

Bus de ampliación
Aquí se pueden conectar numerosos accesorios, desde un módulo de RAM simple de 4 K hasta un chasis de ampliación, que puede recibir varios paquetes de RAM de 16 K así como una selección de programas útiles en paquetes de ROM

CPU
El procesador es un Z80, que funciona a una frecuencia de reloj de 3,5 MHz

Controlador CRT
El diseño de la electrónica que controla la visualización de video es el aspecto más importante del diseño informático. Este chip controlador es más grande que el propio microprocesador

Chip de seguridad
Este chip de diseño a medida está pensado para hacer muy difícil que alguien, excepto el fabricante del ordenador, pueda producir cartuchos de programas para ejecutar con el Aquarius

Conector para cinta
La interface para cinta es un enchufe tipo DIN y posee conexiones para controlar el motor de la grabadora de cassette

Ramificación

A medida que se va desarrollando un programa, su estructura va tomando el aspecto de un árbol, que adquiere nuevas ramas al pasar cada una de las sucesivas etapas de refinamiento

En el capítulo anterior de nuestro curso de programación BASIC dimos una mirada a algunos de los problemas que implica la búsqueda a través de una lista para hallar un dato determinado (suponiendo que la lista ya estuviera clasificada por orden). Éste es un tema del que nos volveremos a ocupar con más detalle cuando llegue el momento de escribir rutinas de búsqueda. No obstante, mientras tanto desarrollaremos el tema de la programación *top-down* (de arriba abajo) para producir un código para las dos segundas partes del programa principal. Éste contiene cuatro llamadas a subrutinas o procedimientos:

```
PROGRAMA PRINCIPAL
EMPEZAR
  INICIALIZACION (procedimiento)
  PRESENTACION (procedimiento)
  ELECCION (procedimiento)
  EJECUCION (procedimiento)
FIN
```

El primer procedimiento, *INICIALIZACION*, implicará numerosas actividades bastante complicadas (establecer matrices, leer datos de ellas, realizar diversas verificaciones, etc.) y los detalles de su desarrollo los dejaremos para más adelante. Las dos partes siguientes del programa principal son las relativas a los procedimientos PRESENTACION y ELECCION. Para el desarrollo de estos procedimientos sugeriremos una metodología que ayude a evitar que se desorganicen y se confundan los muchos estratos involucrados en la realización de un programa *top-down*.

El problema que plantea el enfoque de un refinamiento de arriba abajo en el desarrollo de un programa, es que no se puede precisar el número de pasos necesarios antes de que estemos preparados para empezar la codificación en un lenguaje de alto nivel. Para los sistemas simples quizá podrían ser suficientes dos o tres pasos, pero los procedimientos más difíciles pueden requerir muchos antes de que el problema se haya analizado suficientemente como para permitir que se escriba el *código fuente* (así se denomina al programa en lenguaje de alto nivel). Esto significa que escribir un programa utilizando este método equivale a dibujar un árbol. A medida que van proliferando las "ramas" (es decir, a medida que los refinamientos se van volviendo más detallados), éstas van ocupando más lugar en la hoja. Finalmente, resulta imposible acomodar todo en una sola hoja y es en este punto donde resulta muy fácil perder la pista de lo que está sucediendo.

Una forma muy eficaz de organizar la documentación del programa consiste en numerar sistemáticamente las etapas de su desarrollo. Hemos empleado números romanos para indicar el nivel de refinamiento y números arábigos para señalar la subsección del programa. Después se utiliza una hoja separada de papel para cada uno de los niveles de refinamiento y las páginas para cada bloque o módulo de programa se pueden mantener juntas fácilmente. He aquí el sistema de numeración para nuestro programa:

```
I PROGRAMA PRINCIPAL
EMPEZAR
  1. INICIALIZACION
  2. PRESENTACION
  3. ELECCION
  4. EJECUCION
FIN
```

Como hemos mencionado más arriba, de momento estamos dejando de lado el desarrollo de INICIALIZACION, concentrándonos en desarrollar los procedimientos PRESENTACION y ELECCION.

```
II 2 (PRESENTACION)
EMPEZAR
  1. Visualizar mensaje de presentación
  2. LOOP (hasta que se pulse barra espaciadora)
     ENDLOOP
  3. Llamar *ELECCION*
FIN

III 2 (PRESENTACION) 1 (visualizar mensaje)
EMPEZAR
  1. Limpiar pantalla
  2. PRINT mensaje de presentación
FIN

III 2 (PRESENTACION) 2 (LOOP esperar barra
espaciadora)
EMPEZAR
  1. LOOP (hasta que se pulse barra espaciadora)
       IF se pulsa barra espaciadora
       THEN
     ENDLOOP
FIN

III 2 (PRESENTACION) 3 (llamar *ELECCION*)
EMPEZAR
  1. GOSUB *ELECCION*
FIN
```

En este punto debería estar claro que **III-2-1** y **III-2-3** están listos para ser codificados directamente en BASIC, pero que **III-2-2** requiere otra etapa de refinamiento:

```
IV 2 (PRESENTACION) 2 (LOOP)
EMPEZAR
  1. LOOP (hasta que se pulse barra espaciadora)
       IF INKEY$ no es espacio THEN continuar
     ENDLOOP
FIN
```

Nos encontramos ahora en el punto donde con muy poco refinamiento mejor se puede abordar toda la codificación en BASIC para el procedimiento PRESENTACION:

IV 2 (PRESENTACION) 1 (visualizar mensaje) CODIGO BASIC

```
REM SUBRUTINA *PRESENTACION*
PRINT
PRINT
PRINT
PRINT
PRINT TAB(11);"*BIEN VENIDO A LA*"
PRINT TAB(9);"*AGENDA COMPUTERIZADA*"
PRINT TAB(12);"*DE MI COMPUTER*"
PRINT
PRINT TAB(0);"(PULSE BARRA ESPACIADORA PARA
  CONTINUAR)"
```

V 2 (PRESENTACION) 2 (LOOP esperar barra espaciadora) CODIGO BASIC

```
LET L = 0
FOR L = 1 TO 1
IF INKEY$ <>" " THEN LENT L = 0
NEXT L
```

IV 2 (PRESENTACION) 3 (llamar *ELECCION*) CODIGO BASIC

```
GOSUB *ELECCION*
RETURN
```

Observe que ahora hemos empezado a inicializar variables en las diversas rutinas que escribimos, utilizando sentencias en forma de LET I = 0. En rigor, esto no es necesario en algunas de las circunstancias en las que las hemos utilizado. No obstante, sería conveniente que usted se acostumbrara a ellas si puede recordarlas y si dispone de suficiente espacio de RAM. Y ello se debe a tres razones: primero, porque tener una lista de sentencias LET al comienzo de cualquier rutina sirve como un recordatorio útil de las variables locales que utiliza esa rutina. Segundo, porque puede que no esté seguro de lo que quedó en una variable desde la última vez que se la utilizó en una rutina (aunque esto no siempre es importante). Tercero, tal como le explicaremos cuando esté más avanzado el curso, porque colocar sentencias en forma de LET I = 0 en el orden correcto puede acelerar la ejecución de un programa.

Hemos alterado la forma en que utilizamos el bucle FOR...NEXT para simular una estructura DO...WHILE o REPEAT...UNTIL, explicadas en anteriores capítulos del curso. En vez de emplear FOR I = 0 TO 1 o FOR I = 0 TO 1 STEP 0, ahora utilizamos FOR I = 1 TO 1. Esto funcionará correctamente en todos los ordenadores personales con los que tratamos habitualmente, mientras que los otros procedimientos harían necesario "Complementos al BASIC" para varias máquinas. FOR I = 1 TO 1...NEXT I ejecutará el bucle sólo una vez. Sin embargo, si en algún punto del cuerpo del bucle I se estableciera en 0, entonces el bucle se ejecutaría otra vez, y así sucesivamente. Podemos insertar una sentencia LET I = 0 como resultado del fracaso de una condición de salida, o bien establecer I en 0 inmediatamente después de la sentencia FOR, y establecerlo en 1 si tuviera éxito la condición de salida. De modo, entonces, que los dos bucles siguientes alcanzan el mismo objetivo:

```
FOR I = 1 TO 1
IF INKEY$ <> " " THEN LET I = 0
NEXT I
```

o

```
FOR I = 1 TO 1
LET I = 0
IF INKEY$ = " " THEN LET I = 1
NEXT I
```

El código BASIC que acabamos de producir es todo cuanto se necesita para el bloque de PRESENTACION completo del programa principal. No hemos colocado números de línea porque realmente no podemos hacerlo hasta que todos los módulos del programa estén listos para la codificación final. Por ejemplo, en esta etapa no sabemos cuáles son los números de línea adecuados para las órdenes GOSUB. Si usted deseara comprobar el módulo en esta etapa, sería necesario crear algunas entradas ficticias y subrutinas ficticias. Algunos puntos de este fragmento de programa que es necesario señalar son el empleo de la función TAB y las sentencias para "limpiar la pantalla". TAB hace que el cursor se mueva a lo largo de la línea según el número (el "argumento") especificado entre paréntesis. Los números que hemos dado harán que el mensaje se imprima exactamente en el centro de una pantalla de 40 caracteres. Si su visualización fuera menos ancha que ésta (por ejemplo, el Spectrum visualiza 32 caracteres por línea) o más ancha (los ordenadores mayores normalmente visualizan 80 caracteres), será necesario modificar, consecuentemente, estos argumentos TAB. En muchas versiones de BASIC la instrucción para limpiar la pantalla es CLS, pero la versión de BASIC Microsoft utilizada para desarrollar este programa no la admite. En cambio, hemos empleado PRINT CHR$(12), dado que nuestra máquina utiliza ASCII 12 como su carácter no imprimible para "limpiar la pantalla" (otras suelen utilizar ASCII 24 para realizar la misma función).

```
10 REM PROGRAMA PRINCIPAL FICTICIO
20 PRINT CHR$(12)
30 GOSUB 100
40 END
100 REM SUBRUTINA *PRESENTACION*
110 PRINT
120 PRINT
130 PRINT
140 PRINT
150 PRINT TAB(11);"*BIEN VENIDO A LA*"
160 PRINT TAB(9);"*AGENDA COMPUTERIZADA*"
170 PRINT TAB(12);"*DE MI COMPUTER*"
180 PRINT
190 PRINT TAB(0);"(PULSE BARRA ESPACIADORA
    PARA CONTINUAR)"
195 LET L = 0
200 FOR L = 1 TO 1
210 IF INKEY$ <> " " THEN LET L = 0
220 NEXT L
230 PRINT CHR$(12)
240 GOSUB 1000
250 RETURN
1000 REM SUBRUTINA FICTICIA
1010 PRINT "SUBRUTINA FICTICIA"
1020 RETURN
```

Ahora utilizaremos exactamente el mismo enfoque para refinar el procedimiento ELECCION.

II 3 (ELECCION)
EMPEZAR
1. PRINT menú
2. INPUT OPCION
3. Llamar subrutina seleccionada

FIN

III 3 (ELECCION) 1 (PRINT menú)
EMPEZAR
1. Limpiar pantalla
2. PRINT menú y aviso

FIN

III 3 (ELECCION) 2 (INPUT OPCION)
EMPEZAR
1. INPUT OPCION
2. Verificar que OPCION esté dentro de la escala

FIN

III 3 (ELECCION) 3 (llamar OPCION)
EMPEZAR
1. CASO DE OPCION
   FIN DEL CASO

FIN

Ahora **III-3-1 (PRINT menú)** se puede codificar en BASIC:

IV 3 (ELECCION) 1 (PRINT menú) CODIGO BASIC

```
REM LIMPIAR PANTALLA
PRINT CHR$(12):REM 0 'CLS'
PRINT
PRINT
PRINT
PRINT
PRINT "1.HALLAR REGISTRO (DE NOMBRE)"
PRINT "2.HALLAR NOMBRES (DE NOMBRE
      INCOMPLETO)"
PRINT "3.HALLAR REGISTRO (DE CIUDAD)"
PRINT "4.HALLAR REGISTRO (DE INICIALES)"
PRINT "5.LISTAR TODOS LOS REGISTROS"
PRINT "6.AGREGAR REGISTRO NUEVO"
PRINT "7.MODIFICAR REGISTRO"
PRINT "8.BORRAR REGISTRO"
PRINT "9.SALIDA Y GUARDAR"
```

Sin embargo, **III-3-2 (INPUT OPCION)** y **III-3-3 (llamar OPCION)** requieren todavía más refinamiento. Analicemos primero el siguiente nivel de desarrollo de **III-3-2**.

Asignarle un valor numérico a la variable OPCION es muy sencillo: después del aviso, lo hará una orden de INPUT OPCION. No obstante, sólo hay nueve opciones posibles. ¿Qué sucedería si por error diéramos entrada a 0 o 99? Dado que la OPCION que hagamos determinará a cuál de las partes del programa se llamará a continuación, deseamos asegurarnos de que no se produzcan errores, de modo que necesitamos llevar a cabo un procedimiento de "verificación de escala". Éste consiste en una pequeña rutina de verificación para ver si el número al que se ha dado entrada se halla dentro de la escala aceptada, antes de permitir que el programa continúe. He aquí una rutina de muestra diseñada para interrumpir una entrada errónea.

RUTINA DE VERIFICACION DE ESCALA

```
1 REM RUTINA
10 LET L = 0
20 FOR L = 1 TO 1
30 INPUT "DE ENTRADA A 1-9";OPCION
40 IF OPCION <1 THEN LET L = 0
50 IF OPCION >9 THEN LET L = 0
60 NEXT L
70 PRINT "LA OPCION ES";OPCION
80 END
```

Muchas versiones de BASIC pueden simplificar esta rutina mediante la inclusión en la condición de un operador de Boole como éste:

```
10 LET L = 0
20 FOR L = 1 TO 1
30 INPUT "DE ENTRADA A 1-9";OPCION
40 IF OPCION <1 OR OPCION>9 THEN LET L = 0
50 NEXT L
60 PRINT "LA OPCION ES";OPCION
70 END
```

Estas rutinas ilustran asimismo otro punto acerca de la sentencia INPUT. Esta sentencia hace que el programa se detenga y espere una entrada desde el teclado. El BASIC no sabe cuál es el número completo al que se ha dado entrada hasta que se pulsa la tecla RETURN, de modo que el usuario también tendrá que acordarse de pulsar RETURN después de dar entrada al número.

Un enfoque más "amable para el usuario" sería hacer que el programa continuara apenas se diera entrada a un número válido. Esto es posible gracias a la utilización de la función INKEY$. En este caso, el BASIC lee un carácter del teclado cada vez que se encuentra con INKEY$. Sin embargo, el programa no se detiene y continuará sin ninguna pausa por la parte siguiente. En consecuencia, es frecuente emplear INKEY$ dentro de un bucle. El bucle para comprobar si se está pulsando una tecla puede ser IF INKEY$ = "" THEN...; en otras palabras, si no se está pulsando ninguna tecla, vuelva y verifíquelo de nuevo. A nuestros fines, un bucle adecuado sería:

```
LET I = 0
FOR I = 1 TO 1
LET A$ = INKEY$
IF A$ = "" THEN LET I = 0
NEXT I
```

El único inconveniente que comporta la utilización de INKEY$ es que devuelve un carácter del teclado en vez de uno numérico. Cuando hay un menú de ELECCION, en el que se realiza una selección entre varias opciones (una ramificación multicondicionada), en BASIC es más fácil utilizar números que caracteres. Aquí es donde entran en juego las funciones NUM o VAL de BASIC. Éstas convierten a los números de las series de caracteres en números "reales" (es decir, valores numéricos y no códigos ASCII que representen numerales). Se pueden utilizar de la siguiente manera:

```
LET N = VAL(A$) o LET N = NUM(A$)
```

Utilizando las funciones NUM o VAL podemos hacer que, empleando INKEY$, el programa convierta las entradas en variables numéricas. Este procedimiento elimina la necesidad de emplear la tecla RETURN después de haber pulsado la tecla numérica. No obstante, es recomendable la verificación fuera de escala.

El siguiente fragmento de programa incluye dos bucles, uno anidado dentro del otro. El bucle inte-

rior espera a que se pulse una tecla; el bucle exterior convierte la variable en un número y verifica que esté dentro de la escala:

```
FOR L = 1 TO 1
  PRINT "DE ENTRADA A OPCION (1-9)"
    FOR I = 1 TO 1
      LET A$ = INKEY$
      IF A$ = "" THEN LET I = 0
    NEXT I
  LET OPCION = VAL(A$)
    IF OPCION <1 THEN LET L = 0
    IF OPCION >9 THEN LET L = 0
NEXT L
```

Por último, reproducimos un programa completo en BASIC para el módulo *ELECCIÓN*, incluyendo subrutinas y entradas ficticias con fines de prueba. Debemos señalar, nuevamente, que los números de línea sólo se han incluido como prueba y habrán de ser sustituidos cuando se elabore el programa final.

```
10 PRINT CHR$(12)
20 PRINT "SELECCIONE UNA DE LAS
   SIGUIENTES OPCIONES"
30 PRINT
40 PRINT
50 PRINT
60 PRINT "1. HALLAR REGISTRO (DE NOMBRE)"
70 PRINT "2. HALLAR NOMBRES (DE NOMBRE
   INCOMPLETO)"
80 PRINT "3. HALLAR REGISTRO (DE CIUDAD)"
90 PRINT "4. HALLAR REGISTRO (DE INICIALES)"
100 PRINT "5. LISTAR TODOS LOS REGISTROS"
110 PRINT "6. AGREGAR REGISTRO NUEVO"
120 PRINT "7. MODIFICAR REGISTRO"
130 PRINT "8. BORRAR REGISTRO"
140 PRINT "9. SALIDA Y GUARDAR"
150 PRINT
160 PRINT
170 LET L = 0
180 LET I = 0
190 FOR L = 1 TO 1
200 PRINT "DE ENTRADA A OPCION (1-9)"
210 FOR I = 1 TO 1
220 LET A$ = INKEY$
230 IF A$ = "" THEN LET I = 0
240 NEXT I
250 LET OPCION = VAL(A$)
260 IF OPCION <1 THEN LET L = 0
270 IF OPCION >9 THEN LET L = 0
280 NEXT L
290 ON OPCION GOSUB 310,330,350,370,390,410,
   430,450,470
300 END
310 PRINT "SUBRUTINA FICTICIA 1"
320 RETURN
330 PRINT "SUBRUTINA FICTICIA 2"
340 RETURN
350 PRINT "SUBRUTINA FICTICIA 3"
360 RETURN
370 PRINT "SUBRUTINA FICTICIA 4"
380 RETURN
390 PRINT "SUBRUTINA FICTICIA 5"
400 RETURN
410 PRINT "SUBRUTINA FICTICIA 6"
420 RETURN
430 PRINT "SUBRUTINA FICTICIA 7"
440 RETURN
450 PRINT "SUBRUTINA FICTICIA 8"
460 RETURN
470 PRINT "SUBRUTINA FICTICIA 9"
480 RETURN
```

En el próximo capítulo analizaremos las estructuras de archivo y empezaremos a refinar el procedimiento INICIALIZACION.

Complementos al BASIC

SPECTRUM

En el programa principal ficticio, y de principio a fin, sustituir PRINT CHR$(12) por CLS y END por STOP.

RUTINA DE VERIFICACION DE ESCALA

```
1 REM RUTINA
10 LET L = 0
20 FOR L = 1 TO 1
30 INPUT "ENTRADA A 1-9";OPCION
40 IF OPCION <1 THEN LET L = 0
50 IF OPCION >9 THEN LET L = 0
60 NEXT L
70 PRINT "LA OPCION ERA";OPCION
80 STOP
```

LISTADO FINAL

```
10 CLS
```

después copiar la lista del texto principal hasta:

```
240 NEXT I
250 LET OPCION = CODE A$ - 48
260 IF OPCION <1 THEN LET L = 0
270 IF OPCION >9 THEN LET L = 0
280 NEXT L
290 GOSUB (OPCION*20 + 290)
300 STOP
```

luego copiar la lista principal desde la línea 310 hasta la 480.

TAB

Algunas versiones del Oric-1 no obedecen a la orden TAB, aun cuando está incluida en el BASIC del Oric-1; en este caso, insertar esta línea al principio del programa:

```
5 LET S$ = ""
```

En esta línea, entre las comillas debería haber tantos espacios como caracteres haya en una línea de pantalla completa (40, para un Oric-1). Después, siempre que el programa diga TAB(11), reemplácelo por LEFT$(S$,11), copiando el número de la sentencia TAB en la función LEFT$( ).

CHR$(12)

En el Oric-1, el Dragon 32, el Lynx y el BBC Micro, sustituir PRINT CHR$(12) por CLS. En el Commodore 64 y en el Vic-20, para reemplazar CHR$(12) consulte el manual.

ON.. GOSUB

No está disponible en el Lynx, pero se puede sustituir por la línea 290 del listado final que hemos dado arriba para el Spectrum

VARIABLES

Ver "Complementos al BASIC", p. 257.

INKEY$

Ver "Complementos al BASIC", p. 175. Los usuarios de un Commodore han de sustituir LET A$ = INKEY$ por GET A$, e IF INKEY$ = "" THEN por: GET A$:IF A$ = "" THEN

Un útil "ratón"

Los diseñadores desean reemplazar el teclado por un dispositivo de uso más sencillo. Éste podría ser el "ratón"

Hasta no hace mucho, la única forma de acceder a los ordenadores era a través de unas enormes máquinas de escribir electromecánicas llamadas "teletipos". Éstos eran dispositivos ruidosos, difíciles de manejar y nada fiables, que desde entonces han ido siendo sustituidos por la silenciosa y veloz VDU (*Visual Display Unit:* unidad de representación visual) con teclado. La VDU eliminó muchos de los problemas relacionados con los teletipos, uno de los cuales era la enorme cantidad de papel consumido en las cintas perforadas a medida que se iba digitando la información. No obstante, tanto el terminal mecánico como la VDU más teclado están limitados por su formato carácter a carácter, línea a línea. El usuario no puede desplazarse rápidamente por la pantalla (seleccionar ítems de un menú por aquí, modificar un dato por allá, o cambiar archivos y programas) sin encontrarse con las limitaciones del formato del cursor tecleado. La liberación del teclado se consigue al emplear terminales para gráficos o al practicar juegos por ordenador con mandos de bola y palancas de mando; pero ¿qué utilidad pueden reportar estos dispositivos si se quiere destinar el ordenador a otros fines?

La mayoría de los ordenadores personales que existen en el mercado están equipados con mandos para el cursor en cuatro direcciones, que se pueden desplazar a través de un listado de programa o del texto de un documento hasta la posición donde se necesita hacer una corrección. Pero el cursor sólo se puede mover por pasos de una línea o un carácter; el usuario no lo puede desplazar directamente hasta su destino. Si el cursor de texto fuera susceptible de moverse como un cursor de gráficos, que se puede manipular con entera libertad bajo el control de un mando de bola o de una palanca de mando, los datos se podrían desplazar de una manera considerablemente más rápida.

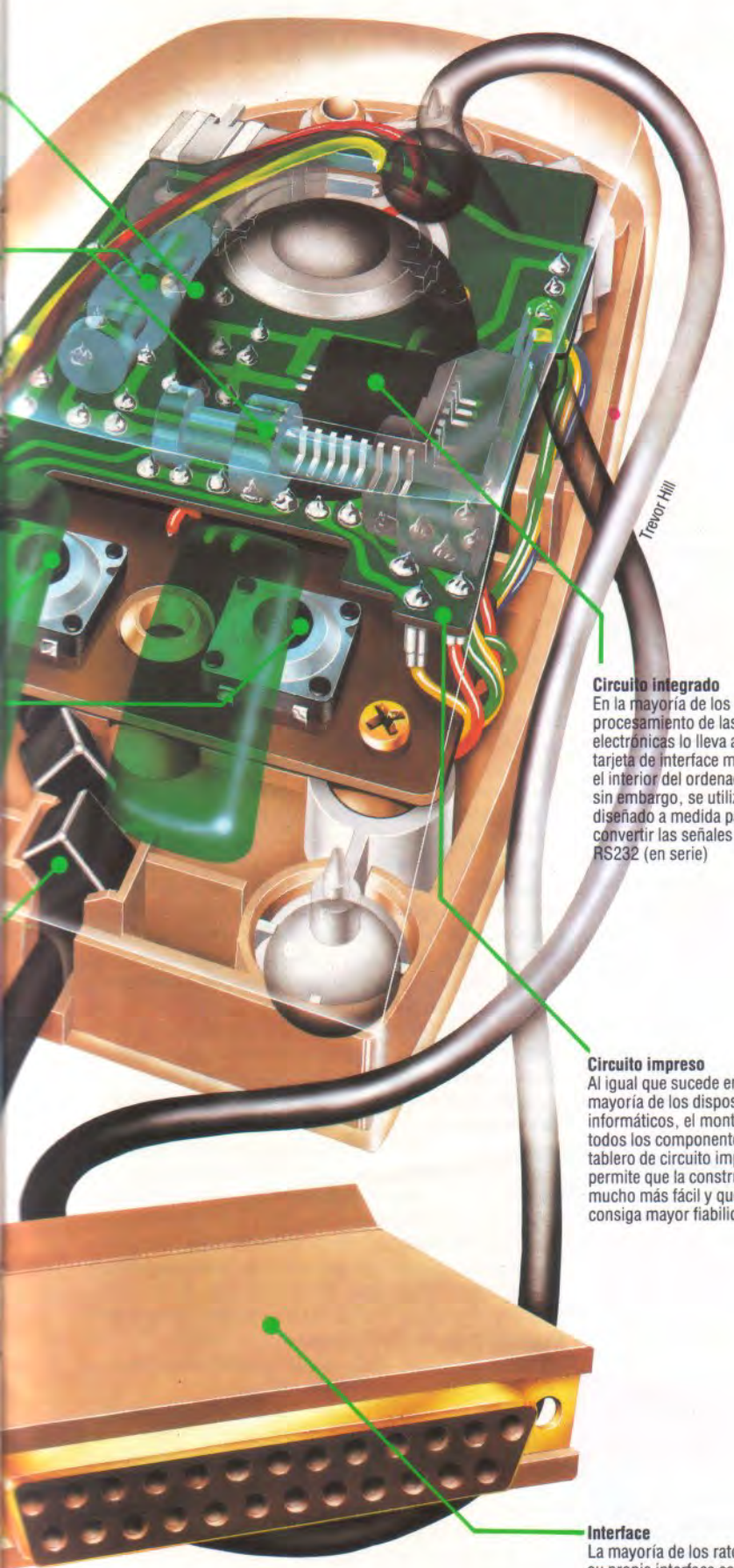
"Tres ratones ciegos"
Muchos de los microordenadores para gestión empresarial más recientes incorporan un ratón como estándar, y algunas empresas ofrecen unidades como accesorios para máquinas ya existentes. La mayoría de los ratones opera mediante una bola rotatoria e incorpora uno, dos o tres botones de "SELECT"

Ian McKinnel. Cortesía de Microsoft, Apple y Xerox

En el Stanford Research Institute de California se estudió por primera vez una solución a este problema, en los años sesenta; y el primer "ratón" (tal como se bautizó al nuevo tipo de controlador que se desarrolló) se patentó en 1970. Al dispositivo se lo denominó *ratón* debido a su aspecto: un ratón es lo suficientemente pequeño como para caber en la palma de la mano; tiene un "rabo" (el cable), y los primeros dispositivos solían tener dos "orejas" (los botones de control). No se utilizan mandos de bola ni palancas de mando convencionales porque no se requiere la precisión que éstos proporcionan para colocar el cursor en la posición deseada.

El ratón funciona detectando su movimiento a través de cualquier superficie plana en las direcciones arriba-abajo e izquierda-derecha, así como en las combinaciones de ambas. Estos movimientos se convierten directamente en desplazamientos del cursor (o señalador, como también se lo suele llamar) en la pantalla. Existen dos métodos para generar las señales eléctricas del movimiento del ratón. En ambos procedimientos, en la cara inferior del ratón hay una bola grande que se apoya en la superficie sobre la cual se está moviendo.

La rotación del punto de apoyo de la bola del ratón se transfiere a unos cojinetes cilíndricos internos. En uno de los sistemas, los extremos de estos cilindros están provistos de ruedas de código que poseen pistas alternadas de material conductor y no conductor. Los impulsos recibidos los cuenta el



Circuito integrado
En la mayoría de los ratones, el procesamiento de las señales electrónicas lo lleva a cabo una tarjeta de interface montada en el interior del ordenador. Aquí, sin embargo, se utiliza un chip diseñado a medida para convertir las señales a la forma RS232 (en serie)

Circuito impreso
Al igual que sucede en la mayoría de los dispositivos informáticos, el montaje de todos los componentes sobre un tablero de circuito impreso permite que la construcción sea mucho más fácil y que se consiga mayor fiabilidad

Interface
La mayoría de los ratones utiliza su propia interface especial ("ratonera"), pero ésta se puede enchufar en cualquier conexión RS232, usando el conector estándar de 25 canales

software operativo del ratón y le permiten dar una lectura de la posición del cursor en la pantalla. En el otro sistema, los cojinetes llevan acoplados dos discos acanalados. A los discos se les dirige continuamente una luz y, al otro lado de ellos, una célula fotoeléctrica detecta ópticamente el haz. Los impulsos de luz que pasan a través de las ranuras se convierten luego en señales eléctricas, que se tratan de la misma manera que en el sistema mecánico.

Existen, asimismo, otros sistemas. Hay uno, por ejemplo, en que el ratón se utiliza en conjunción con un relleno especial cubierto por un patrón de puntos. Una luz en el interior del cuerpo del ratón ilumina la superficie de relleno cubierta por el ratón y este patrón lo detecta un chip procesador óptico especial. Cualquier movimiento del ratón hará que cambie el patrón que detecta el chip, que puede calcular al instante hasta dónde se ha movido el dispositivo y en qué dirección. Este sistema ofrece la ventaja de que no posee partes móviles, pero es mucho más caro que los otros.

Una vez que se ha desplazado el cursor hasta el lugar de la pantalla requerido, se puede dar entrada a su posición en el ordenador pulsando una de las "orejas" (botones) del ratón. El número de botones de que dispone el ratón varía de un fabricante a otro. Algunos sistemas utilizan tres; Microsoft ha optado por incorporarle dos, mientras que el ratón del Lisa de Apple posee sólo uno. Los botones también se pueden emplear para seleccionar ítems de un menú (programas como el *MultiTool Word* de Microsoft ofrecen esta facilidad) y para confiarle al ratón el control del movimiento del cursor normal. Estos dispositivos se pueden utilizar con software altamente sofisticado, como el que se proporciona con el Lisa. Aquí el botón se pulsa una vez para seleccionar un "icono" (véase p. 262) de un menú en pantalla, y dos veces para posibilitar esa aplicación determinada.

La ventaja principal de todos los ratones, y del software que se ha producido para complementarlos, es que los pueden utilizar todas aquellas personas que no poseen experiencia con teclado. En vez de tener que digitar el nombre de un programa o de pulsar ciertas letras o números para seleccionar una función, el usuario simplemente mueve el ratón de modo que el cursor de la pantalla señale la aplicación o el cursor de acción que se requiera, y pulsa un botón para activarlo.

Lamentablemente, este nuevo dispositivo no elimina por completo la necesidad del teclado (aún se debe alimentar el ordenador con textos y números nuevos), pero simplifica en gran medida la manipulación de la información. Las pruebas que llevó a cabo la Apple durante el desarrollo del Lisa demostraron que un usuario que no tuviera ninguna experiencia con un ordenador podía aprender a trabajar con el software activado por ratón del Lisa en apenas 15 minutos. Ejecutado en un sistema convencional, familiarizarse con un software similar lleva aproximadamente 20 horas, básicamente debido a los problemas que entraña aprender a emplear el teclado y a la necesidad de aprender órdenes largas y complicadas. Los ratones electrónicos muy pronto serán un componente integral de los ordenadores personales. Son eficaces y fáciles de utilizar, y a las personas poco emprendedoras no las atemorizan tanto como la simple vista de un teclado QWERTY tradicional.

Labor detectivesca

Cuando se pasa información de un ordenador a otro, se corre el riesgo de que ciertos datos se alteren. Los códigos Hamming pueden detectar y corregir estos errores

Todos hemos oído alguna historia acerca de garrafales errores cometidos por ordenadores, como, por ejemplo, enviarle por correo 500 ejemplares del folleto de una empresa a una misma persona. La verdad, por supuesto, es que la máquina no tiene culpa alguna: la equivocación se origina en un fallo humano, quizá tan sencillo como un error de digitación. El ordenador sirve tan sólo para magnificar el problema.

Algunas veces, no obstante, los ordenadores cometen fallos no imputables a la intervención humana que por lo general se manifiestan en forma de "errores de bits". Un error de bits se produce cuando se transpone un bit simple de una sección de datos de 1 a 0 o viceversa. Un error de bits puede surgir cuando falla un componente del hardware, como un chip de RAM. Por ese motivo muchos ordenadores personales se someten a un software de "diagnóstico" para verificación de errores cada vez que se encienden.

Sin embargo, la mayoría de los errores de bits son "errores de *soft*": los bits "se dan vuelta" aun cuando toda la RAM haya pasado la prueba de diagnóstico. Los ordenadores personales están diseñados para trabajar en interiores, pero durante una intensa ola de calor en verano, es muy posible que la temperatura supere la escala térmica operativa de los componentes. Es poco probable que el daño sea de tipo permanente, pero los errores de bits podrían ocasionar que un carácter de la pantalla cambiara súbitamente de una "A" a una "B", por ejemplo, o, en el caso de que el bit formara parte de un indicador importante, podría "romper" el programa, haciendo necesario restaurarlo.

Los errores de bits también pueden surgir en períodos de intensa actividad de las manchas solares, cuando partículas subatómicas pueden penetrar en la atmósfera e interferir el flujo de electrones de un circuito en miniatura. En aplicaciones tales como sistemas militares, control industrial, experimentos científicos o movimiento bancario internacional, los errores pueden tener consecuencias desastrosas, de modo que para detectarlos se han adoptado diversos procedimientos.

El más sencillo de estos métodos es el del control de paridad (véase p. 253). Un procedimiento alternativo es la suma de control, que se utiliza mucho al escribir datos en cinta magnética o en disco. Comúnmente, los datos se manipulan en bloques de 128 bytes, de los cuales el último en leerse o escribirse será el byte de suma de control. Éste representa la suma de los otros bytes (cada uno de los cuales posee un valor entre 0 y 255) módulo 256; su significado corresponde al resto de la suma cuando se la ha dividido por 256. He aquí un ejemplo:

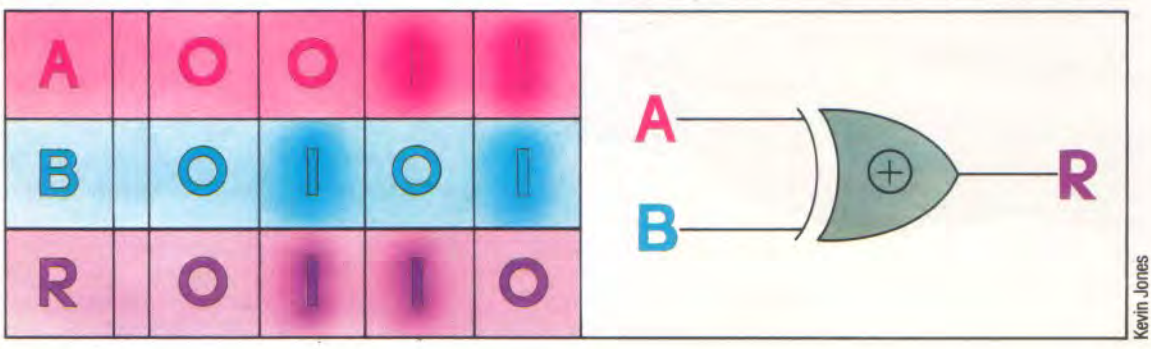
Datos: 114,67,83... (otros 121 valores)... 36,154,198
Total de estos 127 bytes = 16 673
Total dividido por 256 = 65, resto 33
Por tanto, suma de control = 33

El total de los bytes (16 673) es igual a 65 porciones de 256 más un resto de 33 (el valor escrito en el byte 128 como suma de control). Cuando el ordenador vuelve a leer el bloque, efectúa su propio cálculo de suma de control de la información, y si este valor difiere de 33, entonces sabe que en el proceso de grabación se ha producido un error de bits.

Tanto al emplear el método de paridad como el de suma de control, el ordenador no dispone de ningún medio que le permita saber cuál es el bit del dato que se ha alterado. Si el error se produjo en la transmisión, entonces el ordenador receptor puede solicitar que se le vuelva a transmitir un byte o un bloque de bytes determinados; en el caso de un error de grabación, bien podría ser que no hubiera forma alguna de recuperar el dato correcto.

Cuando sea indispensable evitar que se produzcan errores, se ha de utilizar un sistema que sea capaz tanto de detectarlos como de corregirlos. Los *códigos Hamming*, así llamados en honor a su inventor, R. W. Hamming, de Bell Telephone Laboratories, cumplen esta función.

Todos los sistemas de corrección de errores trabajan sobre el principio de la redundancia. El lenguaje humano contiene un elevado nivel de redundancia; si al mecanografiar un manuscrito se comete un error, o si durante una conversación telefóni-



Puerta "Or" exclusiva
Una puerta "Or" exclusiva sencilla tiene dos entradas y una salida. Si ambas entradas están en un 0 lógico, entonces la salida es 0. Si alguna de las entradas es 1, entonces la salida es 1. No obstante, si ambas entradas son 1, entonces la salida es 0. Esta última situación es la que determina la diferencia entre la puerta Or y la Or-ex (para mayor brevedad). La operación se puede representar con una tabla de verdad. Cuando una Or-ex posee más de dos entradas, la salida será 1 si en la entrada hay un número impar de unos. Es mediante dispositivos de este tipo como se crean los bits de control de paridad y de error

ca alguna interferencia impide oír algunas palabras, con frecuencia éstas se pueden deducir del contexto de la oración. A veces incorporamos redundancia extra al hablar en entornos "ruidosos": cuando en las comunicaciones telefónicas empleamos los términos "Almería", "Barcelona" y "Cáceres" para dar a entender inequívocamente que nos referimos a una "a", una "b" y una "c", por ejemplo.

Supongamos que desde un ordenador enviamos una palabra de x bits de longitud, compuesta por y bits de datos reales y por z bits redundantes (es decir, $x = y + z$). En nuestra explicación de la paridad teníamos para y un valor de siete y para z, otro de uno. Para los códigos Hamming, z habrá de ser proporcionalmente mayor. Supongamos ahora que en cualquiera de los x bits se pueda producir un error de un solo bit (nuestros bits redundantes z, por supuesto, son tan susceptibles de error como los bits de datos y). Si la probabilidad de que en una palabra se produzca un error de bits es, pongamos por caso, de una en un millón, la probabilidad de que en una palabra tengan lugar dos errores es de una en un millón de millones, de modo que descartaremos esta eventualidad.

Cuando los datos se reciban al otro extremo, habrá $x + 1$ probabilidades. O no habrá ningún error, o habrá un error en el primer bit de datos, y así sucesivamente hasta el último bit x. Ahora bien, con bits redundantes z podemos representar 2^z situaciones, de modo que para que la palabra esté a prueba de un error de bits:

$$2^z \geq y + z + 1$$

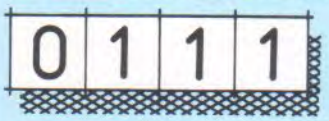
Si y es siete (en código ASCII), entonces z habrá de ser cuatro. Si y es cuatro (como en nuestro ejemplo del panel), z habrá de ser tres. Sin embargo, si y es 16, z sólo se habrá de aumentar a 5. Los códigos Hamming son más eficaces para palabras largas que para las cortas.

En un código Hamming, cada uno de los bits redundantes actúa como un control de paridad par en una combinación diferente de los bits de la palanca. Si en la transmisión se cambiara algún bit, uno o más de los bits de control estaría mal y la combinación de estos bits señalaría el bit erróneo de la palabra (véase ejemplo). El software del ordenador receptor puede entonces volver a cambiar el bit.

La clave para la forma en que funcionan los códigos Hamming son las distintas combinaciones de bits sobre las cuales actúa como control de paridad cada bit Hamming. El número total de bits, efectivamente, se divide en juegos distintos pero superpuestos, diseñados para que no aparezcan dos bits en la misma combinación de juegos. El ordenador receptor efectúa los controles de paridad en los mismos juegos que lo hizo el dispositivo transmisor para crear el código Hamming. Si cualquiera de los bits, incluyendo los bits Hamming, se hubiera alterado durante la transmisión, entonces uno o más de estos juegos no pasaría la prueba de paridad.

Algunos ordenadores utilizan los códigos Hamming incluso para sus operaciones de memoria interna. Cuando es éste el caso, ¡se puede quitar un chip de RAM entero y ver cómo el ordenador sigue funcionando! Algunos ordenadores destinados a uso militar llevan el principio de la redundancia hasta el extremo de duplicar cada componente simple del ordenador y comparar, luego, los resultados de las dos mitades después de cada operación.

Cómo funciona un código Hamming



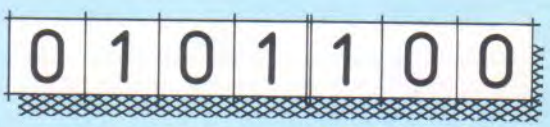
Supongamos que deseamos enviar estos cuatro bits de datos

A ellos les debemos agregar un código Hamming de tres bits, un patrón de bits exclusivo generado por el ordenador para satisfacer las siguientes condiciones:

Mirando sólo a estos cuatro de los siete, el número de unos visibles debe ser par

Del mismo modo, entre estos cuatro debe haber un número par de unos

Y en este juego de bits, debe haber asimismo un número par de unos. Para elaborar los tres bits que satisfagan estas condiciones, el ordenador debe resolver tres ecuaciones simultáneas

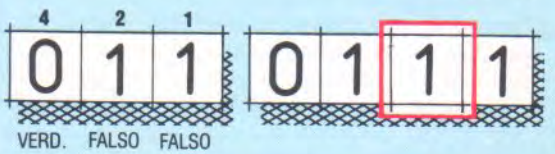


Pero imaginemos que durante la transmisión el tercer bit empezando por la izquierda se altera, es decir, se cambia de 1 a 0

Al efectuar el ordenador receptor en los datos la primera de las tres pruebas, ésta fracasará porque el número de unos visibles es impar. Esto nos indica que se ha producido un error, pero aún no sabemos cuál es el bit afectado

De igual manera, la segunda prueba muestra un resultado falso

Sin embargo, los datos pasan con éxito la tercera prueba: se observa un número par de unos



Es la combinación de las pruebas satisfactorias y las fallidas lo que indica el bit erróneo. Si expresamos una prueba fallida como un 1 y una prueba satisfactoria como un 0, escribiendo después los resultados en orden inverso obtendremos el número binario tres, que señala que el tercer bit estaba alterado y que se debe volver a cambiar de 0 a 1

Este principio funcionará aun cuando fuera uno de los bits Hamming el que se hubiera alterado. Por ejemplo, si todas las pruebas fracasaran, 111 indicaría que el bit erróneo era el de la derecha, mientras que si los tres dieran resultado satisfactorio, no habría habido error. Este tipo de código de corrección fracasaría sólo si en los siete bits se hubiera producido más de un error

Norbert Wiener

El científico norteamericano a quien se considera el padre de la ciencia cibernética

Norbert Wiener nació en 1894 en Missouri (Estados Unidos). Después de graduarse en matemáticas a la edad de 14 años y de obtener un doctorado en lógica a los 18, se trasladó a Gotinga (Alemania) para estudiar con David Hilbert.

La contribución de Wiener a la ciencia de la informática se produjo al final de su vida. Durante muchos años trabajó en el Massachusetts Institute of Technology, estudiando la nueva física probabilista y concentrándose en el estudio estadístico del movimiento de las partículas en un líquido (fenómeno conocido como "movimiento browniano"). Los movimientos de las partículas eran tan imprecedibles que era imposible describirlos utilizando la física tradicional de las fuerzas deterministas. De modo que lo más apropiado era aplicar un método "probabilístico", en virtud del cual sólo se podía predecir en un momento dado la localización *probable* de una partícula determinada.

Cuando estalló la segunda guerra mundial, Wiener ofreció sus servicios al gobierno de Estados Unidos y empezó a trabajar en los problemas matemáticos que implica apuntar un arma hacia un blanco móvil. El desarrollo de los sistemas automáticos para el guiado de la mira, sus estudios de física probabilística y su marcado interés por temas que iban desde la filosofía hasta la neurología, todo ello se conjugó en 1948 cuando publicó un libro titulado *Cibernética, o Control y comunicación entre el hombre y la máquina (Cybernetics)*.

La cibernética es el estudio de los controles autogobernados que existen en los sistemas estables, ya sean mecánicos, eléctricos o biológicos. Fue Wiener quien vio que la información era cuantitativamente tan importante como la energía o la materia: un alambre de cobre, por ejemplo, se puede estudiar por la energía que puede transmitir o la información que puede comunicar. La revolución que anuncia el ordenador se basa en parte en esta idea: la fuente de poder pasa de la propiedad de la tierra, la industria o la empresa al control de la información. Su contribución a la ciencia de la informática no consistió en el diseño de elementos de hardware, sino en la creación de un medio intelectual en el cual se pudieron desarrollar los ordenadores y los autómatas.

El término *cibernética* deriva de una palabra griega que significa "arte de gobernar". Wiener había estudiado el regulador del motor a vapor de James Watt, que ajustaba automáticamente la velocidad de la máquina, y comprendió que para que fuera posible desarrollar los ordenadores, éstos deberían imitar la capacidad de los seres humanos de regular sus propias actividades.

El termostato de una casa constituye un ejemplo de un sistema de control. Regula la calefacción según las fluctuaciones de la temperatura por encima o por debajo de un nivel óptimo. Sólo se necesita que un ser humano determine este nivel. A esta facultad de autorregulación y control Wiener la denominó *feedback negativo* ("feedback", porque la salida del sistema —el calor— afecta al comportamiento futuro del sistema, y "negativo", porque las variaciones del termostato se producen para restaurar la temperatura al nivel establecido).

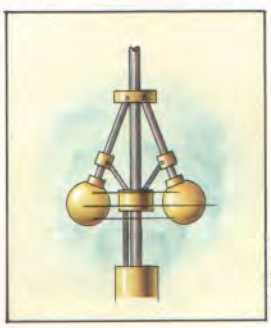
Se dice que el sistema que puede cumplir esta función y también seleccionar su propia temperatura (y lograr otros objetivos) es un sistema de *feedback positivo*. Cuando un autómata puede realizar todos estos cometidos y además reproducirse a sí mismo, entonces se acerca a la condición humana.

La teoría de la cibernética de Wiener se puede considerar como una superciencia (una ciencia de ciencias) y ha fomentado la investigación en muchas áreas de sistemas de control y de sistemas que tratan con la información. Todo es información. Todo cuanto sabemos acerca de los cambios del mundo nos llega a través de nuestros ojos y nuestros oídos y otros receptores sensoriales, que son dispositivos para seleccionar sólo ciertos datos de un total que, de lo contrario, nos desbordaría.

La información también se puede estudiar de forma estadística, independientemente de cualquier significado que pueda tener. Por ejemplo, observando la frecuencia con que se producen ciertos símbolos se pueden interrumpir muchos tipos de códigos. En castellano, las letras "a" y "e" están entre las que aparecen con mayor frecuencia. Analizando grandes muestras de un código y comparando los resultados con muestras típicas de castellano, se pueden identificar letras clave y, por consiguiente, empezar a descifrar el código.

Wiener murió en 1964, antes de que empezara la revolución del microordenador, a pesar de lo cual previó muchos de los problemas que surgirían en esta nueva tecnología y escribió acerca de ellos.

Restricción de velocidad
Wiener quedó fascinado con la idea del regulador a vapor, uno de los mejores y más sencillos ejemplos de feedback negativo. Mediante brazos pivotantes se conectan dos pesas a un eje giratorio, que a su vez está conectado al volante de la máquina de vapor. A medida que aumenta la velocidad del motor, las pesas vuelan hacia afuera. Este movimiento, mediante una articulación apropiada, cierra ligeramente la válvula de estrangulación del motor. Esto tiene el efecto de establecer la velocidad de la máquina en cualquier nivel determinado por el operador. Los ordenadores modernos pueden realizar tipos de control mucho más sofisticados, pero el principio sigue siendo el mismo



Kevin Jones

Comunicación por cable

La televisión por cable abre un nuevo camino para que los usuarios de ordenadores personales se comuniquen entre sí

Ian McKinnell

Ciudad para el futuro
Milton Keynes, una ciudad que se construyó empezando desde cero, dispuso desde el principio de televisión por cable. Los 22 000 hogares de la ciudad tienen acceso a siete canales de televisión (seis de programas variados y en directo y uno que sólo emite películas) y a seis canales de radio VHF (Very High Frequency: frecuencia muy alta). El siguiente paso hacia una red de información integrada prevé que los suministros de gas y electricidad se midan centralmente y no en cada edificio. Pronto se agregarán redes de área local que se instalarán en los edificios públicos, así como un sistema comunitario de videotex

Se habla mucho acerca de los beneficios que nos reportaría la denominada "revolución del cable", que nos permitiría contemplar en nuestro hogar 20 o 30 canales de televisión, pero se habla muy poco de un efecto colateral que podría cambiar la forma en que nos comunicamos con los otros miembros de la comunidad. Porque uno de esos canales se podría reservar para que sirviera como vínculo entre los ordenadores personales. Ya hemos analizado dos procedimientos para utilizar éstos como terminales de comunicaciones: los sistemas de videotex (véase p. 268), que permiten acceder a una gran base de datos general, y las redes de área local (véase p. 218).

Utilizando un sistema de videotex, como el Mailbox de Prestel, en Gran Bretaña, les es posible a los suscriptores enviarse mensajes, y cualquiera de ellos podría contratar bienes y servicios (p. ej., vacaciones) directamente del proveedor de información. Sin embargo, estos dos servicios son de aplicación limitada. Del mismo modo, si el ordenador que emplea el usuario forma parte de uno de los centros de trabajo de una red de área local, le será posible comunicarse con cualquier otra estación; pero incluso la más grande y poderosa de las redes basadas en microordenadores abarcará solamente dos o tres kilómetros y es poco probable que tanto el banco como el supermercado y la farmacia de los que el usuario es cliente se hallen dentro de ese perímetro. Por supuesto, éste puede estar suscrito a un sistema de videotex y ser miembro de una red de área local. No obstante, la situación distaría de parecerse a la que disfrutaría si fuera miembro de una comunidad en la que cada hogar se hallara conectado electrónicamente a los demás.

El factor físico más importante que impide la creación de redes de este tipo que cubran ciudades enteras, es la pérdida de señal en el medio de transmisión, que se produce ya sea utilizando pares de cables enroscados, cables coaxiales (como el de la antena de su televisor) o fibras ópticas. Es este fenómeno lo que en la actualidad limita las dimensiones de las redes de área local, y la única forma de superar el problema consiste en insertar estaciones reamplificadoras o "reforzadoras" en la red a intervalos frecuentes.

Una segunda consideración es la de la "densidad del tráfico" o el volumen de información a comunicar. Éste influye en la anchura de banda o alcance de las frecuencias de transmisión requeridas. Por regla general, se necesita una anchura de banda de dos hertzios para cada bit por segundo que se desee transmitir. Una transmisión de 300 baudios (300 bits por segundo) exige 600 Hz; por su parte, una transmisión de 1 200 baudios necesita 2,4 kHz. El habla normal utiliza un mínimo de 3 kHz a través de una línea telefónica. No obstante, la transmisión de una imagen de televisión en color requiere 8

MHz (tres mil veces la del habla). En otras palabras, la anchura de banda que se necesita para transportar una imagen de televisión podría difundir 3 000 conversaciones telefónicas separadas.

La capacidad de un medio de transmisión tal vez se exprese mejor en términos del número de conversaciones telefónicas que puede retener. En Gran Bretaña, el cable de mayor capacidad que emplea actualmente la British Telecom puede transmitir 20 000 conversaciones a la vez, pero en pruebas experimentales dicha empresa ha conseguido evaluar en 500 MHz el cable coaxial, lo que implica que potencialmente es capaz de transportar 167 000 conversaciones simultáneamente. El cable tiene 1 cm de espesor. Se le podría comparar con la tecnología de las fibras ópticas, según la cual una simple hebra de vidrio más delgada que un cabello tiene capacidad para difundir hasta 10 000 conversaciones telefónicas.

El hardware para comunicaciones ya está en el mercado, y crear una red para una población o una ciudad sería una cuestión sencilla. Consideremos ahora cómo podríamos establecer una red informática comunitaria basada en el microordenador BBC Modelo B utilizando el Econet de Acorn. Cuando analizamos las redes de área local, señalamos que el controlador de la red no podía estar situado a más de 500 metros de ningún centro de trabajo, lo que implica que nuestra red puede cubrir una circunferencia de un kilómetro de diámetro. También vimos que puede dar cabida hasta a 254 miembros, dejando una estación dedicada a servir el archivo (manejando el disco comunitario) y otra para controlar la impresora. Pero si reserváramos otra estación como canal de comunicación con una segunda red, entonces un programa relativamente sencillo nos permitiría conectar dos redes entre sí. La conexión requeriría dos máquinas dedicadas a pasar mensajes de una red a la otra, comunicándose a través de sus respectivas conexiones en paralelo. Cuanto mayor fuera el número de máquinas que estuviéramos dispuestos a reservar con este fin, más redes podríamos conectar entre sí.

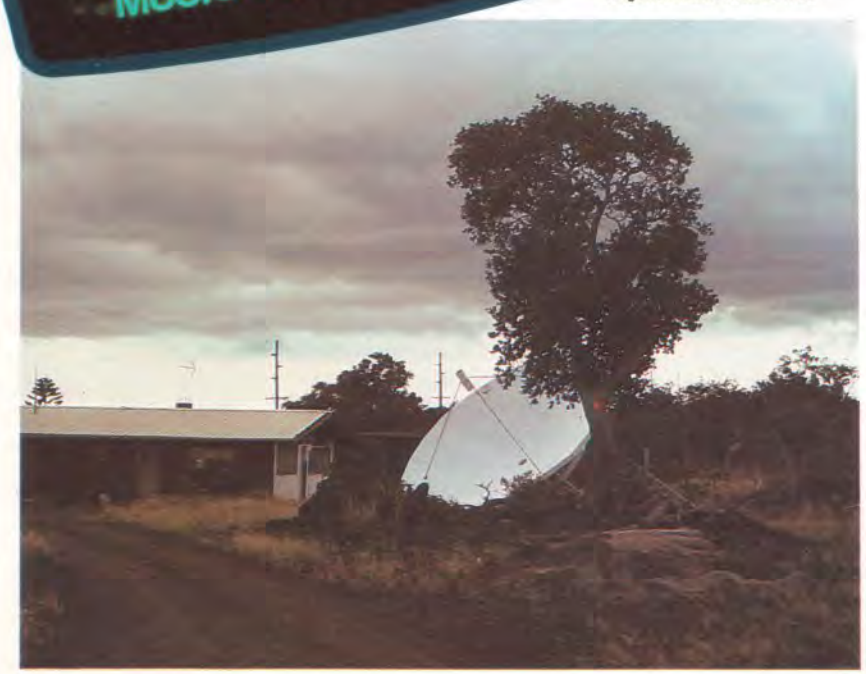
Por supuesto, ésta es una solución muy provisional al problema de enlazar por cable una comunidad entera, pero si la idea tuviera verdadera aceptación se podría reemplazar por una conexión de red construida especialmente. Sin embargo, es poco probable que se pudiera realizar una conexión de este orden dado que requiere que todos los miembros utilicen el mismo tipo de microordenador. Para que sea realmente eficaz, una red así ha de ser completamente "transparente", lo que significa que debe permitir que un Spectrum, por ejemplo, se comunique con un Dragon 32. Ello exige un controlador central del sistema que manipule la conversión entre los diversos protocolos que utilizan los distintos fabricantes de ordenadores. Asimismo, el controlador podría ser el punto de enlace con el Prestel y otros servicios de videotex, el sistema bancario y las entidades financieras, el servicio sanitario y otros servicios públicos, y todas las otras áreas de la sociedad que se hubieran informatizado sin contar antes con la compatibilidad entre las máquinas.

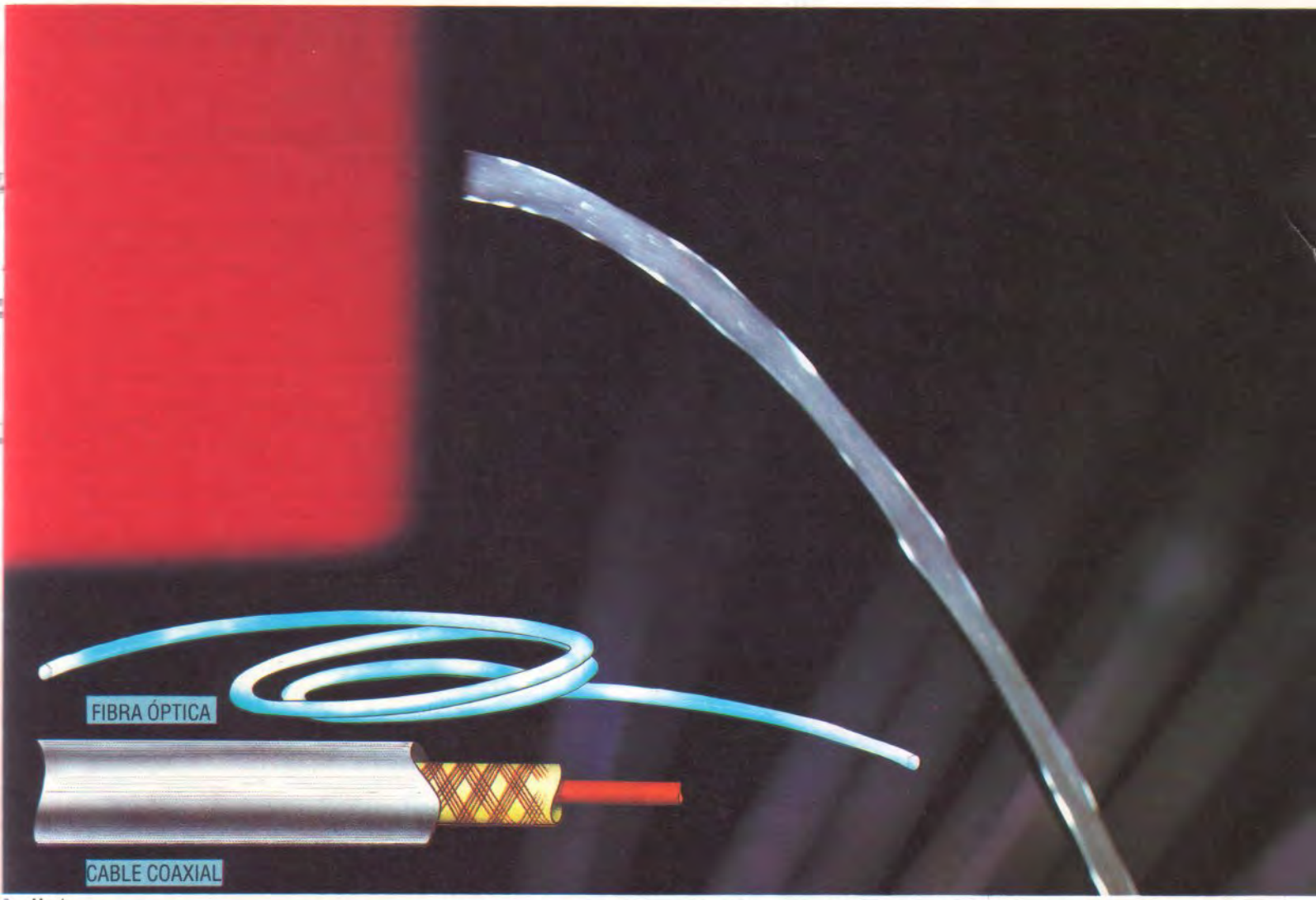
¿Quién proporcionaría los fondos para instalar y hacer funcionar un servicio de tales características? La experiencia británica sugiere que éste es el punto más conflictivo. Cuando el gobierno británico formuló propuestas por primera vez para la instalación de un sistema de televisión por cable, puso especial énfasis en los beneficios que tal sistema reportaría a la industria de la tecnología de la información, y, en consecuencia, varios usuarios potenciales de dicho sistema se abocaron a proyectos de investigación. Una cadena nacional de supermercados, en un gesto de audacia, puso en marcha un sistema piloto propio de telecompra para probar su viabilidad y la reacción del público ante el mismo. El esquema no sobrevivió ni siquiera a su breve período de prueba, y en todas partes se llegó a la misma conclusión: los servicios de esas características tenían poca aceptación.

Cuando se presentó en el Parlamento británico el borrador del proyecto de ley proponiendo la instalación de la televisión por cable, aún contenía referencias a servicios informáticos como "telecompra" y "telebanco", pero decía más abiertamente que la dirección del sistema del cable sólo habría de "fomentar la provisión de servicios informáticos en dos direcciones". En consecuencia, se concedía mayor importancia a la tecnología del entretenimiento que a la de la información. De hecho, a los operadores del cable se les prohibiría expresamente ofrecer cualquier forma de servicio telefónico, ni siquiera facilidades para videoconferencias, si bien el borra-

Cortesía de MTV

Estación satélite
MTV utiliza un sistema de transmisión por satélite para emitir programas a sus 1 650 compañías afiliadas, que luego los envían por cable a los hogares de los trece millones y medio de suscriptores. MTV proporciona 168 horas de programación semanal, toda en estéreo. Se incluyen 8 minutos de publicidad por hora de emisión. Aunque el servicio es estrictamente en una sola dirección, la empresa fomenta un cierto grado de interacción de la audiencia a través de números de teléfono gratuitos. Con la introducción del tráfico por cable en dos direcciones, este tipo de acción recíproca instantánea de los televidentes se generalizará mucho más





Gary Marsh

Impulso rápido
Las fibras ópticas, que se desarrollaron por primera vez en 1966 en los Standard Telecommunication Laboratories de Harlow (Inglaterra), se basan en el fenómeno de la reflexión interna total. La luz introducida por un extremo viajará a través de la fibra de vidrio con una pérdida muy escasa de luminosidad, volviéndose a reflejar cada vez que golpea contra la "pared" exterior, al igual que lo haría en un prisma o en una corriente de agua. Para conseguir este resultado, el nivel de pureza del vidrio empleado ha de acercarse al 100 %. Se utiliza la luz de un láser infrarrojo, porque la señal se debe impulsar (encenderse o apagarse) muy rápidamente. Los cables coaxiales, familiares por su utilización para conexiones a la antena de los televisores, constan de un único cabo de alambre de cobre grueso, rodeado por una pantalla de cable más delgado tejido en un tubo. Estos dos conductores están aislados el uno del otro

dor de un nuevo documento contempla la posibilidad de que finalmente la red por cable asuma todos los servicios de telecomunicaciones.

La experiencia de otros países es similar. En Estados Unidos, donde la televisión por cable existe desde hace mucho tiempo, aún se utiliza muy poco el sistema de cable para proporcionar un medio de comunicaciones digital en dos direcciones. Incluso allí donde se emplea parece que se hace más bien con fines triviales.

En los países escandinavos y en Holanda ha habido un movimiento tendente a crear una interconexión de ordenadores personales a nivel comunitario. En la actualidad se encuentran en funcionamiento redes locales de correo electrónico y de contratación de niñeras por medio de ordenador; algunas comunidades incluso efectúan plebiscitos informales acerca de temas de interés local. Lamentablemente, estos beneficios sociales tan evidentes no siempre son tenidos en cuenta, en algunos países, por los patrocinadores de sistemas de red por cable. Tal vez quienes perciban con mayor lucidez los beneficios de las redes de comunicaciones sean aquellos miembros de la comunidad que tienen muy poco interés en adquirir siquiera un sencillo microordenador. Una vez instalados los cables interactivos, se podrían crear incentivos para que las industrias de servicios, como los bancos o el servicio de telecomunicaciones, distribuyan terminales a un costo reducido, o incluso completamente libres de todo cargo, como sucede en algunas regiones de Francia, donde la red nacional de teléfonos ha sustituido las guías telefónicas y el servicio de consulta de la guía por terminales de videotex.

© BBC Stills

¿Vale una imagen más que mil palabras?
Es difícil determinar el número de bits digitales que se necesitan para componer una imagen de televisión en color, pero considerando que la anchura de banda de 9 MHz permite la transmisión de 4,5 millones de baudios, y que cada imagen completa se transmite 25 veces por segundo, el simple cálculo aritmético nos da una cifra de 180 000 bits por fotograma. Por otra parte, la emisión de mil palabras requiere unos 60 000 bits

Las nuevas memorias

Hallar medios más compactos de almacenar información ha sido la meta de los diseñadores desde que se inventara el ordenador

Desde que se investigara por primera vez sobre la electricidad, los científicos han estado buscando formas de poder utilizarla para almacenar información. La electricidad se puede concebir o como un flujo de electrones o como una onda en movimiento, pero en ambas descripciones la característica implícita es el movimiento. La imposibilidad de retener algo que por su propia naturaleza siempre ha de estar desplazándose, ha llevado a la adopción de métodos de almacenamiento indirecto. Se han hallado muchas soluciones al problema de manipular datos en forma de señales eléctricas.

Durante la segunda guerra mundial se trabajó intensamente en el radar, para separar la señal reflejada desde un aeroplano en movimiento de los impulsos que reverberaban en objetos fijos, como los árboles. Se inventó un dispositivo, denominado *línea de demora de mercurio*, que podía almacenar temporalmente los impulsos de las ondas de radio y comparar así las ondas reflejadas en los sucesivos "barridos" del radar, de modo que se podían eliminar los patrones permanentes.

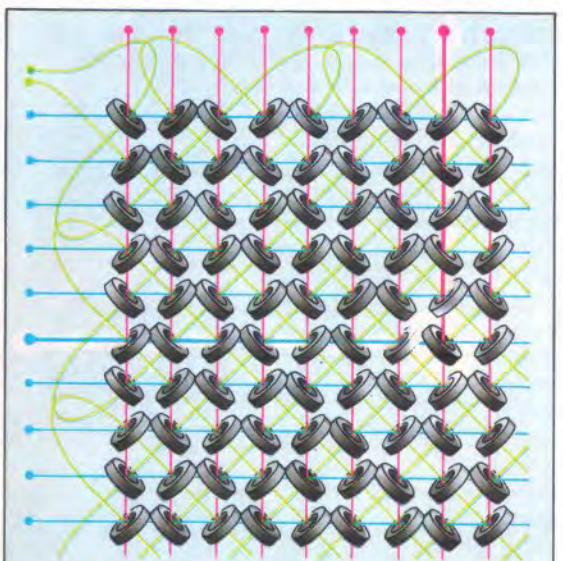
Una línea de demora de mercurio se compone de un tubo de vidrio de un metro de longitud, lleno de mercurio, y con un cristal de cuarzo a cada extremo. Cuando se aplica una señal eléctrica en uno de los extremos, el cristal de cuarzo vibra y crea una onda física que viaja hasta el otro extremo del tubo, donde el otro cristal la detecta y la vuelve a convertir en una señal eléctrica. La onda tarda aproximadamente 660 microsegundos en desplazarse a través del tubo, pero volviendo a alimentar la señal en un bucle a través del mercurio, se pueden almacenar los impulsos durante varios minutos antes de que la señal se distorsione demasiado.

Con un reloj que regulaba la secuencia de impulsos a 500 000 ciclos por segundo, se podían almacenar 330 bits en un tubo de un metro. El tamaño de la memoria se podía ampliar alargando el tubo, pero sólo al precio de aumentar el tiempo de acceso, dado que un impulso sólo se podía leer cuando estaba en el circuito eléctrico fuera del tubo. Estos sistemas eran caros y voluminosos, ya que requerían una ingeniería de gran precisión para construir juegos de tubos de una exactitud de una centésima de milímetro.

En Gran Bretaña, F. C. Williams inventó en la Universidad de Manchester otro método para almacenar datos; Williams investigaba el empleo de la electricidad estática generada en la superficie interior de un tubo de rayos catódicos (una pantalla de televisión) como medio de almacenamiento. Un "disparador de electrones" componía en la pantalla un patrón de cargas estáticas, y este patrón se podía detectar por una rejilla de alambre próxima a la superficie exterior de la pantalla de vidrio. Hacia 1947 se podían almacenar 2 048 bits de información durante varias horas en una única pantalla. Aunque la velocidad de acceso era alta, las cargas estáticas se habían de refrescar cada 30 microsegundos, ya que de lo contrario se extinguían y se perdían.

La utilización de la cinta magnética se probó por primera vez con el LEO (véase p. 320) en Gran Bretaña, y en Estados Unidos con el UNIVAC (Universal Automatic Computer) a fines de los años cuarenta. Ésta fue la primera técnica en virtud de la cual se pudo almacenar gran cantidad de información de forma barata y fiable. Se almacenaron ocho bits en un "cuadro" a través de la cinta a una densidad de 2 360 "cuadros" por centímetro. Con una longitud total de cinta de 61 metros o más, se consiguieron memorias permanentes de al menos un megabyte. No obstante, aun con la rapidez de los motores activadores, el tiempo de acceso a los datos en mitad de la cinta era de algunos segundos. En consecuencia, esta clase de almacenamiento halló su utilización natural en aquellos archivos donde los datos se requieren secuencialmente, como en el cálculo de la nómina de una empresa. La cinta se podía acelerar haciendo flotar la "cabeza" que detecta las señales magnéticas en un "colchón" de aire, que también alivia el desgaste por uso y tensión.

En 1950 se inventó en el Massachusetts Institute of Technology una memoria de gran capacidad, gran velocidad de acceso y bajo costo por bit. La memoria "de núcleos de ferrita" empleaba anillos de hierro ensartados en una rejilla de alambres que se intersectaban entre sí. Haciendo pasar una corriente por un alambre que atravesaba un anillo de hierro, el metal se magnetizaba. En el campo mag-



Memoria de núcleos de ferrita

En una rejilla de alambres cruzados se ensartan anillos de hierro. Cualquier anillo se puede direccionar enviando una corriente a través de los alambres horizontales y verticales. El alambre lector, que traspasa todos los anillos, recoge los cambios que se producen en el flujo magnético del anillo, indicando si hay almacenado un 1 o un 0

Kevin Jones

nético del anillo hay dos direcciones posibles y éstas se utilizan para representar los dos estados binarios, 0 y 1. La dirección del magnetismo del anillo se puede "invertir" mediante una corriente adecuada. Y así como una corriente induce magnetismo, también sucede lo contrario: cuando "se invierte" la dirección del magnetismo, se induce una corriente inversa pequeña pero detectable. El campo magnético sólo se invertirá cuando se emplee una corriente por encima de un umbral determinado. Proporcionando la mitad de la corriente umbral a través de un alambre vertical y la otra mitad a través de un alambre horizontal, sólo uno de los anillos del enrejado recibirá suficiente corriente como para invertirse.

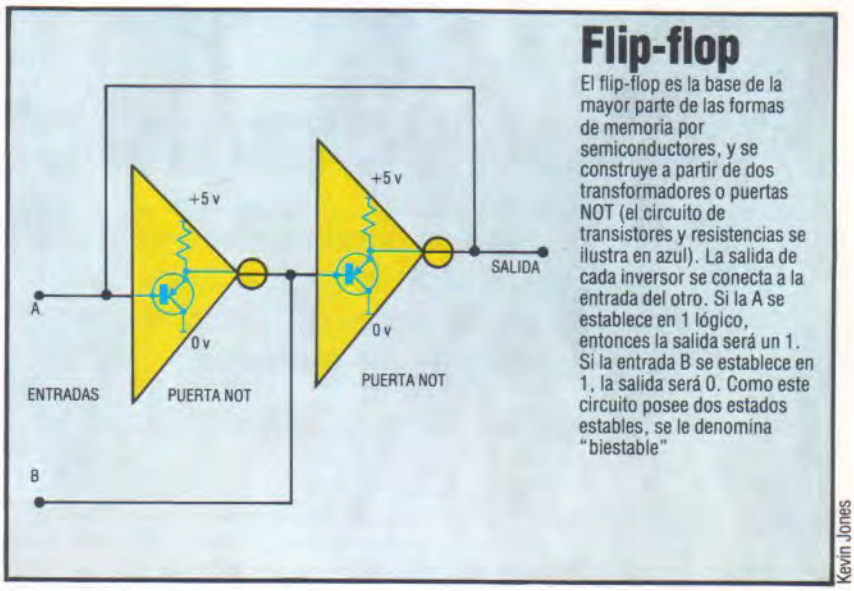
El diámetro de los anillos individuales disminuyó con el tiempo de aproximadamente cuatro milímetros a algunas centésimas de milímetro, reduciendo, de este modo, la energía necesaria para escribir datos en una celda. La información de una celda de memoria determinada se leía intentando invertirla. Mediante el control de una corriente inducida se podía descubrir si la celda se había invertido o no y, en consecuencia, se podía deducir su estado original. Sin embargo, este método de lectura colocaba todas las celdas en un estado magnético, de modo que después de cada lectura se debían volver a escribir los datos.

Con la integración a gran escala (LSI) de los transistores en un solo chip, se ha vuelto a restablecer uno de los primeros métodos para retener datos (que se venía utilizando desde el ENIAC): el *flip-flop* (circuito biestable). Un flip-flop es un dispositivo que es estable en uno de dos estados (de allí su característica de "biestable"). Se construye a partir de dos interruptores electrónicos uno detrás del otro, alimentando la salida de cada uno en la entrada del otro. De este modo se puede "atrapar" un impulso en el flip-flop hasta que se lo requiera.

Cada bit necesita dos dispositivos interruptores; además, los primeros flip-flops de válvula eran caros, nada fiables y se quemaban muy a menudo. Sin embargo, con los circuitos integrados, en los cuales en un único chip se incorporan cientos de miles de interruptores transistorizados, el flip-flop ha vuelto a recuperar su importancia.

Los chips de RAM "estática", como los que poseían todos los primeros ordenadores personales, contienen miles de circuitos biestables, uno para cada bit. No obstante, la mayoría de las máquinas más nuevas incorporan una RAM "dinámica". El usuario no nota ninguna diferencia, pero la RAM dinámica es más rápida y su construcción es más barata. En realidad es como una matriz de condensadores, cada uno de los cuales puede almacenar una carga eléctrica. El inconveniente está en que, al tratarse de una carga almacenada, tiende a perderse; por este motivo, los contenidos se han de "refrescar" miles de veces por segundo, lo que se lleva a cabo mediante un sistema de circuitos especial incorporado en el chip.

Los tipos de memoria más recientes que existen en el mercado son la memoria de "burbuja" y la memoria de láser óptico. Esta última es similar a los discos que se utilizan para almacenar música o películas de video y su lectura se realiza por medio de un haz de rayos láser. En la memoria de burbuja, en el interior de un chip de material magnético se crean diminutos campos magnéticos o burbujas. Estas burbujas se pueden mover en bucles y



Flip-flop

El flip-flop es la base de la mayor parte de las formas de memoria por semiconductores, y se construye a partir de dos transformadores o puertas NOT (el circuito de transistores y resistencias se ilustra en azul). La salida de cada inversor se conecta a la entrada del otro. Si la A se establece en 1 lógico, entonces la salida será un 1. Si la entrada B se establece en 1, la salida será 0. Como este circuito posee dos estados estables, se le denomina "biestable"

Kevin Jones

la presencia o la ausencia de una burbuja representa un 1 o un 0 binarios. Los bucles se mantienen pequeños para conseguir un tiempo de acceso rápido. Su densidad es potencialmente tan elevada que este tipo de memoria puede llegar a reemplazar los actuales sistemas de discos y cassettes de superficie móvil.

Las memorias criogénicas podrían, asimismo, llegar a ser realidad. Al alcanzar temperaturas próximas al cero absoluto, la resistencia eléctrica en un medio desaparece para proporcionar lo que se denomina superconductividad, y entonces resulta teóricamente posible almacenar una carga equivalente a la de un electrón. Las cargas sobre las que operan las clases actuales de memoria de chips son muy pequeñas, pero aun así cada una de ellas equivale a algunos millones de electrones. Con la superconductividad que se obtiene enfriando estos medios de memoria a temperaturas criogénicas en baños de helio líquido, la velocidad y la capacidad de las memorias alcanzan un punto que supera muchísimo al que ofrecen los sistemas actuales.

Biotecnología
Pese a haber alcanzado la electrónica un increíble nivel de miniaturización, toda operación en una memoria por semiconductores implica el flujo de muchos miles de electrones. Los científicos ya están intentando desarrollar memorias que funcionen a nivel molecular, es decir, con electrones individuales, lo cual tiene más relación con el campo de la biotecnología que con el de la microelectrónica. Se sabe, por ejemplo, que la estructura de una molécula de DNA (ácido desoxirribonucleico) almacena el código genético de los seres vivos. No obstante, pasarán muchos años antes de que el hombre sea capaz de reproducir una molécula similar

Tablones de anuncios electrónicos

¿Qué se puede hacer con un modem? Cualquier usuario de ordenador puede acceder a los tablones de comunicaciones electrónicos, que no son sino la versión computerizada de los tablones de anuncios

Un ordenador personal por sí solo tiene una capacidad de comunicación más bien limitada, pero mediante un teléfono puede "hablar" con toda una gama de ordenadores diversos, ya sean poderosos ordenadores universitarios de unidad principal, bases de datos públicas a gran escala o, simplemente, con el ordenador personal de la casa de al lado.

En Gran Bretaña, a través del BBS (Bulletin Board Services: servicio de tablones de comunicaciones), cualquier usuario de ordenador puede hacerles llegar mensajes a otros usuarios, anunciar artículos que desee vender, e incluso intercambiar programas. Un BBS es, en realidad, el equivalente electrónico de los tablones de anuncios que se encuentran en la mayoría de las sedes de clubs o asociaciones. Por lo general están a cargo de voluntarios que utilizan un ordenador personal corriente con un gran dispositivo de almacenamiento (p. ej., un disco), al que puede acceder cualquier persona. La distancia existente entre un ordenador y otro no cuenta: se puede acceder a máquinas de la misma ciudad, de centros urbanos situados en el otro extremo del país e incluso de otro continente. La única restricción, por supuesto, la constituirá el importe de la factura telefónica.

Sin embargo, antes de que el ordenador personal pueda empezar a "hablar" por teléfono, se necesita un elemento de hardware que los conecte a los dos. Dicho dispositivo se denomina *modem* (véase p. 108) y se enchufa en la conexión en serie (RS232), situada en la parte posterior del ordenador. Algunas máquinas, como el Vic-20 y el Spectrum, no poseen dicha conexión; en tal caso, el usuario también tendría que adquirir una ficha de interface en serie para enchufar en la parte posterior. El modem se conecta a la línea telefónica ya sea directamente, a través de un enchufe telefónico extra, o bien acústicamente, empleando un acoplador acústico. Existen dos "velocidades" de modem (velocidades baudio) de uso generalizado: 300 y 1200/75 baudios. Los operadores comerciales, como el Prestel y la British Telecom, utilizan la velocidad mayor, de 1200/75, mientras que el BBS (que es básicamente para ordenadores personales) emplea 300 baudios. Velocidad de 1200/75 significa que la información le llega al usuario a 1200 baudios y que éste, a su vez, la envía al ordenador central a 75 baudios. Por supuesto, es preferible comprar un modem que tenga velocidades baudio conmutables. Si el modem es del tipo de "conexión directa", el usuario también tendrá que utilizar un enchufe extra para él, que le alquilará la British Telecom.

Para hacer funcionar el modem también es necesario obtener software que le permita a la máquina cumplir el cometido de un terminal de ordenador. Existen dos tipos de software de terminal: el "mudo" y el "vivo". Un terminal vivo puede cargar programas y guardar mensajes de otros ordenadores. Un terminal mudo no puede realizar estas funciones, pero es más sencillo y, por lo tanto, más apropiado para ser empleado en los primeros intentos de comunicación por parte del usuario. Este tipo de software existe para los ordenadores personales "más antiguos", como el Apple II y el Tandy TRS80, así como para algunos de los modelos más nuevos; el BTERM es el programa para el BBC Modelo B. En cuanto a otros ordenadores, este software lo suelen proporcionar gratuitamente los grupos de usuarios que los utilizan. En caso de que no existiera, el usuario puede escribir sin excesiva dificultad su propio software para el terminal. Lo que necesita es un programa que envíe todos los caracteres digitados en el teclado a la conexión RS232 y que visualice en la pantalla los datos recibidos desde esta conexión.

Para ilustrar cómo se utiliza un modem, sigamos los pasos necesarios para comunicarse con un BBS. Primero, el usuario ha de averiguar el número de teléfono y los detalles técnicos (como la velocidad baudio) del ordenador al cual desea acceder. Después de ejecutar el software de comunicaciones en su ordenador, se procede a marcar el número de teléfono. Si el ordenador con el que se desea comunicar está "escuchando", el usuario oirá un tono alto a través del auricular. Éste es el "tono portador". Si el usuario estuviera utilizando un acoplador acústico, únicamente habría que insertar el receptor en las tazas de goma. Con un modem de los de tipo de "conexión directa", usted con sólo pulsar el interruptor marcado "LINE" o "DATA" reemplaza el receptor. En ambos casos, el ordenador debería estar conectado.

Probablemente lo primero que se vea en la pantalla sea algún tipo de "saludo", generado por el ordenador al otro extremo de la línea telefónica. Luego, al "huésped" se le pedirá su identificación y/o contraseña. Si el servicio al que se ha llamado es privado y el usuario no se encuentra registrado en él, entonces probablemente no conseguirá mucho más; ¡a menos que sea un buen adivino o una persona muy paciente! No obstante, muchos ordenadores permiten el acceso restringido a los "huéspedes" que no sean usuarios registrados, de modo que vale la pena probar con algunas palabras como

TELECOM GOLD

Correo electrónico
Telecom Gold es el sistema de correo electrónico de la British Telecom, una versión más sofisticada de los Bulletin Board Services destinada al mercado comercial. Los usuarios inscritos pueden alquilar un "buzón" en un miniordenador de la British Telecom, al que los otros usuarios pueden enviar sus mensajes. Éstos se pueden leer o copiar en el ordenador del usuario. El Telecom Gold permite acceder a otras redes electrónicas en otros lugares del mundo, con la ventaja de que todas las diferencias técnicas existentes entre las máquinas de los usuarios son obviadas automáticamente por los ordenadores que integran la red

Cortesía de British Telecom

Tablón de anuncios computerizado
Una de las aplicaciones más gratificantes del modem consiste en permitir el acceso a un tablón de anuncios electrónico. Éstos son versiones informatizadas de los tradicionales tablones de anuncios existentes en clubs y asociaciones diversas. Los mensajes se pueden "colgar" para que los lea cualquier persona o bien un usuario determinado que posea las contraseñas correctas. Se anuncian reuniones y se ponen en venta equipos de informática de segunda mano. Incluso se pueden cargar juegos u otros listados de programas en un disco o cassette

NEWUSER (nuevo usuario), GUEST (huésped) o HELP (socorro).

El BBS, por otra parte, está abierto a cualquiera y es gratuito (excepto en lo que atañe al costo de la llamada telefónica). Basta con que el usuario dé su nombre y el de su ciudad cuando se le solicite, y una vez "alistado" se le preguntarán detalles de su ordenador, como el ancho de la pantalla o quizá la marca. Así, el ordenador "anfitrión" lo identificará en futuras llamadas y adaptará el sistema para que funcione adecuadamente con su ordenador.

Una vez cumplido todo ello, se le proporcionará alguna información relativa al sistema, como los horarios de funcionamiento y detalles técnicos, y un límite de tiempo para su llamada (tal vez 30 minutos). Ahora se llega al menú principal, que está compuesto de una lista de media docena de órdenes, y el usuario debe elegir la que desea simplemente pulsando la letra inicial de la orden correspondiente. Por ejemplo, para registrarse como nuevo usuario habría de pulsar la N; para terminar su llamada tendría que digitar la G de *goodbye*. Con la mayoría de las opciones aparecerán nuevos menús, y el usuario podrá ir seleccionando la opción que le interese hasta llegar a lo que desea. El BBS es como un árbol: se empieza por el tronco (el menú principal) y se van escogiendo distintas "ramas" (bifurcaciones) con cada submenú. Prestel, al igual que la mayoría de las otras bases de datos, se vale de la misma idea.

La sección *New user* es para quienes desean inscribirse en el BBS. Al interesado se le solicita su nombre y dirección, y éste puede elegir su propia contraseña para identificarse en futuras llamadas. La orden *Information* (información) le proporciona una detallada relación técnica acerca del sistema. La sección *Utilities* (servicios) le informa de cuánto ha durado su llamada y también le da a conocer detalles relativos a todos los otros BBS.

La sección *Bulletins* (comunicaciones) contiene avisos al público que los usuarios han colocado en el sistema para que los lean otras personas. La sección *Messaging* (mensajes), por su parte, es para que el afiliado al servicio lea mensajes de índole privada que otras personas pueden haber dejado para él. Del mismo modo, puede enviarles mensajes privados a otros usuarios, o a un grupo de usuarios que posean algo en común (por ejemplo, que todos sean propietarios de ordenadores BBC Modelo B). Quizá estas características sean el aspecto más atractivo de los servicios y probablemente a ellas se debe su creciente popularidad.

Un servicio ligeramente diferente, denominado Rewtel, ofrece una base de datos especializada en componentes electrónicos e información relativa a los mismos. También acepta desde el teclado pedidos directos de compra de artículos en existencia, pero sólo a los suscriptores. A diferencia del BBS, el usuario da entrada a "palabras-clave" para especificar su área de interés. Por ejemplo, si deseara utilizar el servicio de compra, debería digitar HELP REWSHOP. También posee una configuración del BBS: dando entrada a la palabra CHALK puede luego dejar un mensaje. Las personas no suscritas al servicio pueden emplearlo: se les permite permanecer ocho minutos en el sistema. Distel y Maptel son servicios similares de base de datos destinados ante todo al pedido de equipos electrónicos e informáticos. La ventaja de estos sistemas comerciales reside en que funcionan las 24 horas del día y no se encuentran ocupados con tanta frecuencia como ocurre con los BBS.

Mientras que los grandes ordenadores de unidad principal han tenido acceso al teléfono desde hace algún tiempo, sólo recientemente, con el descenso de los precios de los modems y la creciente sofisticación de los microordenadores, este tipo de comunicación se ha puesto al alcance de los usuarios de ordenadores personales. En los próximos años, es probable que el modem se convierta en un accesorio tan corriente para el ordenador personal como la impresora o la grabadora de cassette.

Editores de pantalla

La mayoría de los microordenadores permite editar programas en la pantalla, ahorrando tiempo y esfuerzo

Todo el mundo comete errores al utilizar el teclado de un ordenador, y por esa sencilla razón es necesario disponer de medios de edición. Son muchas las situaciones en que sería deseable modificar los datos visualizados, no sólo con la finalidad de corregir un error de digitación o rectificar una sentencia errónea, sino de actualizar información.

Muchos programas aplicativos incluyen alguna forma de "editor" especializado, como la posibilidad de eliminar oraciones, trasladar párrafos a otros lugares y reemplazar un nombre o una frase por otros siempre que aparezcan.

No obstante, casi todos los ordenadores personales poseen algún tipo de editor (incorporado en su sistema operativo en ROM), preparado para editar listados de programas. Los programas son sumamente proclives a los errores. En todo programa se producen errores de sintaxis con gran regularidad y, casi con toda certeza, en su funcionamiento habrá defectos que será necesario eliminar, por no hablar de las mejoras que fuera preciso realizar con posterioridad. Las configuraciones que ofrezca el editor de un ordenador pueden suponer una enorme diferencia en cuanto al tiempo de desarrollo de un programa largo. Debemos recalcar, no obstante, que un buen programador invierte una considerable cantidad de tiempo en verificar el funcionamiento de su programa sobre el papel antes de proceder a digitar el listado en un ordenador. Digitar la primera solución que a uno se le pase por la cabeza es un pésimo procedimiento de programación, pues luego se ha de destinar el 90 % del tiempo de desarrollo del programa a eliminar errores.

Existen dos clases de editores: los "editores de pantalla" y los "editores de línea". Los primeros son considerablemente más flexibles y fáciles de utilizar, pero los segundos están mucho más difundidos entre los ordenadores personales. Los editores de línea provienen de la época en que todos los cálculos se efectuaban en un ordenador remoto a través de teletipos o terminales. Los teletipos tenían una memoria buffer de sólo una línea de 80 caracteres (80 bytes). El programador podía obtener una lista impresa de todo el programa digitando LIST, pero si, por ejemplo, se necesitaba efectuar una corrección en la línea 120, se debía volver a digitar entera otra vez esa línea. En algunos sistemas, digitar EDIT 120 resultaría en una salida impresa de esa línea determinada, pudiéndose entonces introducir modificaciones o supresiones mediante las teclas de "retroceso" y "borrar" (las inserciones seguían siendo imposibles). Se agregaron otras órdenes como DELETE (una escala especificada de números de línea), pero aún existía la limitación de tener que llamar y modificar una línea entera.

Los editores de muchos ordenadores personales aún funcionan como si sólo tuvieran un buffer de una línea cuando, de hecho, toda la pantalla es un mapa de memoria: cada localización de carácter corresponde a un byte de memoria.

Un editor de pantalla es mucho más eficaz. Con él se pueden desplazar texto o gráficos por la pantalla con gran comodidad. Cuando se pulsa RETURN, el editor lee al intérprete toda la línea sobre la cual se halla el cursor, aquél la ejecuta (si la línea consiste en una orden) o le da entrada en el programa (si comienza con un número de línea). Empleando las cuatro teclas con flechas, el usuario puede desplazar el cursor hasta cualquier punto del programa, tal como éste se visualice, y después insertar, eliminar o volver a escribir caracteres como desee.

Para obtener la velocidad necesaria, el editor de pantalla se ha de escribir en código de lenguaje máquina y puede incorporar algunas configuraciones verdaderamente útiles. Los mejores editores de pantalla le permitirán «deslizar» un listado hacia arriba o hacia abajo, e insertar o eliminar líneas o caracteres individuales. Algunos disponen de órdenes similares a las de los procesadores de texto, para contabilizar y modificar las instancias en que se produce una determinada serie de caracteres.

Con cada nueva generación de ordenadores los editores ganan en sofisticación y sencillez de uso. Con la introducción del "ratón" (véase p. 296) y del software que imita los procesos manuales para cortar y pegar trozos de texto, se está reduciendo gradualmente el tiempo necesario para editar un listado o un documento en su formato definitivo.

Línea por línea

Las configuraciones de edición de un Sinclair Spectrum son considerablemente mejores que las de la mayoría de los tipos de editores de línea, aunque de ningún modo es tan fácil de utilizar como un editor de pantalla. Para cambiar una línea determinada de un programa, se ha de desplazar la posición normal del señalador (>), que aparece entre el número de línea y la línea en sí, hasta la línea correspondiente, utilizando las teclas arriba y abajo del cursor

```
  5 REM first draft
  6>DIM a(12)
 10 FOR i=1 TO 100
 20 READ a
 30 NEXT i
 40 PRINT "which value"
 50 INPUT v
 60 LET t=a(v)*1.15
 70 PRINT t
 80 FOR j=1 TO 100
 85 LET r=r+a(j)
 90 NEXT j
 95 PRINT "total is ";r
100 DATA 12,24,7.5,90
101 DATA 30,21.5,78,95
102 DATA 12,8,43,0.9
110 STOP
```

```
  5 REM first draft
  6>DIM a(100)
 10 FOR i=1 TO 100
 20 READ a
 30 NEXT i
 40 PRINT "which value"
 50 INPUT v
 60 LET t=a(v)*1.15
 70 PRINT t
 80 FOR j=1 TO 100
 85 LET r=r+a(j)
 90 NEXT j
 95 PRINT "total is ";r
100 DATA 12,24,7.5,90
101 DATA 30,21.5,78,95
102 DATA 12,8,43,0.9
110 STOP
```

Ian McKinnell

Sharp MZ-711

Un ordenador personal con la ventaja de que permite acoplar los periféricos opcionales dentro de su carcasa

El Sharp MZ-711 es una máquina interesante con algunas configuraciones bastante insólitas. En cuanto a construcción, responde al elevado estándar del diseño japonés y da buen resultado. El color es bueno (muy uniforme y brillante) y, aparte de un ligero aspecto borroso al utilizar ciertas combinaciones de colores, produce textos muy legibles.

No obstante, algunos aspectos de su diseño son extraordinarios. El teclado, por ejemplo, podría ser muy satisfactorio si no fuera porque su modalidad es exactamente la contraria de la que uno pudiera esperar: la normal es la de mayúsculas, mientras que las minúsculas se obtienen pulsando las teclas de cambio (SHIFT). Aunque esta disposición, sin duda alguna, es muy cómoda para la programación en BASIC, dificulta notablemente cualquier otra operación Afortunadamente, existe un sencillo procedimiento para hacer reversible la modalidad del teclado de este ordenador: al pulsar las teclas CTRL y E las mayúsculas se transforman en minúsculas, mientras que al digitar CTRL y F se vuelve a la modalidad anterior.

La unidad para cassette del Sharp MZ-711 está concebida como un accesorio opcional que se introduce en la carcasa principal; pero la mayoría de los usuarios la necesitarán. Este sistema es lento pero fiable y todas las cintas se cargan a la primera vez. No obstante, el sistema operativo de la cassette posee varios puntos débiles: sólo existe disponibilidad para los nombres de archivo más elementales y no hay control sobre el motor de la cassette (que hubiera sido muy fácil de incluir). Esto limita la utilidad de la unidad virtualmente a un almacenamiento de un archivo por cinta. Las limitaciones de este sistema son sorprendentes, ateniéndonos a la experiencia de Sharp en otros campos, como el de los equipos de alta fidelidad, en que son habituales sus lotes de cintas controladas por solenoide.

La impresora-plotter incorporada (se trata, nuevamente, de un accesorio opcional que encaja en la carcasa) posee un mecanismo de cabeza de impresión y unidades de lápiz muy frágiles y susceptibles de estropearse. Cuando la máquina no se va a emplear durante un período prolongado, conviene retirar estas últimas y guardarlas. Lamentablemente, son unidades muy pequeñas que se pueden perder con facilidad al quitarlas de la impresora. También tienen cierta tendencia a secarse con rapidez; en

Chris Stevens

El teclado
El teclado del Sharp MZ-711 le confiere a la máquina un aire profesional. Las teclas están correctamente espaciadas, son de tamaño natural, de recorrido total y aptas para el tratamiento de textos. Cinco teclas de función programable, que responden a las teclas de cambio (SHIFT), ofrecen diez funciones y simplifican las operaciones. Los símbolos para gráficos están dispuestos por secuencia lógica, a diferencia de las otras teclas, que responden al formato QWERTY. Ello significa que puede resultar difícil descubrir cuál es la combinación de teclas que producirá el símbolo requerido

cuanto a la calidad de la impresión, a veces no es del todo satisfactoria.

El intérprete de BASIC se carga de la cinta en vez de estar incorporado en la ROM. Gracias a ello la máquina puede utilizar lenguajes alternativos.

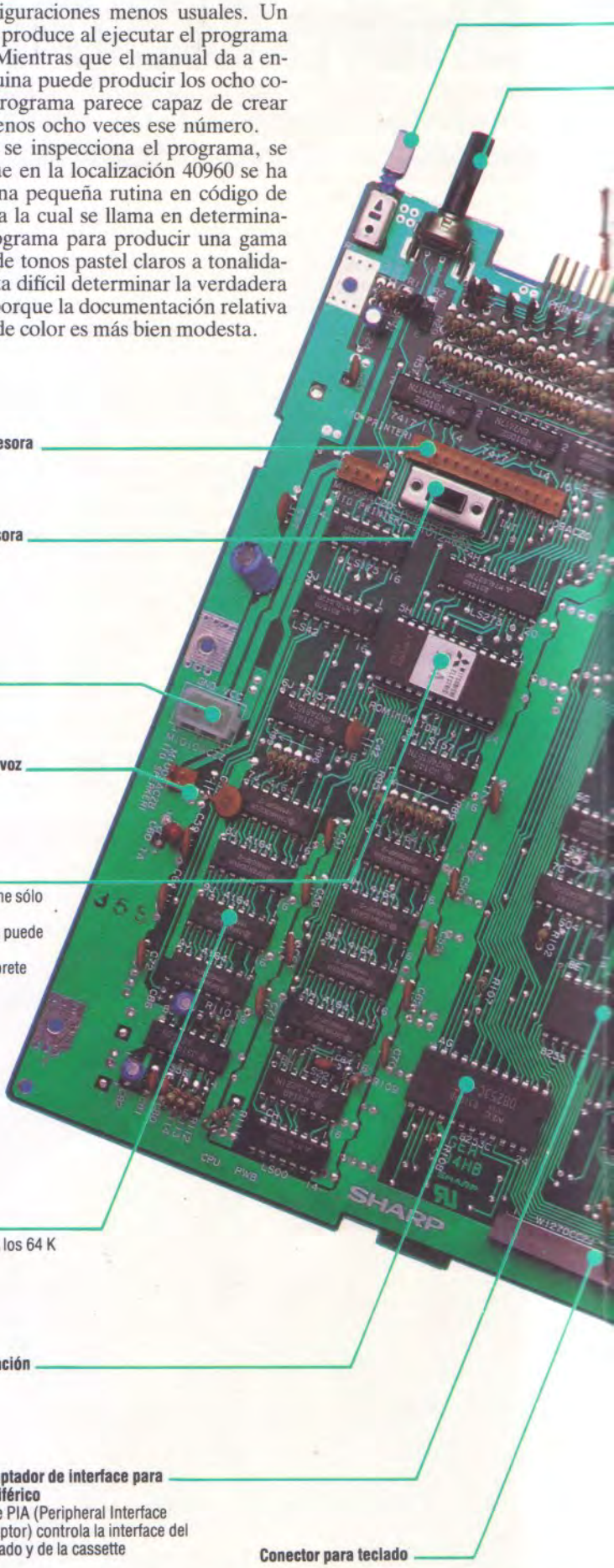
La documentación técnica es de alto nivel. Incluye mapas de memoria y diagramas de circuitos, un listado de lenguaje ensamblador del software incorporado del sistema, así como otros detalles técnicos. No obstante, la documentación carece de diagramas de tiempos, aunque así y todo el manual es mucho mejor que la mayoría de los que acompañan a otros ordenadores.

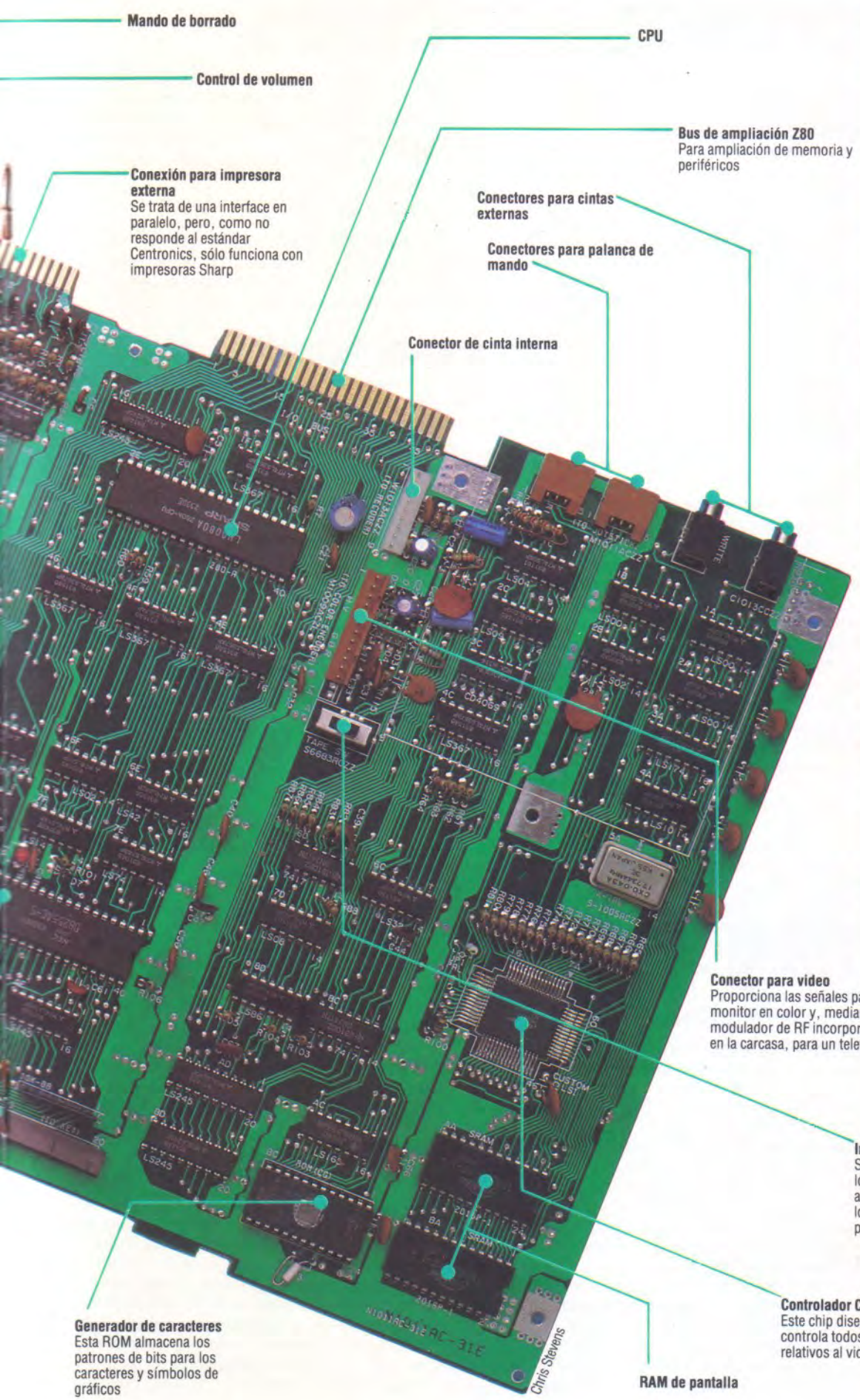
Los fabricantes del Sharp MZ-711 tienen una inexplicable tendencia a restarles importancia a algunas de sus configuraciones menos usuales. Un ejemplo de esto se produce al ejecutar el programa de demostración. Mientras que el manual da a entender que la máquina puede producir los ocho colores usuales, el programa parece capaz de crear una gama de al menos ocho veces ese número.

Sin embargo, si se inspecciona el programa, se podrá descubrir que en la localización 40960 se ha colocado (POKE) una pequeña rutina en código de lenguaje máquina a la cual se llama en determinados puntos del programa para producir una gama de colores que va de tonos pastel claros a tonalidades oscuras. Resulta difícil determinar la verdadera escala de colores, porque la documentación relativa a las posibilidades de color es más bien modesta.

Cassette
La cassette del MZ-711 es un accesorio opcional, si bien se puede incorporar a la carcasa principal. No obstante, el sistema operativo no posee control sobre el motor de la cassette, lo cual limita sus aplicaciones

Impresora-plotter
Al igual que la cassette, la impresora-plotter es un accesorio opcional. Utiliza cuatro lápices esferográficos en miniatura para producir texto y gráficos en un papel de 11,5 cm. Tres modalidades de funcionamiento permiten tres dimensiones de texto y, en consecuencia, 26, 40 u 80 columnas en el papel





Mando de borrado

Control de volumen

CPU

Bus de ampliación Z80
Para ampliación de memoria y periféricos

Conexión para impresora externa
Se trata de una interface en paralelo, pero, como no responde al estándar Centronics, sólo funciona con impresoras Sharp

Conectores para cintas externas

Conectores para palanca de mando

Conector de cinta interna

Conector para video
Proporciona las señales para un monitor en color y, mediante el modulador de RF incorporado en la carcasa, para un televisor

Interruptor de cassette
Si su cassette se negara a cargar los programas, este interruptor alteraría la fase de la señal, con lo cual se debería eliminar el problema

Controlador CRT
Este chip diseñado a medida controla todos los aspectos relativos al video

RAM de pantalla

Generador de caracteres
Esta ROM almacena los patrones de bits para los caracteres y símbolos de gráficos

Chris Stevens

SHARP MZ-711

DIMENSIONES
440 x 305 x 86 mm

CPU
Z80A

VELOCIDAD DEL RELOJ
4 MHz

MEMORIA
ROM, 12 K; RAM, 64 K

VISUALIZACION EN VIDEO
24 líneas de 40 caracteres, o gráficos pixel de baja resolución (50 x 80), 8 colores nominales y un número no especificado de tonalidades, con fijación independiente de fondo y primer plano; 127 caracteres predefinidos y 127 caracteres definibles por el usuario

INTERFACES
RS232 en serie, cassette, bus de sistema, impresora en paralelo

LENGUAJE SUMINISTRADO
BASIC en cinta de cassette

OTROS LENGUAJES DISPONIBLES
PASCAL, ASSEMBLER, FORTRAN, FORTH, DAFMOD, TOOLKIT

VIENE CON
Manuales de instalación y de BASIC, cable para TV, cable para cassette, cassette de BASIC y cassette de demostración

TECLADO
Teclas de recorrido total y cinco teclas de función programable con valores con o sin cambio (SHIFT)

DOCUMENTACION
Es muy amplia y cubre importantes detalles técnicos. Un tanto pesada, constituye, no obstante, una buena introducción a la máquina

Sonidos sencillos

La generación de sonido del Spectrum de Sinclair

El Spectrum se está convirtiendo rápidamente en el ordenador personal más popular. Incorpora excelentes medios para gráficos en color y útiles opciones de memoria por un precio reducido. Lamentablemente, se han sacrificado algunas configuraciones con el fin de no encarecer la máquina. La mayoría de las quejas se refieren al teclado y a la utilización de un BASIC no estandarizado, pero tal vez su mayor debilidad radique en su capacidad para sonido. El Spectrum proporciona los elementos esenciales para la generación de sonido; no obstante, produce música mediante un único oscilador de tipo de "pulso". Se puede controlar la duración de una nota y su altura, pero es imposible alterar el tono de una nota ni modificar su envoltura (véase p. 276). La salida es estándar, y se efectúa a través de un altavoz piezoeléctrico interno muy pequeño, que resta musicalidad al sonido. No obstante, Sinclair ha proporcionado salidas de señal alternativas en las conexiones para cassette *mic* o *ear*, aptas para amplificación externa mediante un sistema de alta fidelidad. La capacidad de sonido del Spectrum ofrece la ventaja de la simplicidad de las órdenes de BASIC BEEP y PAUSE, que están íntimamente relacionadas y que le permiten al usuario comprender con mayor facilidad los principios del sonido generado por ordenador. Además, la máquina posee una escala de extensión muy notable: 10 octavas.

Antes de que se pueda producir algún sonido con el Spectrum, primero es necesario conectarlo a un equipo de alta fidelidad o bien hacer audible el *beeper* interno dando entrada a la siguiente orden directa antes de proceder a ejecutar (RUN) un programa de sonido:

```
POKE 23609,100
```

Por otra parte, es preciso advertir que esta orden también aumenta el volumen de la realimentación "click" que se produce en el momento en que se pulsa una tecla en el teclado.

Control de sonido

Para instruir al ordenador para que dé salida a una nota determinada, se utiliza la orden BEEP del siguiente modo:

```
BEEP d,n
```

donde d representa la duración de una nota y n, la altura de ella. El valor de duración se puede establecer entre 0,00125 y 10 segundos; la altura se expresa como el número de semitonos desde *do mayor* (valor 0) de la escala entre −60 y 69. Por ejemplo, la sentencia siguiente toca la nota *la* en 440 Hz, que está a nueve semitonos de *do mayor*, durante medio segundo:

```
BEEP 5,9
```

Para tocar toda una serie de notas separadas por un espacio medido con exactitud, podemos utilizar la orden PAUSE:

```
PAUSE ms
```

Imágenes elementales

Las capacidades para gráficos del Commodore Vic-20

El Commodore Vic-20, al igual que los otros ordenadores personales de la firma Commodore —el 64 y el PET—, está muy bien construido, pero sus fabricantes no se han prodigado en el juego de instrucciones en BASIC de la máquina. En efecto, el usuario del Vic-20 no dispone de órdenes especiales para gráficos, y para tal fin deberá tener un gran conocimiento acerca del funcionamiento interno del ordenador, o bien tendrá que adquirir alguno de los accesorios diseñados para hacer que la programación de gráficos resulte más sencilla.

El Vic-20 dispone de dieciséis colores y cada cuadrado de caracteres puede contener cuatro colores. La visualización en pantalla consta de 23 filas y 22 columnas de celdas de caracteres de ocho pixels por ocho, pero los caracteres también se pueden visualizar en un formato rectangular de 16 por 8.

Capacidad en baja resolución

Dispone de caracteres alfabéticos en mayúsculas y en minúsculas, así como de más de 60 caracteres para gráficos PET especiales. Muchas de las teclas del Vic-20 llevan marcados dos pequeños cuadrados en su parte anterior, cada uno de los cuales visualiza un patrón determinado. Además de los cuadrados de medio carácter y de un cuarto de carácter, hay símbolos para jugar a los naipes, diseños de tableros de damas, círculos y numerosos símbolos que se pueden combinar en la pantalla para trazar gráficos, componer tablas, producir

donde ms representa el tiempo en unidades de 0,001 segundo (milisegundos). Para tocar una octava en una escala de *do mayor (do, re, mi, fa, sol, la, si, do)* desde el *do* normal durando cada nota medio segundo y dejando, además, una pausa (PAUSE) de un cuarto de segundo entre cada nota y la sucesiva, cabe utilizar este formato:

```
10 FOR I = 1 TO 8
20 READ N
30 BEEP .5,N
40 PAUSE 250
50 NEXT I
60 DATA 0,2,4,5
70 DATA 7,9,11,12
```

Este programa es una buena ilustración del formato de una escala. Una octava se extiende desde la nota base (en este caso *do mayor*) hasta la siguiente nota con el mismo nombre (el siguiente *do* citado) y abarca 12 semitonos. Se denomina octava porque se compone de ocho notas en una escala mayor.

Gary Marsh

do mayor

Escalas y pianos

Utilizando la orden INKEY$ se puede emplear el teclado del Spectrum como aproximación al teclado de un piano:

```
10 REM**************
20 REM *OCTAVA PIANO*
30 REM**************
40 IF INKEY$ = "Q" THEN BEEP 1,0
50 IF INKEY$ = "2" THEN BEEP 1,1
60 IF INKEY$ = "W" THEN BEEP 1,2
70 IF INKEY$ = "3" THEN BEEP 1,3
80 IF INKEY$ = "E" THEN BEEP 1,4
90 IF INKEY$ = "R" THEN BEEP 1,5
100 IF INKEY$ = "5" THEN BEEP 1,6
110 IF INKEY$ = "T" THEN BEEP 1,7
120 IF INKEY$ = "6" THEN BEEP 1,8
130 IF INKEY$ = "Y" THEN BEEP 1,9
140 IF INKEY$ = "7" THEN BEEP 1,10
150 IF INKEY$ = "U" THEN BEEP 1,11
160 IF INKEY$ = "I" THEN BEEP 1,12
170 GOTO 40
```

En un programa como éste se pueden introducir muchas mejoras con el fin de crear un "instrumento" de teclado más eficaz. El Spectrum, si bien sus configuraciones sirven sólo para generar sonidos más bien sencillos, puede constituir, sin embargo, un recurso auxiliar muy útil para enseñar música. Por otra parte, es posible conseguir un sonido de mayor sofisticación si el usuario adquiere hardware adicional para generación de sonido que sea compatible con la máquina.

grandes rótulos y crear otros efectos. Los caracteres se pueden visualizar invertidos (negro sobre blanco, en lugar de blanco sobre negro), aumentando las posibilidades. Con paciencia e imaginación se producen visualizaciones de gran calidad.

Los caracteres se pueden hacer aparecer en la pantalla mediante la utilización de la sentencia PRINT o bien a través de (POKE) los códigos requeridos en la pantalla del Vic-20 y las memorias de color. La versión Commodore de la sentencia PRINT es muy eficaz, ya que permite que el usuario defina individualmente el color de cada carácter. Desde dentro de la sentencia PRINT también se puede controlar el movimiento del cursor para producir con facilidad visualizaciones móviles. Colocar (POKE) los caracteres en la pantalla no es tan rápido como imprimirlos (PRINT), pero este procedimiento puede ser muy útil en ciertas circunstancias.

Capacidad en alta resolución

En un Commodore Vic-20 sin ampliar se pueden obtener gráficos de alta resolución, pero la memoria disponible sólo es suficiente para utilizar la mitad de la pantalla. El proceso por el cual se puede obtener una alta resolución se denomina "mapa de bits", técnica que le permite al programador controlar cada pixel individual dentro de una superficie determinada de la pantalla. Cada celda de caracteres de la cuadrícula de 23 filas por 22 columnas se compone de 64 pixels dispuestos en ocho filas de ocho. Existe una relación matemática entre las coordenadas dadas de un punto de pixel (x,y) y el bit correspondiente en la matriz del carácter. Ésta se puede emplear, junto con una combinación de órdenes POKE, para producir visualizaciones compuestas por pixels individuales.

Cartucho de superampliación

Producir gráficos de alta resolución mediante mapas de bits puede ser un proceso largo y difícil. Una solución alternativa es comprar el cartucho Super Expander de Commodore, que le propociona al usuario órdenes en BASIC para alta resolución, música y color. Las órdenes para alta resolución incluyen GRAPHIC, para establecer la modalidad de visualización, POINT, para trazar un punto de pixel en la pantalla, y PAINT.

La utilización del cartucho entraña dos inconvenientes principales: no existe orden UNPOINT para borrar el punto de pixel, y no se puede pintar (PAINT) a ambos lados de una línea en diagonal sin alterar la línea en sí misma.

Los gráficos de baja resolución del Vic-20 están bien diseñados y son muy flexibles, pero los gráficos de alta resolución resultan difíciles de utilizar sin adquirir un cartucho enchufable.

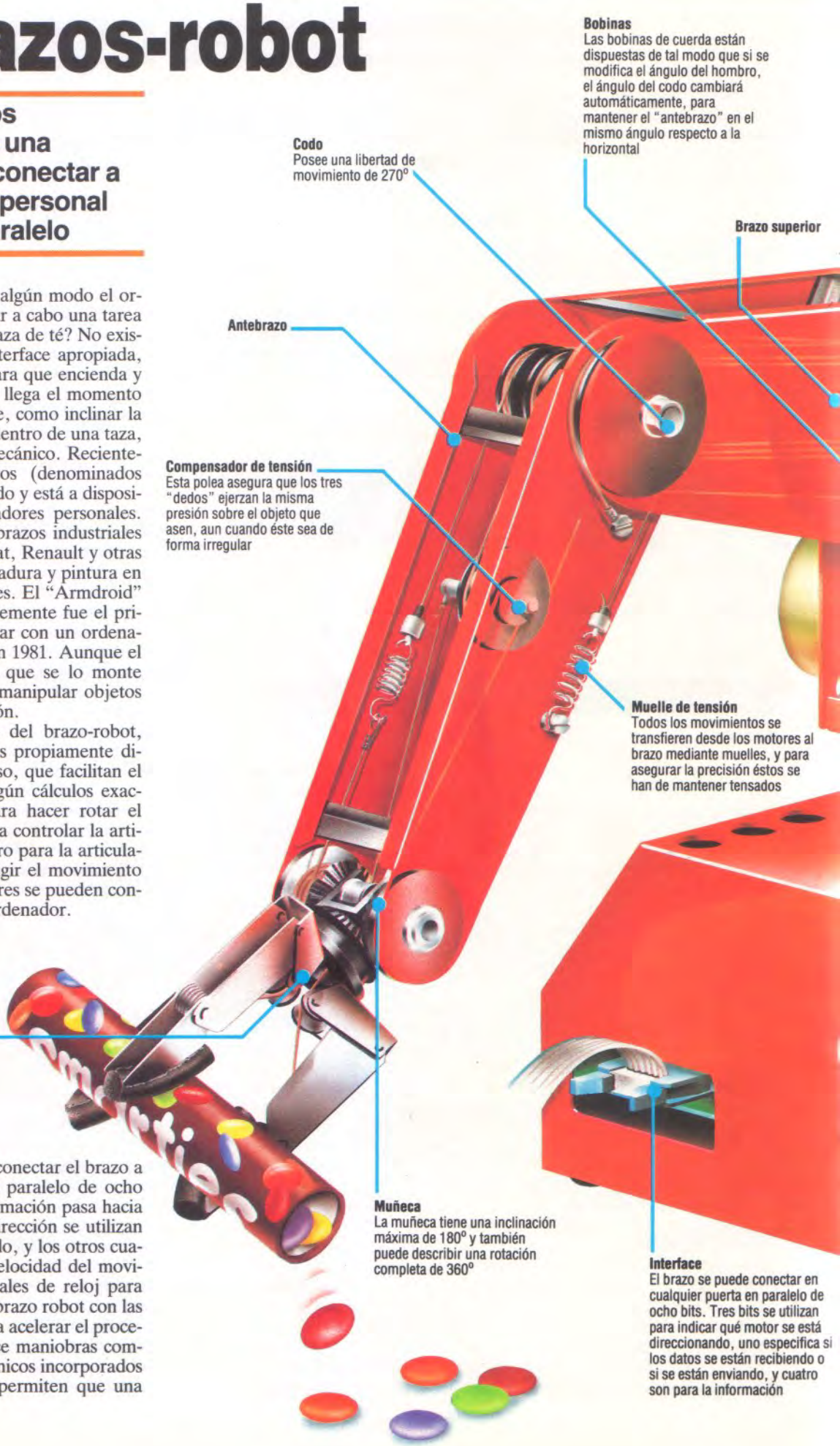
Un conejo programado
Este listado de programa para el Vic-20 demuestra la versatilidad del juego especial de caracteres de Commodore. La figura del conejo está realizada enteramente a partir de este juego de caracteres predefinidos. Resulta más sencillo ver exactamente cómo está construida haciendo mover el cursor por la visualización en pantalla.

Los brazos-robot

Estos precisos brazos mecánicos mediante una interface se pueden conectar a cualquier ordenador personal con una puerta en paralelo

¿Ha deseado alguna vez que de algún modo el ordenador que utiliza pudiera llevar a cabo una tarea tan sencilla como preparar una taza de té? No existe problema alguno, con una interface apropiada, en programar a un ordenador para que encienda y apague una tetera. Pero cuando llega el momento de manipular objetos físicamente, como inclinar la tetera para verter agua caliente dentro de una taza, entonces se necesita un brazo mecánico. Recientemente este tipo de dispositivos (denominados *brazos-robot*) ha salido al mercado y está a disposición de los usuarios de ordenadores personales. Son pequeñas versiones de los brazos industriales que emplean empresas como Seat, Renault y otras para realizar los trabajos de soldadura y pintura en sus cadenas de montaje de coches. El "Armdroid" de Colne Robotics, que probablemente fue el primer brazo-robot apto para utilizar con un ordenador personal, salió al mercado en 1981. Aunque el brazo no es movible (a menos que se lo monte sobre un robot móvil), permite manipular objetos con un notable grado de precisión.

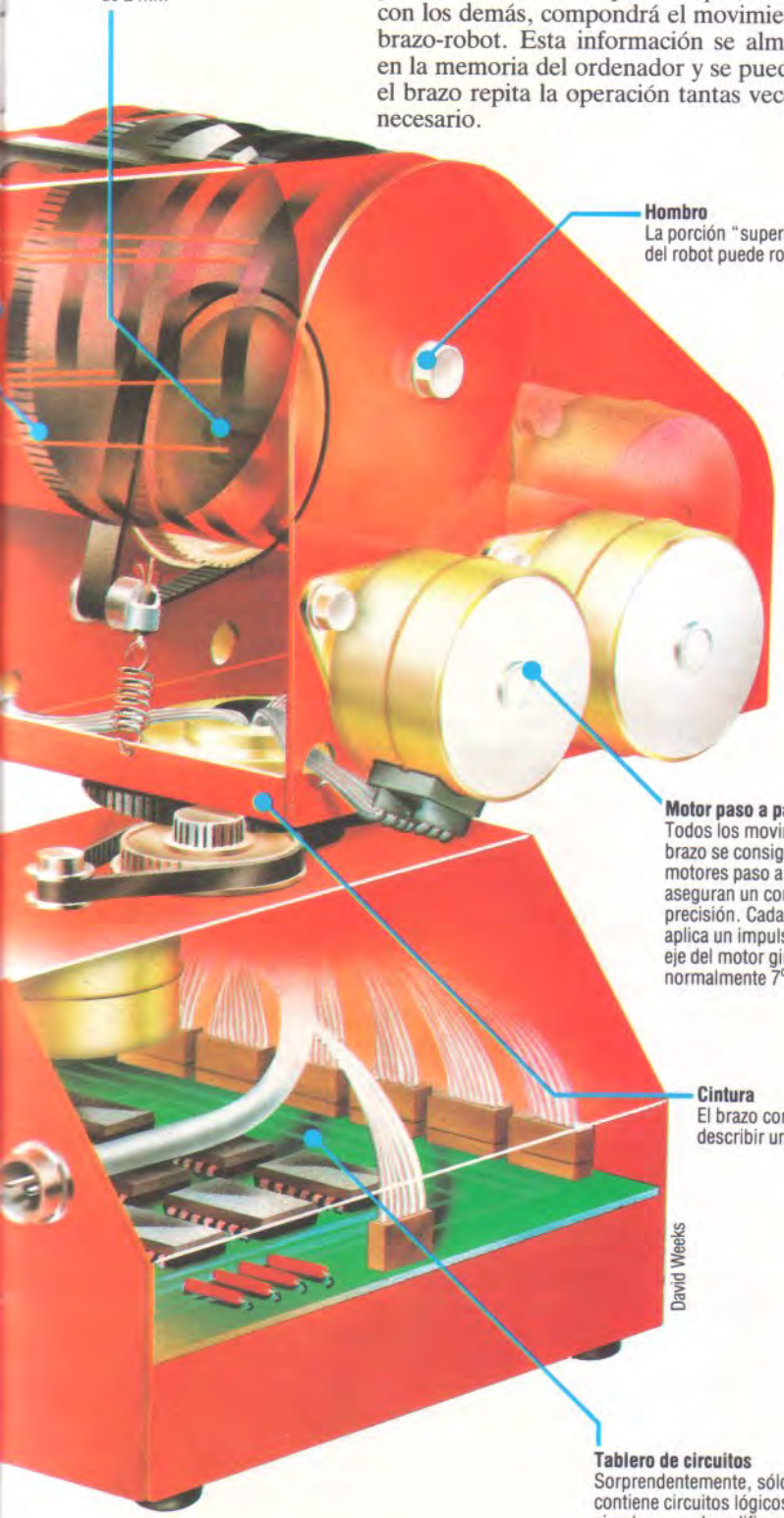
Los principales componentes del brazo-robot, aparte de las secciones metálicas propiamente dichas, son los motores paso a paso, que facilitan el movimiento de las secciones según cálculos exactos. Hay seis motores: uno para hacer rotar el brazo por la "muñeca", otro para controlar la articulación del "hombro", un tercero para la articulación del "codo", y tres para dirigir el movimiento de la "mano". Todos estos motores se pueden controlar fácilmente mediante un ordenador.



Todo lo que se requiere para conectar el brazo a un ordenador es una puerta en paralelo de ocho bits. Un bit determina si la información pasa hacia o desde el robot. Tres bits de dirección se utilizan para seleccionar el motor deseado, y los otros cuatro controlan la dirección y la velocidad del movimiento. También se envían señales de reloj para sincronizar los movimientos del brazo robot con las instrucciones del ordenador. Para acelerar el proceso y permitir que el brazo realice maniobras complejas, hay unos cerrojos electrónicos incorporados en el sistema de circuitos que permiten que una

combinación de motores opere simultáneamente, "reteniendo" la información de un motor mientras se está instruyendo a los otros.

Para hacer que el brazo se coloque en posición y agarre un objeto, primero es necesario dividir el movimiento global en una serie de pasos simples. Cada motor habrá de ser instruido por separado para un movimiento preciso que, conjuntamente con los demás, compondrá el movimiento total del brazo-robot. Esta información se almacena luego en la memoria del ordenador y se puede hacer que el brazo repita la operación tantas veces como sea necesario.



David Weeks

Si el brazo estuviera manipulando objetos delicados (las pruebas se suelen realizar con un huevo), se ha de hacer que el ordenador controle la presión de la pinza. Si ésta es escasa, el huevo caerá; si es excesiva, se romperá la cáscara. Para transmitir la información desde el brazo al ordenador se utilizan diversos procedimientos, pero el más común se basa en microinterruptores sencillos. Éstos se pueden acoplar para que establezcan los límites del desplazamiento del brazo (la mayoría de los brazos de bajo costo no incluyen sensores), o se pueden incorporar en la pinza para detectar un límite de presión preestablecido.

De los sistemas alternativos de interruptores que se emplean en la mayoría de los brazos más grandes, el principal se basa en la detección de presión. La resistencia eléctrica de ciertos materiales se altera cuando éstos se someten a un cambio de presión, y estas fluctuaciones se pueden medir. Aunque este método es más caro, proporciona resultados de mucha precisión.

Si el programa no admite realimentación de información del brazo al ordenador, se dice que es un "bucle abierto" o determinista. Si existe alguna forma de realimentación que ajuste las acciones ejecutadas bajo el control del programa, entonces el sistema es de "bucle cerrado" o estocástico. Aquí los interruptores o sensores de presión se utilizan para limitar el cierre de la pinza en un punto en el cual el huevo se sostiene con firmeza pero sin que se rompa.

Muchos de los sistemas de robot más sofisticados incluyen múltiples sensores para medir la luz, el calor y otras variables. Estos sensores se pueden emplear para llevar el registro de lo que sucede mientras el brazo lleva a cabo su tarea y para informar si algo no funciona bien: por ejemplo, que un robot soldador está practicándose agujeros a sí mismo.

Ian McKinnell

Manejo del lenguaje
Escribir un programa para controlar un brazo robot es relativamente sencillo. En BASIC, la principal tarea sería la de lograr que el ordenador aceptara las órdenes de control provenientes del teclado y que las pasara al brazo a través de la puerta valiéndose de POKE. De la misma manera, se debería poder leer la entrada desde el brazo mediante la función PEEK por la puerta relacionada. Si el principal requerimiento es el relativo a la velocidad, entonces es esencial la programación en código de lenguaje máquina. El FORTH es un lenguaje que ofrece la facilidad de programación del BASIC y gran parte de la velocidad asociada al código de lenguaje máquina. A las secciones de un programa, como las subrutinas o procedimientos, se les dan nombres que se pueden incorporar al juego de órdenes del lenguaje. Ello consigue que este lenguaje sea sumamente eficaz para aplicaciones especializadas, como lo son los programas de control para brazos-robot.

Ampliación de archivos

Proseguimos con nuestro proyecto de programación dando una mirada a la manipulación de archivos

El programa para la agenda de direcciones que hemos venido desarrollando en los últimos capítulos de nuestro curso de programación BASIC es, en realidad, un tipo de base de datos sencillo y, como tal, implica el concepto de "archivos". Esta palabra se emplea de varias maneras que, aunque ligeramente diferentes, están relacionadas entre sí. Empezaremos por analizarlas con cierta profundidad, de modo que de aquí en adelante podamos utilizar la palabra con mayor precisión.

En la programación de ordenadores, nos podemos hacer la idea de que un "archivo" se asemeja mucho a los archivos de un archivador. Es un conjunto de informaciones relacionadas entre sí que se almacenan juntas. Los ordenadores almacenan los archivos en discos o en cintas magnéticas. A cada "archivo" de información se le otorga un nombre propio para que el ordenador pueda acceder a él cada vez que sea necesario. La información contenida en una cinta de cassette o en un disco flexible puede ser un programa o pueden ser los "datos" que utiliza un programa. Tomando como ejemplo la agenda de direccciones computerizada, la información necesaria consta de dos partes separadas: el programa propiamente dicho y los datos sobre los cuales trabaja el programa. El programa es el conjunto de instrucciones que le permiten al ordenador (y al usuario) manejar la información y trabajar con ella.

Los datos que emplea el programa son el conjunto de registros que contienen la información que usted esperaría hallar en una agenda de direcciones: nombres, direcciones, etc. También incluye cierto tipo de datos que normalmente no están disponibles para el usuario. Éstos son los datos "domésticos" que utiliza el programa para funcionar. Entre este tipo de datos podríamos incluir como ejemplos: las "banderas"; la información relativa a las dimensiones habituales de la base de datos (es decir, el número de registros que contiene); si se ha llevado a cabo o no alguna clasificación desde la última vez que se insertó un registro nuevo; o, posiblemente, una indicación de cuántas veces se ha accedido a un registro determinado o cuántas veces se lo ha impreso. La razón por la cual los datos como éstos (y los datos que componen los registros) se han de tratar separadamente del programa quedará clara apenas intentemos ejecutar el programa.

En capítulos anteriores del curso de programación BASIC hemos utilizado las sentencias READ y DATA como formas de poner datos para que trabajen en un programa. Esto sólo es viable cuando los datos no están sujetos a modificaciones, como el número de días de un mes. Si los datos son susceptibles de modificaciones, el programa puede prepararse para ello en la pantalla y se pueden utilizar INPUT, INKEY$ u otros métodos para traspasarle los datos al programa. Un ejemplo del uso apropiado de esta forma de entrada de datos podría ser un juego de adivinanza de números, en el cual una parte del programa podría asumir la forma:

```
PRINT "ADIVINE EL NUMERO"
INPUT N
IF N<> COMPNUM THEN...
```

Los datos del programa de la agenda de direcciones están sujetos, sin embargo, a considerables modificaciones. En teoría, todos los registros se podrían almacenar en el programa y se los podría leer en matrices apropiadas utilizando sentencias READ y DATA. Pero después, a todos los datos de que constaran los registros se les habría de dar entrada como parte del programa. Cada vez que se efectuaran cambios (agregar o eliminar nombres y direcciones, por ejemplo) sería necesario introducir considerables alteraciones en el propio programa. Como mínimo, ello significaría imprimir el programa, verificarlo para determinar dónde sería necesario introducir cambios, escribir nuevos segmentos del programa y luego digitarlos. No obstante, el mayor problema residiría en que los segmentos nuevos del programa no serían módulos completos de programa que se pudieran probar de manera independiente; los cambios estarían diseminados al azar a lo largo de todo el programa. La única forma de saber si el programa modificado funciona correctamente sería ejecutarlo y ver qué sucede.

Por suerte, nada de esto es necesario porque los datos se pueden almacenar independientemente del programa. Esto se consigue creando archivos de datos en la cassette o en el disco. Estos archivos son conjuntos de registros que se tratan de forma muy similar a los datos de una sentencia DATA. El programa es capaz de "abrir" uno o más de estos archivos, leer los datos de ellos (por lo general, en una matriz) y después "cerrar" el archivo. Cuando se necesita introducir un cambio en los datos, el programa abre el archivo adecuado, lee los datos, los modifica y después vuelve a escribir en el archivo los datos modificados.

Con los sistemas de ordenadores basados en disco, localizar un archivo determinado y leerlo o escribir en él es bastante rápido: la localización del archivo se realiza en una fracción de segundo y las operaciones de lectura o escritura por lo general tardan, a lo sumo, unos pocos segundos. Por otra parte, un sistema de ordenador basado en cassette puede ser mucho más lento; además, es probable que el usuario tenga que rebobinar la cinta y esperar a que ésta sea reproducida hasta hallar el archivo correcto. Otra ventaja de la utilización del disco es que se puede tener "abierto" más de un archivo a la vez, mientras que esto no es posible en los sistemas basados en cassette.

Los archivos, entonces, son conjuntos de datos almacenados en un medio de almacenamiento de gran capacidad y son aptos para que los emplee un programa o varios. A un programa de tratamiento

de textos, por ejemplo, se le podría dar acceso al mismo conjunto de nombres y direcciones para que escriba automáticamente cartas "personalizadas".

Los archivos se manipulan de diferentes maneras a tenor de la versión de BASIC que se utilice. La mejor forma de saber cómo los maneja su ordenador consiste en ver lo que dice el manual acerca de las sentencias OPEN y CLOSE ("abrir" y "cerrar", respectivamente) y probar con algunos ejemplos. La descripción que le estamos ofreciendo aquí es muy general e intenta proporcionarle una idea global acerca de la utilización de los archivos.

Los archivos pueden ser secuenciales o de acceso directo. En un archivo secuencial, la información se almacena con el primer dato en primer lugar, el segundo a continuación, en seguida el tercero, etc. Un archivo de acceso directo, por el contrario, está organizado de modo que el ordenador pueda remitirse directamente a la información requerida, sin tener que comenzar por el principio e ir revisando los datos hasta localizar la información requerida. Un archivo secuencial es como ver una película: uno comienza desde el principio y la va mirando hasta el final. En cambio, si usted mirara la película en su casa con un equipo de video, sería en cierta manera como un archivo de acceso directo: podría hacer avanzar o retroceder la cinta y, de esa manera, contemplar exclusivamente la secuencia que le interesara. Nosotros sólo vamos a tener en cuenta los archivos secuenciales, porque son mucho más adecuados para los sistemas de cassette.

Supongamos que desea llevar un registro de la media de las temperaturas diarias durante la semana. Éstas podrían ser:

LUNES	13.6
MARTES	9.6
MIERCOLES	11.4
JUEVES	10.6
VIERNES	11.5
SABADO	11.1
DOMINGO	10.9

Para mantenernos en un nivel simple, todos estos datos se tratarán como datos numéricos, con el lunes como día 1 y el sábado como día 7. Entonces la información se puede representar así:

```
1,13.6,2,9.6,3,11.4,4,10.6,5,11.5,6,11.1,7,10.9
```

Para almacenar todos estos datos en un archivo secuencial, en el programa serán necesarios los siguientes pasos:

```
OPEN el archivo
Escribir los datos en el archivo
CLOSE el archivo
```

Siempre que se utilice la sentencia OPEN será necesario aclarar si vamos a escribir en el archivo datos del ordenador (una salida) o si vamos a leerle al ordenador datos del archivo (una entrada). En el BBC Micro, esto se realiza utilizando las sentencias OPENOUT y OPENIN. Sus equivalentes en BASIC Microsoft son OPEN "0" y OPEN "I". Un breve fragmento de programa para escribir los datos anteriores en un archivo (en BASIC Microsoft) podría ser:

```
100 OPEN "0", # 1, "DAT.TEMP"
110 PRINT # 1,1,13.6,2,9.6,3,11.4,4,10.6,5,
    11.5,6,11.1,7,10.9
120 CLOSE # 1
```

La palabra OPEN de la línea 100 hace que el archivo quede disponible para el programa. OPEN va seguida de "0" para indicar que los datos saldrán del programa para ser almacenados en el archivo. Luego sigue # 1, que le dice al ordenador que en nuestro programa nos referiremos a éste como el archivo número 1. A cada archivo se le da un número convencional que desde ese momento se utilizará con las sentencias INPUT # o PRINT # cuando deseemos leer o escribir datos en él. Por último, tenemos el nombre del archivo entre comillas dobles. Hemos denominado DAT.TEMP a nuestro archivo para indicar que contiene temperaturas y no es un programa sino un archivo de datos.

A continuación escribimos un programa completo en BASIC Microsoft para dar entrada a los datos en un archivo y posteriormente leerlos e imprimirlos:

```
100 OPEN "0", # 1, "DAT.TEMP"
110 PRINT # 1,1,13.6,2,9.6,3,11.4,4,10.6,5,
    11.5,6,11.1,7,10.9
120 CLOSE # 1
130 REM LAS LINEAS 130 y 140 SON LINEAS
    "FICTICIAS" PARA
140 REM REPRESENTAR EL PROGRAMA
    INTERPUESTO
150 OPEN "I", # 1, "DAT.TEMP"
160 FOR X = 1 TO 7
170 INPUT # 1, DIA, TEMP
180 PRINT "DIA";DIA, TEMP
190 NEXT X
200 CLOSE # 1
210 END
```

Éste abre un archivo, numerado # 1 y denominado DAT.TEMP, escribe datos en él utilizando la sentencia PRINT # y luego cierra (CLOSE) el archivo. Más adelante en el programa se abre el mismo archivo utilizando tanto el número del archivo como su nombre (el número no ha de ser necesariamente el mismo que se empleó al crear el archivo, pero el número usado en las sentencias PRINT # o INPUT # debe ser el mismo que se le asignó al nombre del archivo cuando se abrió éste). En la línea 170, INPUT # 1 indica que la entrada provendrá de un archivo numerado # 1 (el archivo DAT.TEMP) y no desde el teclado.

De momento vamos a dejar el tema de la manipulación de archivos y volveremos al programa de la agenda de direcciones y a algunos de los componentes incluidos en la subsección INICIALIZACIÓN del programa. Primero, observemos la cantidad de espacio de memoria que se necesita para un único registro en el archivo de la agenda de direcciones (aquí la palabra "archivo" se está empleando en el sentido de base de datos, o sea, conjunto de todos los registros relacionados, y no en el sentido de sistema operativo).

La utilización de campos de longitud fija hace que en cierto modo se desaproveche espacio de memoria, pero simplifica de manera considerable la programación. Si dejamos una línea entera para cada campo, con 40 caracteres por línea, todos ellos se guardarán en una matriz aun cuando la línea estuviera compuesta básicamente de espacios en blanco. No obstante, en algunas versiones de BASIC, cuando las matrices se DIMensionan, cada elemento puede tener hasta 256 caracteres de longitud. El dimensionamiento establece tan sólo el número de

elementos de la matriz, no las dimensiones de cada uno de los elementos.

Si su ordenador posee un BASIC capaz de manipular matrices multidimensionales, se podría utilizar una dimensión separada para cada uno de los campos; pero muchas versiones de BASIC no pueden hacerlo, de modo que investigaremos aproximaciones alternativas. El método más sencillo consiste en emplear una matriz separada para cada uno de los campos. Pero aquí hay una manera de "hacer trampa" si desea utilizar matrices multidimensionales y su BASIC no pudiera manipularlas.

El truco consiste en tratar a todos los elementos que estarían en la matriz multidimensional como si fueran los elementos de una matriz unidimensional. Por ejemplo, una matriz bidimensional con tres filas y cinco columnas se dimensionaría así: DIM A(3,5) y contendría un total de 15 elementos: de A(1,1) a A(3,5). La misma información se podría retener en una matriz subíndice corriente como ésta: DIM A(15). Siempre que una matriz bidimensional fuera referida como A(F,C) (fila,columna), nosotros la sustituiríamos por A((F − 1)*5 + C).

En caso de emplear una matriz separada para cada campo, hemos de decidir cómo DIMensionarlas. La forma más sencilla es utilizar un tamaño de matriz fijo, pero ello limita el número total de registros que podemos almacenar en la base de datos. Una medida más atinada sería establecer las dimensiones de la matriz de acuerdo a cuantos registros haya en uso. No obstante, no todas las versiones de BASIC admiten que las matrices sean todo lo grandes que el usuario desearía. Aun cuando lo hicieran, un elevado número de registros en la base de datos ocuparía muy pronto toda la memoria disponible en el ordenador. He aquí un programa que le permitirá averiguar el número máximo de elementos que admite el ordenador que emplea usted. Sin embargo, muchas versiones de BASIC permiten que en una matriz haya tantos elementos como se desee, exactamente hasta el punto de ocupar toda la memoria disponible. Cada vez que el programa le pregunte "¿QUE TAMAÑO DE MATRIZ?" dé entrada a un valor mayor, hasta que al fin obtenga un mensaje de error. El CLEAR de la línea 100 tiene el efecto de eliminar la matriz al final de cada pasada. Sin esta sentencia, se obtendría un mensaje de error en la línea 30 al volver a dimensionar una matriz.

```
10 READ D$
20 INPUT "¿QUE TAMAÑO DE MATRIZ?";A
30 DIM N$(A)
40 FOR L = 1 TO A
50 LET N$(L) = D$
60 NEXT L
70 FOR L = 1 TO A
80 PRINT L,N$(L)
90 NEXT L
100 CLEAR
110 GOTO 10
120 DATA "MI COMPUTER"
130 END
```

Aun cuando en cada elemento sólo se admitieran 40 caracteres, con cinco campos por registro, y si hubiera 256 elementos apartados para cada matriz, la cantidad de memoria que se necesitaría para retener todos los datos en la memoria principal sería enorme. Así, si se requiriera un byte por cada carácter a almacenar, necesitaríamos 51 200 bytes (5 × 40 × 256 bytes) sólo para los datos. Obviamente, no es práctico ocupar tanta memoria principal con la información, y ése es el motivo por el cual lo más indicado es emplear archivos de datos separados.

Lamentablemente, como ya hemos sugerido, las rutinas para manipulación de archivos pueden resultar algo difíciles de utilizar. Si no deseamos emplear archivos externos, la única alternativa es colocar los datos en una sentencia DATA de modo que siempre esté presente en el programa. No obstante, una vez haya probado con algunos programas cortos tanto para escribir datos en archivos externos como para tomarlos, todo el proceso le resultará mucho más claro y fácil de entender. A modo de ilustración, hemos elegido dos máquinas y dos versiones de BASIC muy diferentes para complementar el corto programa para la temperatura diaria que dimos en BASIC Microsoft.

Estas versiones son para el Sinclair Spectrum y el BBC Micro; ambas difieren considerablemente de nuestro habitual BASIC Microsoft y para obtener detalles relativos a alguna de estas diferencias los lectores han de remitirse a los recuadros "Complementos al BASIC" de anteriores capítulos del curso de programación BASIC.

En la versión del Spectrum, OPEN # y CLOSE # están reservadas para emplear con microdisco. Cuando se utiliza almacenamiento en cassette, se necesitan versiones especiales de las órdenes SAVE y LOAD. La orden SAVE corriente se emplea para almacenar programas y variables de programas en cinta (y datos normales en sentencias DATA). Las matrices se pueden guardar en cinta usando la sentencia SAVE-DATA. Esta asume la forma siguiente:

```
SAVE nombre del archivo DATA nombre de la matriz ( )
```

El nombre del archivo es el nombre dado al archivo (DAT.TEMP, en el programa en Microsoft). El nombre de la matriz corresponde, simplemente, a éste seguido de un par de paréntesis que encierran un espacio en blanco. Para guardar (SAVE) los resultados de la temperatura diaria, primero deberíamos crear una matriz DIMensionada y escribir los datos en ella, empleando sentencias READ-DATA. Para hacer más evidente la diferencia entre el nombre del archivo y el nombre de la matriz, a la matriz la denominaremos c$ y el nombre del archivo será "DATTEMP".

```
10 DIM c$(14)
20 FOR x = 1 TO 14
30 READ c$(x)
40 NEXT x
50 DATA 1,13.6,2,9.6,3,11.4,4,10.6,5,11.5,
   6,11.1,7,10.9
60 SAVE "DATTEMP" DATA c$( )
70 STOP
80 LOAD "DATTEMP" DATA c$( )
90 FOR L = 1 to 14 STEP 2
100 PRINT "DIA";c$(L,c$(L + 1))
110 NEXT L
120 STOP
```

La línea 60 guarda todos los datos de la matriz c$ en un archivo de datos denominado DATTEMP. El programa se detendrá en la línea 70 y deberá rebobinar la cinta. La tecla CONT reanudará la ejecución del programa. La línea 80 invierte el proceso y almacena los datos de DATTEMP en la matriz c$.

El BBC Micro posee una de las versiones de BASIC más sofisticadas de que disponen los ordenadores personales. Admite la programación estructurada con configuraciones tan avanzadas como una construcción REPEAT-UNTIL y "procedimientos". Antes de que se pueda utilizar un procedimiento, primero éste se ha de definir; más abajo veremos cómo se realiza esto en la versión del programa para el BBC. El BASIC de este ordenador define la dirección del flujo de datos en la sentencia de "apertura", usando sea OPENOUT u OPENIN:

```
10 DIM c$(2,7)
20 FOR F = 1 TO 7
30    FOR C = 1 TO 2
40       READ C$(C,F)
50    NEXT C
60 NEXT F
70 DATA 1,13.6,2,9.6,3,11.4,4,10.6,5,11.5,
   6,11.1,7,10.9
80 INPUT "DIGITE S PARA GUARDAR DATOS",K$
90 IF K$<> "S" THEN GOTO 80
100 PROCSAVE: CLEAR: DIM C$(2,7)
110 INPUT "REBOBINE CINTA DE DATOS DESPUES
    DIGITE L",K$
120 IF K$<>"L" THEN GOTO 110
130 PROCLOAD
140 PRINT "TEMP DIA"
150 FOR F = 1 TO 7
160    FOR C = 1 TO 2
170       PRINT C$(C,F);"  "
180    NEXT C: PRINT
190 NEXT F
200 END
300 DEF PROCSAVE
310 X = OPENOUT ("DATTEMP")
320 FOR F = 1 TO 7
330    FOR C = 1 TO 2
340       PRINT # X,C$(C,F)
350    NEXT C
360 NEXT F
370 CLOSE # X
380 ENDPROC
400 DEF PROCLOAD
410 X = OPENIN ("DATTEMP")
420 FOR F = 1 TO 7
430    FOR C = 1 TO 2
440       INPUT # X,C$(C,F)
450    NEXT C
460 NEXT F
470 CLOSE #X
480 ENDPROC
```

Una de las ventajas del BASIC del BBC la constituye el hecho de que las sentencias para manipulación de archivos trabajan igualmente bien tanto para archivos en cassette como para archivos en disco.

Hasta el momento hemos visto cómo se pueden transferir datos de sentencias DATA, a través de matrices, a archivos de datos en cinta (o disco) y viceversa. El próximo paso será retomar el proceso de INICIALIZACION para ver exactamente cuántas matrices se necesitarán, cuántos elementos precisará cada una y en qué puntos del programa se habrán de transferir datos de y hacia ellas.

Ejercicio

Escriba usted un programa con los siguientes componentes:

```
EMPEZAR
   INICIALIZACION
   DAR ENTRADA A DATOS
   ELECCION
      Guardar datos
      Cargar datos
      Salir del programa
FIN
```

INICIALIZACION inicializará las matrices y variables que requiera el programa. Los datos comprenderán, digamos, 15 nombres, a los que se dará entrada desde el teclado mediante indicaciones en la pantalla. ELECCION le proporcionará al usuario un menú en el que le preguntará si desea ¿GUARDAR DATOS?, ¿CARGAR DATOS? o ¿SALIR DEL PROGRAMA? Vea si puede crear una "bandera" en la parte ¿SALIR DEL PROGRAMA? que guarde los datos si, y sólo si, aún no se hubiera efectuado una SAVE.

Complementos al BASIC

VARIABLES

Entre los ordenadores personales más populares, sólo el BBC Micro admite nombres de variables largos, como NOMBRE$. El Spectrum permite usar nombres largos para las variables numéricas, pero sólo nombres de una letra (además de "$") para las variables alfanuméricas. El Dragon 32, el Vic-20 y el Commodore 64 admiten nombres de variables largos, pero sólo son significativos los dos primeros caracteres, de modo que NOMBRE$ es válido, pero se refiere a la misma posición de memoria que NOMB$: las dos palabras tienen iguales los dos primeros caracteres.
En el Oric-1, los nombres de las variables no pueden estar compuestos por más de dos caracteres (primero una letra y luego un número o una letra). El Lynx admite sólo nombres de variables de una letra.

OPEN

En el Dragon 32 se debe utilizar este formato:

```
OPEN"0", # -1, "DATTEMP"
PRINT # -1,1,13.6,2,9.6,3,11.4, etc.
CLOSE # -1
```

y

CLOSE

```
OPEN"I", # -1, "DATTEMP"
INPUT # -1, D, T
CLOSE # -1
```

PRINT #

En el Commodore 64 y en el Vic-20 emplee este formato:

```
OPEN 1,1,2, "DATTEMP"
PRINT # 1,1,13.6,2,9.6,3,11.4, etc.
CLOSE 1
```

y

```
OPEN 1,1,0 "DATTEMP"
INPUT # 1, D, T
CLOSE 1
```

LYNX

El Lynx y el Oric-1, en su forma estándar, no admiten archivos en cassettes. No obstante, en el futuro saldrán al mercado ampliaciones que lo permitirán.

ORIC-1

Tony Lodge

El LEO de Lyons

En Gran Bretaña, la informática comercial tuvo su origen... ¡en una tienda de comestibles!

Oficina electrónica
A diferencia de todos los ordenadores anteriores, que se habían diseñado para aplicaciones científicas o militares, el LEO 1 se creó para efectuar sólo operaciones aritméticas sencillas, pero sobre miles de ítems o transacciones por día

En 1947 se tomó una decisión futurista para intentar construir un ordenador que se pudiera utilizar para automatizar el trabajo de oficina de los dependientes. Sería el primer ordenador de gestión comercial del mundo. Esta imaginativa decisión se gestó en un lugar sorprendente: J. Lyons, una tienda de comestibles. El movimiento comercial de la cadena de tiendas Lyons comprendía una inmensa cantidad de pequeñas transacciones, y para que una organización empresarial de dichas características fuera rentable, era necesario mantener un estricto control sobre todo el trabajo administrativo. A título orientativo, consignemos el dato de que, incluso después de la devastación que para el Reino Unido supuso la segunda guerra mundial, la empresa contaba con más de 1 000 empleados encargados de clasificar los recibos de las tiendas.

Tienda de comestibles computerizada
La tradicional tienda de comestibles Lyons no parece un sitio muy apropiado para la primera gran aplicación comercial de la informática; pero fue precisamente este tipo de negocio, con su considerable número de pequeñas transacciones diarias, el que primero se interesó por los métodos informáticos

Lo cierto es que la firma Lyons tenía ya una larga tradición de inquietud innovadora en los métodos empresariales: ya en 1896 había introducido en sus establecimientos las máquinas de calcular y durante la década de los treinta experimentó con llevar registros de las transacciones en microfilm y fundó el primer centro de investigación empresarial para estudiar los métodos operativos.

Cada cierto tiempo, Lyons hacía viajar al extranjero a algunos representantes para que investigaran los nuevos adelantos que pudieran ser útiles para la empresa, y en 1947 envió a dos empleados a Estados Unidos para que se informaran acerca de los nuevos "cerebros electrónicos". El descubrimiento más útil que efectuaron fue que en su mismo país, en Cambridge, se estaba construyendo un ordenador.

La junta directiva de Lyons encargó un estudio de viabilidad para considerar el posible desarrollo de un ordenador propio para la empresa. El informe sugirió que se podía construir un ordenador por 100 000 libras y que éste supondría un ahorro de 50 000 al año. En consecuencia, en octubre de 1947 Lyons empezó a trabajar en el proyecto. El cometido era muy temerario, porque en aquel entonces el ordenador de Cambridge estaba sólo en la fase de diseño. Lyons le entregó a la Universidad de Cambridge una subvención de 3 000 libras para contribuir a la construcción de lo que se conocería como EDSAC (Electronic Delay Storage Automatic Computer). Dicho dinero se utilizó para adquirir del gobierno un excedente de válvulas. En 1949, el EDSAC concluyó con rotundo éxito su primera tarea: calcular una tabla de números primos.

Lyons analizó los problemas que habría de resolver su ordenador, esbozando las rutinas que serían necesarias. Estos estudios se convirtieron luego en los dibujos de ejecución de los primeros programas y ayudaron a determinar el diseño del hardware. Pero muy pronto se hizo evidente que un ordenador de gestión era fundamentalmente distinto de una máquina de investigación universitaria. Mientras que el EDSAC estaba diseñado para efectuar operaciones matemáticas largas y complicadas a partir de una entrada compuesta por unos pocos números, el ordenador de gestión debía resolver el tipo de problema opuesto. Las operaciones matemáticas eran mínimas (sumar y multiplicar), pero la cantidad de datos a procesar era enorme.

LEO (Lyons Electronic Office) entró en funcionamiento el 9 de febrero de 1954 y se utilizaba para calcular la nómina de pagos semanal de los 1 700 miembros del equipo de personal. Realizaba en un segundo y medio la misma operación que previamente efectuaba un empleado en ocho minutos.

Para Lyons, LEO representó un rotundo éxito y enseguida comprendieron que necesitaban más de una máquina. Otras firmas se mostraron interesadas y Lyons organizó una empresa que se dedicaría, aprovechando la experiencia adquirida, a fabricar y comercializar ordenadores. Leo Computers tuvo un gran éxito y amplió su campo produciendo una serie de versiones mejoradas de LEO que se utilizaron en la industria, en el mundo empresarial y en las oficinas públicas. En 1963 la empresa fue adquirida por la English Electric Company.

Manejables y atractivos

La "ergonómica" es una ciencia cuyo fin es lograr máquinas más agradables de utilizar. En el caso de los ordenadores, la investigación se ha centrado en la pantalla y en el teclado

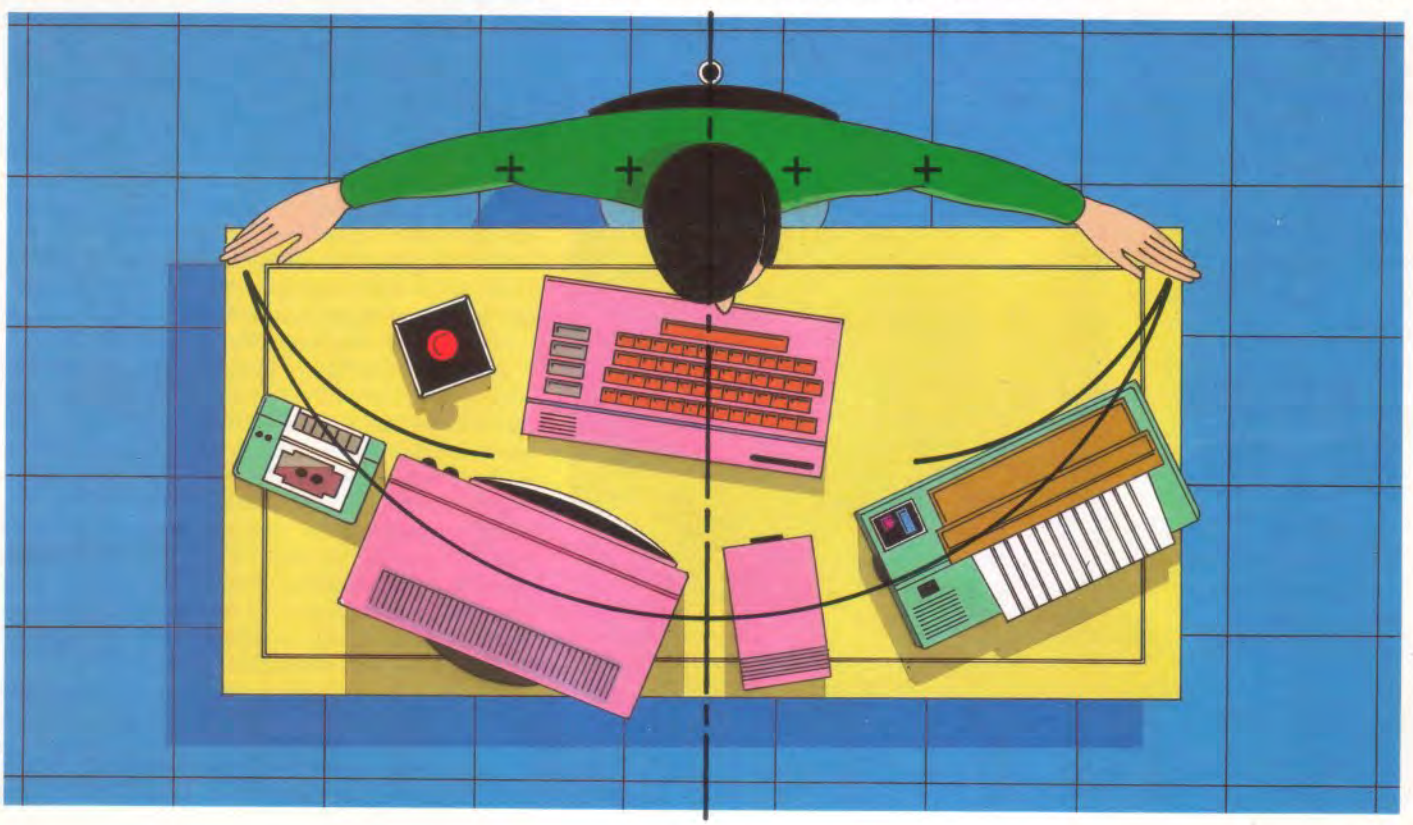
El diseño de una máquina tiene dos facetas: la estética, que se refiere a la belleza formal de su aspecto, y la ergonómica, que se ocupa de la relación existente entre el trabajador y su entorno. Independientemente del hecho de que un aparato funcione bien, no nos sentiremos a gusto usándolo si su aspecto es desagradable. Del mismo modo, el entorno en el cual estemos laborando no debe ser ni incómodo ni perturbador.

Probablemente la calidad ergonómica de una máquina, en cuanto factor a considerar en el momento de decidir qué ordenador comprar, merezca menos atención que su precio y su rendimiento. Sin embargo, vale la pena evaluar el entorno físico en el cual se va a utilizar el ordenador. En primer lugar, ¿trabaja en un sitio con el adecuado espacio libre y sobre una superficie situada a la altura correcta para usted? ¿O simplemente conecta su ordenador personal en el televisor familiar y allí se pone a trabajar, con la máquina en el regazo o, peor aún, tumbado de bruces en el suelo, frente al aparato de televisión?

La programación de ordenadores ya es de por sí lo suficientemente complicada como para que la hagamos más difícil todavía trabajando en un entorno inadecuado. Existen muchas maneras de crearse un centro de trabajo más cómodo. Comencemos por considerar lo que se puede hacer para que resulte más cómodo leer en la pantalla. Si está utilizando un televisor doméstico, entonces no podrá beneficiarse de los últimos avances tecnológicos que ayudan a reducir o que eliminan el brillo de la pantalla de los monitores. Éstos incluyen filtros para reducir al mínimo el reflejo y fósforos de pantalla coloreados especialmente. Pero usted puede mejorar la calidad de la visualización de un televisor colocando un filtro sobre la pantalla. Obtener un filtro coloreado sencillo es fácil y también se puede utilizar un filtro polarizante, que elimina los reflejos. Estos métodos ayudan a lograr un gran contraste a niveles bajos de brillo y, por consiguiente, evitan el innecesario esfuerzo de los ojos.

Los niveles de luz externa también son importantes. Cuando se trabaja de noche es mucho mejor emplear una lámpara de escritorio baja que ilumine el teclado y las notas sobre las que esté ocupado, pero que deje a la pantalla, en comparación, sumida en mayor oscuridad. La distancia desde los ojos a la pantalla también es importante; el espacio adecuado entre el cuerpo y la pantalla corresponde aproximadamente a la longitud de un brazo extendido. La visualización en sí misma debe ser inclinable, de manera que el plano de la pantalla esté a 90° respecto a la línea imaginaria que va desde su vista

El lenguaje del cuerpo
Aunque el cuerpo humano varíe en cuanto a forma y tamaño, las proporciones son siempre muy constantes, tal como comprenden enseguida los estudiantes de dibujo corporal. La ergonómica se vale de esta consecuencia lógica para definir las reglas generales del trazado de entornos de trabajo. En el caso de un ordenador personal o de unidad de representación visual, estas reglas indican que la pantalla ha de estar a una distancia equivalente a la longitud del brazo (para aminorar los cambios en la distancia focal del ojo al dirigir la vista desde la pantalla a la fuente de consulta, y viceversa). La posición del teclado obedece también a estas mismas reglas



Roy Ingram

hasta el centro de la misma. Esto se puede conseguir con facilidad colocando uno o dos libros bajo la parte delantera del aparato. No obstante, en este punto es posible que se encuentre con otro problema: su reflejo en la pantalla. Este inconveniente se puede superar de manera eficaz colocando un filtro de superficie mate.

Una vez que se ha conseguido que la pantalla resulte cómoda de utilizar, podemos pasar a considerar el trazado y las características físicas del teclado. Los factores más importantes son la altura de las teclas por encima del escritorio donde está colocado el teclado, y el ángulo de las filas de teclas entre sí. Lo ideal sería que el teclado fuera lo suficientemente bajo como para que las muñecas y los antebrazos del operador pudieran descansar sobre el escritorio, frente a él, para lo cual debería ser regulable. Lamentablemente, son muy pocos los microordenadores personales diseñados con el perfil bajo requerido. Las series ZX de Sinclair, el Oric-1 y el Jupiter Ace constituyen excepciones en este sentido, pero todos ellos plantean problemas todavía mayores con sus teclados porque, en vez de las teclas de muelle, presentan membranas de capas múltiples o láminas de plástico moldeado. Las membranas de capas múltiples no poseen sensación táctil y, en el caso del ZX80 y del 81, están espaciadas de tal modo que se hace difícil escribir al tacto con ellas. La conjugación de estas características hace que el dar entrada a programas largos sea una tarea agotadora. El Oric-1 y el Spectrum intentan subsanar este inconveniente mediante la producción de una señal audible que indica que la tecla se ha pulsado lo suficiente como para hacer contacto. Pero esta configuración dista mucho de constituir una compensación adecuada. Existen varias empresas que proporcionan teclados alternativos (a escala natural, con teclas de muelle) para los ordenadores Sinclair, pero los modelos bien diseñados son caros. También mantienen la convencional característica de poseer una entrada de tecla única —ideada por Sinclair para acelerar la operación en BASIC— bastante incómoda para el usuario.

El trazado ideal de un teclado requiere que las filas de teclas, miradas desde un costado, estén dispuestas como si formaran parte de la circunferencia de un tambor. Ello reduciría al mínimo el movimiento direccional de los dedos del mecanógrafo. Los únicos ordenadores personales que responden a esta especificación son el BBC Micro, el Commodore 64 (así como los últimos Vic-20) y el Apple II.

Martin Burke

La máquina estenográfica
Cuando hay necesidad de grabar la voz y al taquígrafo no le es posible lograr que el orador hable con mayor lentitud, a menudo se emplea un dispositivo denominado Palantype. Las máquinas de este tipo utilizan una versión estenográfica de la grafía fonética

El trazado del teclado viene siendo desde hace ya mucho tiempo la manzana de la discordia de los diseñadores. Cuando aparecieron por primera vez las máquinas de escribir, en el siglo XIX, había tantos trazados de teclado diferentes como fabricantes, pero en general las teclas de los caracteres que se utilizaban con mayor frecuencia estaban agrupadas en el centro del teclado. Cuando se introdujo la *typebasket* (máquina de escribir con las palancas de letras dispuestas en forma de cesta) en la década de 1870, los fabricantes descubrieron que hasta a los más lentos mecanógrafos se les podían trabar y enredar entre sí las palancas de letras. El problema se presentaba con mayor frecuencia cuando palabras como *ten* (formada por tres de las letras más empleadas en inglés, que se hallaban convenientemente situadas una junto a la otra en el teclado) se utilizaban en rápida sucesión. La solución adoptada consistió en desplazar aquellas letras que con más frecuencia se encontraban junto a otras en las palabras a un lugar más alejado de la *typebasket*; y así nació el teclado QWERTY, hoy estandarizado, que diseñaron Scholes y Gliden en Estados Unidos. No existe razón alguna por la cual un teclado electrónico haya de ceñirse a este trazado, como no sea para conservar una característica convencional; éste es un ejemplo de un estándar universal *de facto* que se está volviendo poco deseable y que, no obstante, es imposible modificar. Sin embargo, se han realizado algunos esfuerzos por crear teclados alternativos. En 1977, Lillian G. Malt empleó la flexibilidad inherente del hardware electrónico para producir un teclado moldeado para adaptarse a la mano, que resulta mucho más cómodo de utilizar que los de diseño estándar. Su funcionamiento también es mucho más rápido: es posible pulsar 300 palabras o más por minuto. Lamentablemente, no ha logrado imponerse al dominio completo que la disposición QWERTY posee en el trazado del teclado de los ordenadores.

Este teclado (que se llama Maltron) tiene una característica muy útil, que comparte con otros microordenadores: es desmontable. La mayoría de los ordenadores personales no poseen monitor incorporado y son lo suficientemente pequeños como para ser trasladados, pero esto no sucede con muchos microordenadores creados para ser utilizados en oficinas. Sin embargo, poco a poco se va imponiendo el diseño de teclados lo más delgados posible y que se acoplan al microordenador mediante un "cordón umbilical". El PC Junior de IBM ha dado otro paso hacia adelante: la conexión para comunicaciones entre el teclado y el microordenador es similar a los controles remotos de los televisores y videos y funciona mediante luz infrarroja.

Debido a que la ergonómica no es una ciencia totalmente objetiva (es el estudio de la forma en que los trabajadores *se relacionan* con su entorno laboral, y esa relación tiende a cambiar de cuando en cuando), no es posible dictar reglas estrictas y expeditas. Su principio fundamental es lograr un confort a largo plazo. Ello requiere que las herramientas y el equipo estén dispuestos de tal modo que uno pueda dedicar toda la energía a la tarea que tiene entre manos, sin que sea necesario cambiar de posición constantemente ni llegar a un grado excesivo de cansancio.

Existen varias posibilidades más que el usuario de un ordenador personal debe considerar y explorar con el fin de mejorar su entorno de trabajo. Cuando analizamos el Lisa de Apple (véase p. 261) observamos que al trabajar con un software activa-

Los dos teclados
Antes de que se desarrollaran los teclados electrónicos, cada tecla de la máquina de escribir se había de conectar físicamente a la diminuta pieza fundida con la forma del carácter. Esto suponía limitaciones en cuanto al trazado del teclado, ya que era esencial mantener separadas las teclas más utilizadas para que las palancas portadoras de los caracteres no chocaran entre sí. Aunque ya no es necesaria esta providencia, aún se conserva el trazado QWERTY tradicional. Los teclados como el Maltron, en el que las teclas están dispuestas de acuerdo a su frecuencia de uso, no han logrado hacerse populares

do por menú había ciertas alternativas para el teclado. Quizá le interese probar con una versión de este tipo de software, utilizando una palanca de mando o un mando de bola, y juzgar por sí mismo las ventajas. Por supuesto, necesitará escribir algunos pequeños programas con los cuales trabajar, pero empleando PEEK y POKE dentro de los límites de la memoria de pantalla no sería tarea difícil.

Por otra parte, si el ordenador que emplea permite que se vuelva a especificar el valor de una tecla determinada, podría cambiar la disposición del teclado, pegando etiquetas sobre las teclas para indicar sus nuevos valores. En este caso quizá fuera más fácil examinar (PEEK) el valor de los ocho bytes que componer el carácter en una matriz con ocho variables, cambiar los valores dentro de la matriz y después volverlos a colocar (POKE). A través de la orden POKE, podría almacenar los ocho bytes que componen el carácter directamente en el espacio destinado al carácter que desea reemplazar, pero si utiliza este procedimiento no olvide guardar el primer juego de valores en una matriz temporal y después desplazar en orden cada carácter a su nueva posición. Guarde en cassette el programa que efectúa esta operación, porque cuando apague el ordenador (o lo restaure), ¡cada carácter recuperará su valor original!

Por último, si sus intereses también se orientan hacia la carpintería, podría considerar la posibilidad de construirse un centro de trabajo diseñado a la medida con el teclado dispuesto en la parte superior y el televisor o el monitor situados en un ángulo apropiado. Las versiones comerciales de los centros de trabajo suelen proporcionar espacio adicional para el almacenamiento auxiliar (unidades de disco o cassette) en estantes situados bajo la mesa. La ergonómica se podría definir como el sentido común aplicado; si usa algunas de las sugerencias reseñadas, apreciará una significativa disminución de los dolores de cabeza y la fatiga visual.

Alternativas para el futuro

Muchos diseñadores de ordenadores prescindirían por completo de los teclados si les fuera posible. Los microordenadores más recientes, con más memoria y mayores velocidades de procesamiento, permiten utilizar, en sustitución, otros dispositivos, como palancas de mando, mandos de bola y "ratones", con el software adecuado

Kevin Jones

Sistema operativo

La función del sistema operativo de disco es no perder de vista dónde está todo cuanto se conserva en el disco. Sin él, la programación sería una tarea muy difícil

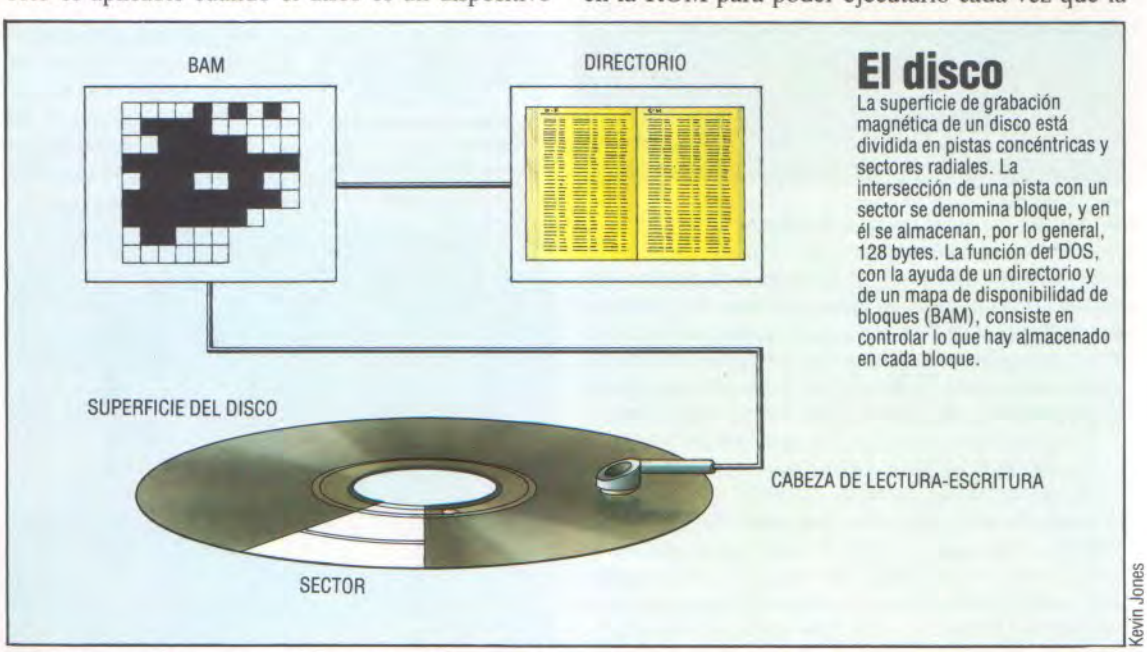
Antes de que un ordenador sea capaz de ejecutar cualquier clase de programa aplicativo, necesita su propio conjunto de programas internos con los cuales administrar los diversos componentes de su sistema e interpretar las instrucciones de que consta el programa del usuario. Este conjunto interno de programas se denomina *sistema operativo* (*Operating System*: OS), y en la mayoría de los ordenadores personales reside con carácter permanente dentro del ordenador, bajo la forma de memoria ROM.

Si su sistema incluye una unidad de disco, entonces gran parte de ese OS estará dedicada a las diversas operaciones del disco. A este conjunto de rutinas lo denominamos *sistema operativo de disco (Disk Operating System)* o DOS. Es posible que haya visto aparecer estas tres letras en los nombres de productos patentados; el sistema operativo de Microsoft, por ejemplo, se llama MSDOS. Un DOS se presenta, por lo general, en tres diferentes formas. En la primera, puede ocupar parte de la ROM interna del ordenador. Un ejemplo de ello es el Sinclair Spectrum, que lleva incorporadas las órdenes para hacer funcionar el Microdrive.

En una segunda forma, el DOS se almacena en ROM dentro de la propia unidad de disco. Esto sólo es aplicable cuando el disco es un dispositivo "inteligente" (como la unidad de disco Commodore), es decir, cuando incorpora su propio microprocesador de ROM y RAM. La fabricación de estas unidades de disco es mucho más cara, pero ofrecen considerables ventajas respecto a las unidades de disco "no inteligentes". Por ejemplo, no llegan a agotar la valiosa memoria del usuario y pueden ejecutar una compleja operación en disco mientras el ordenador prosigue con el programa aplicativo.

En tercer lugar, el DOS puede residir dentro de la RAM del ordenador. Esta técnica se está popularizando cada vez más en los sistemas de gestión empresarial, en los cuales las unidades de disco están incorporadas en el ordenador y la cantidad de RAM disponible es enorme (digamos que el estándar es de más de 128 Kbytes). Desde el punto de vista del fabricante, esto tiene la ventaja de que elimina la necesidad de crear un juego de unidades ROM completamente nuevo cada vez que se modifique el DOS aunque sea en una mínima parte, y el usuario se beneficia al poder escoger un sistema operativo entre los muchos que están patentados y que funcionan con el mismo hardware.

Pero ¿cómo se introduce el DOS en la RAM en primer lugar? Esta pregunta surge de inmediato cuando se enciende el sistema. El DOS ha de transferirse del disco a la RAM, pero si en el ordenador no hay DOS que le diga cómo controlar el disco, ¿cómo podrá cargar algo en la RAM? Un programa no se puede introducir por sus propios medios dentro de la RAM, de modo que el ordenador ha de contar con un pequeño programa incorporado en la ROM para poder ejecutarlo cada vez que la máquina se encienda. Este programa se denomina *bootstrap* y es, en sí mismo, una forma muy sencilla de DOS. La función del bootstrap consiste simplemente en hallar el DOS principal en el disco y transferirlo byte por byte a la RAM, donde ese DOS podrá tomar a su cargo y llevar a cabo algunas



El disco

La superficie de grabación magnética de un disco está dividida en pistas concéntricas y sectores radiales. La intersección de una pista con un sector se denomina bloque, y en él se almacenan, por lo general, 128 bytes. La función del DOS, con la ayuda de un directorio y de un mapa de disponibilidad de bloques (BAM), consiste en controlar lo que hay almacenado en cada bloque.

Kevin Jones

funciones mucho más sofisticadas. Este proceso de encender el ordenador y esperar después a que el DOS se haga cargo, se denomina *booting up*. Una vez concluido, se imprime un saludo en la pantalla junto con un aviso que indica que el ordenador está preparado para recibir una orden del usuario.

Sea cual fuere la forma que asuma el DOS en un sistema, su principal función es la de buscar las localizaciones del contenido del disco. El lector recordará que un disco (véase p. 114) se divide en ocho anillos concéntricos, denominados pistas, que a su vez se dividen en sectores; y la intersección de una pista con un sector se denomina bloque. Normalmente un bloque retiene hasta 128 bytes de información y constituye la unidad más pequeña que el disco puede leer o escribir a la vez. Para permitirle al ordenador recordar la localización exacta de todo cuanto contiene el disco es indispensable disponer de un DOS. Esta tarea es mucho más abrumadora de lo que parece a primera vista. Supongamos que nuestra unidad de disco posee una capacidad de 320 Kbytes, suficiente para almacenar 20 programas de 16 Kbytes cada uno. Reteniendo cada bloque 128 bytes, para cargar uno de estos programas sin la ayuda de un DOS, el usuario tendría que especificar 128 bloques diferentes, ¡cada uno de ellos con su número de pista y de sector!

Para poder cumplir esta función, el DOS lleva un directorio del disco. Éste, para facilitar su lectura, suele estar situado en la pista central del disco, de esa manera se reduce la distancia a la cual ha de moverse la cabeza de lectura-escritura. La velocidad de funcionamiento de un disco depende más de lo que tarda la cabeza en desplazarse de una pista a otra que de la velocidad a que gira el disco.

El directorio

Nombre del archivo	Tipo	Localización (pista-sector)
Invasores	Progr.	20-1,20-7,20-2...
Temper.	Progr.	25-11,26-5,26-12...
Presup.	Progr.	23-12,24-3,24-9...
Datpresup.	Datos	27-1,27-7,27-2...

El directorio de un disco normalmente ocupa la pista central. Contiene una lista de los nombres de los archivos, su tipo (programación, datos u otras categorías) y los números de la pista y del sector donde se encuentra almacenado el archivo

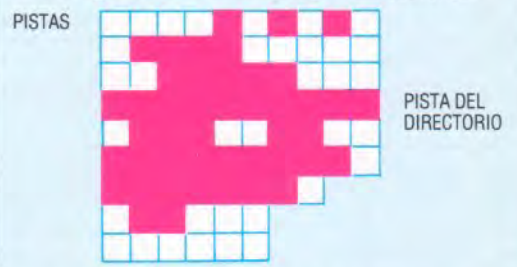
Se denomina *directorio* a la lista de todos los archivos (que pueden ser de programas o de datos) existentes en el disco; en él se detalla el nombre del archivo, a qué tipo corresponde, y se proporciona una relación de los bloques (cada uno de ellos especificado mediante pista y sector) donde se encuentra almacenado. Puede haber algunas otras entradas, como la fecha en que se hizo la última copia del archivo o una lista de los usuarios que pueden tener acceso a un archivo determinado.

Cuando se ha de almacenar un archivo nuevo, el DOS debe primero consultar lo que se conoce como lista de sectores libres o mapa de disponibilidad de bloques (*Block Availability Map*: BAM). Éste posee un único bit correspondiente a cada uno de los bloques del disco, y cuando se emplea un bloque el valor de su bit se cambia de cero a uno. Algunos ordenadores personales con unidades de disco incorporan un programa de utilidad que visualiza el BAM en la pantalla y el usuario puede ver las entradas que se van efectuando mientras guarda un programa. Cuando se borra un archivo, el DOS no necesita limpiar todos los bloques que se utilizaban para ese archivo; simplemente, cambia las entradas en el BAM para indicar que el contenido de aquellos bloques ahora ya no interesa.

Este sistema tiene, además, otra característica: los archivos no se almacenan, como sería de esperar, en bloques colindantes consecutivos. Supongamos, por ejemplo, que una pista consta de 12 sectores, numerados del 1 al 12 en el sentido de las agujas del reloj. Pues bien, los primeros 128 bytes de un programa podrán hallarse en el sector 1, los segundos en el sector 7, los terceros en el sector 2, y así sucesivamente. Ello se debe a que transcurre un breve lapso mientras el contenido de un bloque se transfiere al buffer de memoria que se utiliza para escribir cada bloque. Si el DOS tuviera que escribir sectores consecutivos, entre cada escritura debería esperar una revolución completa del disco, con lo cual el sistema se retardaría. Además, un disco que haya estado en uso durante algún tiempo, con archivos cuya longitud cambiará día a día, acabará por tener un BAM parecido a un trozo de queso de Gruyère, y los archivos nuevos habrán de acomodarse en los agujeros.

Un sistema operativo de disco posee muchas otras funciones, incluyendo dar formato a discos nuevos (marcando las pistas y los sectores en un disco en blanco y creando un directorio vacío), hacer copias *black-up* y "ordenar" los discos llenos. Versiones más sofisticadas incluyen estructuras para manipulación de datos (véase p. 204).

El espacio a ocupar



Antes de que el DOS pueda almacenar un archivo nuevo y hacer una entrada en el directorio, ha de consultar la "lista de sectores libres" o "mapa de disponibilidad de bloques" (BAM). Ésta es una sección de la memoria en la que cada bit corresponde a un bloque del disco. Un 1 binario indica que el bloque está ocupado, un 0 que está libre (en la ilustración aparecen como cuadrados llenos o vacíos). Las pistas más interiores (las de la parte inferior) poseen menos sectores porque son más cortas

Chris Stevens

Unidad "inteligente" Algunas unidades de disco contienen su propio microprocesador y su propia RAM. Se dice que estas unidades son "inteligentes", y el DOS está incorporado en forma de ROM. Cuando se utilizan unidades "no inteligentes", el DOS está almacenado dentro del ordenador

Límites exteriores

Aplicándole los accesorios apropiados, el ZX81 se puede ampliar y convertirse en una máquina sofisticada

El ZX81 de Sinclair es, de todos los microordenadores que existen actualmente en el mercado, el que ofrece la mejor relación entre calidad y precio, aun en su forma básica. Pero se encuentran a la venta una cantidad sorprendente de accesorios que lo pueden convertir en un sistema de microordenador notablemente sofisticado. Estas unidades incluyen gráficos en color de alta resolución, síntesis de voz y medios para comunicaciones. Por supuesto, el ordenador en sí mismo tiene algunas limitaciones, pero éstas se pueden remediar con la adición de numerosos artículos de fácil adquisición, como teclados estándar profesionales, memorias de acceso directo (RAM) extras y controladores programables de palancas de mando.

Paquete de RAM
En su forma estándar, el ZX81 sólo posee un Kbyte de RAM, del cual 123 bytes están reservados para las variables del sistema. Por consiguiente, la ampliación de memoria tal vez sea la primera exigencia del nuevo propietario. La ampliación de memoria de Sinclair viene en una sola forma (16 Kbytes), pero otras alternativas ofrecen hasta 64 Kbytes, como la versión Cheetah que muestra la ilustración

Acopladores acústicos
Los moduladores-demoduladores vienen en dos formas: modems de conexión directa, que requieren enchufar un conector adicional en el sistema telefónico, y acopladores acústicos como el Micro-Myte 60, que vemos en la fotografía, que utiliza el propio teléfono directamente.
Los modems de conexión directa, que suelen ser más caros, generan y reconocen señales electrónicas que representan los unos y los ceros de la información que se está recibiendo o transmitiendo. Los acopladores acústicos, que pueden funcionar a pila, traducen los ceros y los unos en tonos audibles para su transmisión a través de la red telefónica, y para recibir la información llevan a cabo el mismo proceso pero a la inversa

ROM en Forth
Los microordenadores ZX de Sinclair utilizan una versión de BASIC algo particular y, aunque no es posible instalar otra diferente, el usuario puede cambiar el lenguaje por completo: a FORTH, por ejemplo. Esto se puede hacer de dos formas: cargando el nuevo lenguaje en la RAM desde una cassette, lo cual significa que el ordenador revertirá al BASIC cada vez que se lo encienda o se lo restaure; o sustituyendo la ROM en BASIC por otra. Esta ROM en FORTH de la firma David Husband llega aún más lejos que la mayoría: permite ejecutar simultáneamente en el ordenador diez o más programas. Sólo es posible obtener total provecho de esta configuración en las aplicaciones de control, donde se deben programar de manera independiente diversos dispositivos

También disponibles...
Además de las unidades que vemos aquí, existen otros dispositivos para mejorar el rendimiento del ZX81. Una tarjeta de colores, por ejemplo, proporcionará hasta 16 colores en la visualización en televisión, y un generador de sonido dará tres "voces" programables. Las puertas bidireccionales pueden admitir hasta 16 dispositivos de entrada/salida a la vez. Lejos de ser sólo un ordenador personal pequeño y nada sofisticado, ideal para jugar y aprender las primeras nociones de la programación BASIC, el ZX81 de Sinclair se puede ampliar para aprovechar al máximo el potencial de que dispone su microprocesador Z80

Marcus Wilson-Smith

Síntesis de voz
Otro interesante accesorio de la firma Cheetah es la unidad para síntesis de voz Sweet Talker. Utiliza un sistema alofónico y, en consecuencia, es mucho más difícil de programar que las unidades que trabajan con fonemas (los alófonos son grupos de fonemas de sonido similar). Existen unidades semejantes para el ZX81 y otros microordenadores personales

Hebot
La tortuga Hebot, que se vende ya montada o en forma de kit, es uno de los más sofisticados robots móviles. Viene con el software para activarla y existe una variedad de extras disponibles, como fotosensores, que se pueden utilizar en unión de una cinta reflectante adherida al suelo, para hacer que el robot siga un camino predeterminado

Teclados
El teclado de membrana de capas múltiples tal vez sea la configuración menos satisfactoria del ZX81, de manera que no es sorprendente que varias empresas ofrezcan teclados alternativos de tamaño natural con teclas de resorte convencionales. El teclado Mapsoft ZX81, que vemos aquí, se vende montado o como kit de montaje. Además del juego de caracteres normal, el teclado Mapsoft proporciona tres teclas extras. Otra posibilidad es un teclado adhesivo de las mismas dimensiones que el propio teclado del ZX81, que se utiliza conjuntamente con el original. Localizar una tecla determinada por medio de él resulta bastante sencillo, pero no tiene ninguna otra aplicación

Mando de palanca
Si se considera que muchas de las unidades ZX81 que se han vendido se utilizan para juegos, quizá resulte extraño que Sinclair no haya fabricado su propio mando de palanca. Sin embargo, existe una gran variedad de ellos en el mercado; pueden ser no programables (especifican por usted los golpes de tecla que efectuará la palanca de mando) o programables (el usuario decide qué teclas se simularán). El modelo que mostramos aquí, el AGF Hardware, se programa moviendo los cables conectores. Otros se programan a través del ordenador. Éste acepta una palanca de mando con cualquier tipo de interruptor, así como una bola de mando

Impresora ZX
La ZX Printer, la impresora Sinclair, utiliza papel aluminizado, que es sensible a la electricidad. En vez de imprimir de una forma convencional, la cabeza de impresión elimina el revestimiento de aluminio para dejar al descubierto la superficie más oscura que se halla debajo. De funcionamiento razonablemente rápido, los principales problemas los constituyen el tipo y la anchura limitados del papel. No obstante, se puede utilizar una impresora normal por medio de una tarjeta para interface. Existen fichas que admiten interfaces Centronics y RS232

A toda marcha

Prestando una cuidadosa atención a las variables y a la estructura del programa, se puede acelerar la operación de prácticamente cualquier programa en BASIC

El BASIC es, a pesar de lo que afirmen sus detractores, un lenguaje versátil y un eficaz auxiliar educativo. Se puede escribir cualquier programa en BASIC siempre y cuando la máquina utilizada posea suficiente memoria y que el tiempo de ejecución no sea importante. Sin embargo, puesto que por lo general el BASIC se interpreta en vez de compilarse (véase p. 184), puede ser extraordinariamente lento al ejecutar los programas, en especial aquellos que exigen traducir y ejecutar repetidas veces una misma instrucción.

La clasificación, por ejemplo, es un proceso sumamente reiterativo: el procedimiento se efectúa dentro de un bucle y hay bucles más pequeños anidados dentro del bucle principal (véase p. 286). Si se han de clasificar 100 ítems, el programa puede efectuar entre 2 500 y 5 000 iteraciones del bucle. Una clasificación en BASIC siempre será lenta, pero la forma en que se escriba el código puede significar una sustancial diferencia en cuanto a la velocidad de ejecución. Si una instrucción se ha de repetir 5 000 veces, y si codificarla adecuadamente puede ahorrar una centésima de segundo del tiempo de ejecución para cada repetición, entonces el ahorro total será de 50 segundos: un considerable progreso para el usuario.

Para apreciar la diferencia que existe entre una buena y una mala codificación, necesitará un mecanismo de tiempos y un programa *testbed* (de apoyo). Si posee un ordenador Commodore, puede utilizar el reloj del sistema, con las variables correspondientes TI$ y TI, como parte del programa *testbed*. Si su ordenador no posee un reloj accesible, habrá de utilizar un cronómetro para medir el código en ejecución. También es una buena idea hacer que su programa le emita un "beep" cuando empiece y cuando termine, para que pueda saber cuándo está funcionando.

El programa *testbed* sería así:

```
1000 L = 500
2000 PRINT "***ADELANTE***":REM "BEEP"
       aquí instrucciones
2100 TI$ = "000000"
2200 FOR K = 1 TO L
.....
2900 NEXT K:T9 = TI
2950 REM "BEEP" aquí instrucciones
3000 PRINT "******STOP******"
3100 PRINT "Esto tardó    ";(T9/60);"   segundos"
```

Las líneas 2100 y 3100 son para los usuarios del Commodore. En otros ordenadores, elimínelas o sustitúyalas por el código adecuado. El código a medir lo colocaremos entre las líneas 2200 y 2900. Observe que los cronometrajes se ceñirán a las repeticiones de L donde L es el límite del bucle. Comprobar una sola ejecución de un trozo del código será muy inexacto, porque el reloj del sistema mide sólo en sesentavos de segundos y existe asimismo un tiempo general impuesto por el código del programa *testbed*.

He aquí algunas reglas generales para escribir en un BASIC eficaz, transcritas, aproximadamente, en orden de importancia:

1. Evitar en los bucles todo lo que sea aritmética.

Las funciones exponenciales (x^3, por ejemplo, que significa "x elevado al cubo") y matemáticas (cos(y), por ejemplo, que significa "el coseno del ángulo y") son particularmente lentas. La multiplicación y la división son procesos más lentos que la suma y la resta, pero aun la más rápida de estas operaciones (la suma) es relativamente lenta.

Inserte estas líneas en el programa *testbed*:

```
900 Z = 1.1
2300 X = Z↑3
```

y ejecútelo. En nuestra máquina, 500 repeticiones tardaron 27,95 segundos. Ahora sustituya la línea 2300 por:

```
2300 X = Z*Z*Z
```

y ejecútelo. Ello ocupó 3,55 segundos: ¡una diferencia sustancial!

Una investigación ulterior revelará el nivel de exponenciación en el cual vale la pena reemplazar la multiplicación repetida por la función exponencial. En nuestro ordenador, este nivel estaba en la potencia 18 (cuando X = Z↑18). Recuerde, no obstante, que para calcular $Z^{2,3}$, por ejemplo, la multiplicación repetida sería inútil, mientras que la función exponencial (↑) funciona para todos los números reales, incluyendo los negativos.

Utilice el programa *testbed* para comprobar cuánto tiempo tardan los otros procesos aritméticos, y compare las alternativas. ¿Es más rápido dividir un número por 2 o multiplicarlo por 0,5, por ejemplo?

2. Utilice variables en vez de constantes numéricas.

Cada vez que en una instrucción en BASIC se produce una constante numérica (7.280, por ejemplo), se invierte tiempo en traducir el número a una forma utilizable. Pruebe con esta línea:

```
2300 X = X+7280
```

En nuestra máquina, ésta tardó 4,63 segundos en ejecutar 500 repeticiones, mientras que:

```
900 C = 7280
2300 X = X+C
```

tardó 2,75 segundos en realizar la misma cantidad de repeticiones.

3. Si debe utilizar la sentencia GOTO, salte hacia adelante en su programa en vez de saltar hacia atrás. Sin embargo, si debe saltar hacia atrás, hágalo hasta el principio del programa en vez de saltar hacia atrás unas pocas líneas.

Lo mismo es válido para GOSUB. Al encontrarse con una instrucción GOTO o GOSUB, el intérprete de BASIC compara el número de la línea fijada con el número de la línea en curso. Si el número fijado es mayor que el número de la línea en curso, el intérprete simplemente busca hacia adelante, línea a línea, hasta hallarlo. Pero si el objetivo es menor que el número en curso, entonces la búsqueda siempre empieza por la primera línea del programa. Esto significa que puede ser más eficaz colocar las subrutinas y las secciones utilizadas más frecuentemente a cualquier extremo de un programa. Agregue 56 líneas REM al comienzo del programa para que tenga una longitud típica, y pruebe con:

```
2300 GOTO 2400
2400 GOTO 2500
2500 GOTO 2900
```

Con esto costó 2,33 segundos realizar 500 repeticiones, mientras que:

```
2300 GOTO 2500
2400 GOTO 2900
2500 GOTO 2400
```

ocupó 4,85 segundos.

4. Inicialice todas las variables en orden de frecuencia de acceso.

El intérprete almacena los nombres de las variables en una tabla de símbolos en el orden en que aparecieron por primera vez en el programa. Cuanto más tarde se produzca una variable en la tabla, más tiempo llevará hallarla y acceder a su contenido. Por la misma razón se debería evitar una variable nueva en un programa cuando pueda recurrir a alguna utilizada previamente por el programa pero que no se esté empleando en ese momento.

Si se utiliza una variable dentro de bucles anidados (hecho común en la clasificación), a esa variable se accede con mucha frecuencia, de modo que inicialícela al principio del programa antes que cualquier otra variable, con un valor ficticio en caso de que fuera necesario, como:

```
1000 L = 500:C = 7280:X = 0:Z = 1,1
2300 A = 0
```

que ocupó 2,2 segundos en efectuar 500 repeticiones, mientras que:

```
1000 A = 0:L = 500:C = 7280:X = 0:Z = 1,1
2300 A = 0
```

tardó 2,06 segundos.

5. Evitar la utilización de series.

Las operaciones en serie consumen mucha más memoria que las operaciones aritméticas, y de vez en cuando podría ser necesario que el intérprete tuviera que llamar a un programa del sistema denominado "recolector de información inservible" para poner en orden los contenidos de la memoria en serie. Este procedimiento puede tardar muchísimo tiempo.

Es difícil escribir una demostración general de ello, debido a la gran variación que experimentan los ordenadores en cuanto a la administración de su memoria: se ha de llenar con datos gran parte de la memoria para el usuario (bastará una gran matriz numérica) y luego efectuar manipulaciones en serie que hagan necesario llamar al "recolector de información inservible". En nuestra máquina dimos entrada a:

```
40 POKE 52,32:POKE 56,32:CLR
```

para reducir severamente la cantidad de memoria disponible para los programas en BASIC, y después dimos entrada a:

```
1000 L = 500:DIMT$(L)
1100 FOR K = 1 TO L
1200 T$(K) = "A"+"B"
1300 PRINT K
1400 NEXT K
```

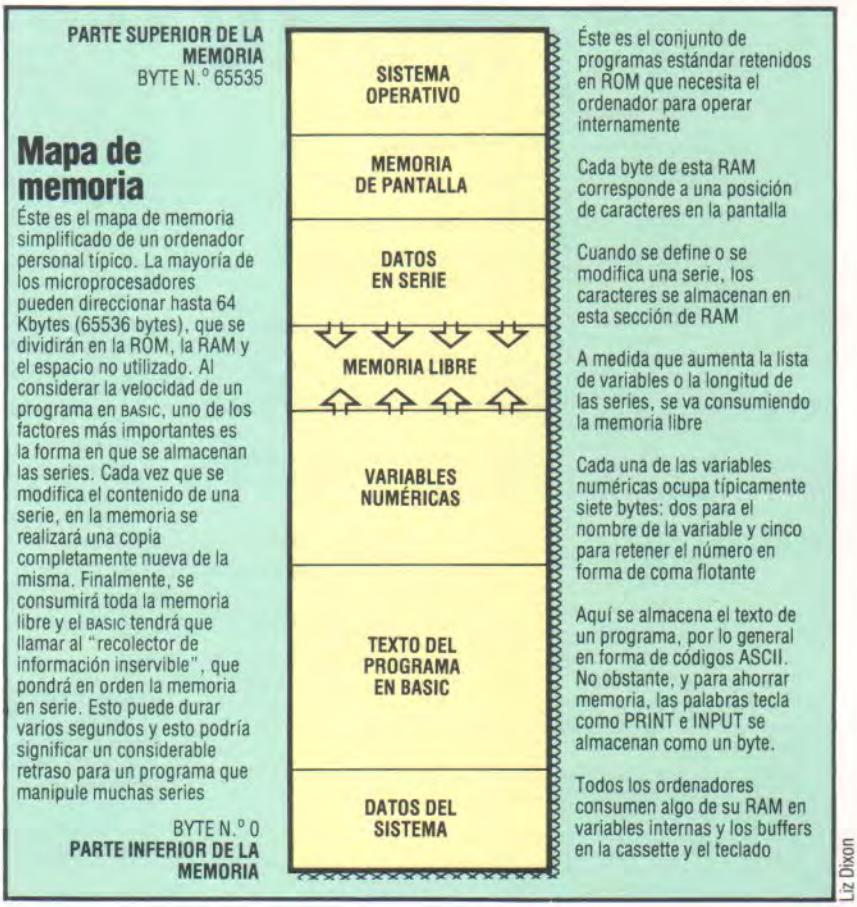
que consume muchísima memoria en serie y proporciona una matriz en serie para su utilización ulterior. La sentencia PRINT se ejecuta en cada iteración, visualizando el valor del contador del bucle. Cuando ejecutamos esta versión del programa *testbed*, la impresión se interrumpía repetidamente mientras se llamaba al "recolector de información inservible" para que reorganizara la memoria. Algunas veces la pausa duró más de tres segundos. El programa continúa:

```
2300 A$=LEFT$(T$(L),1):B$=A$+RIGHT$(T$(L),1)
```

y esto tardó 30,03 segundos en efectuar 500 repeticiones. Cuando ejecutamos el mismo programa con mucha más disponibilidad de memoria, el recolector de información inservible no se vio, y el bucle cronometrado duró 8,66 segundos.

Mapa de memoria

Éste es el mapa de memoria simplificado de un ordenador personal típico. La mayoría de los microprocesadores pueden direccionar hasta 64 Kbytes (65536 bytes), que se dividirán en la ROM, la RAM y el espacio no utilizado. Al considerar la velocidad de un programa en BASIC, uno de los factores más importantes es la forma en que se almacenan las series. Cada vez que se modifica el contenido de una serie, en la memoria se realizará una copia completamente nueva de la misma. Finalmente, se consumirá toda la memoria libre y el BASIC tendrá que llamar al "recolector de información inservible", que pondrá en orden la memoria en serie. Esto puede durar varios segundos y esto podría significar un considerable retraso para un programa que manipule muchas series



Liz Dixon

Tandy MC-10

Pese a su precio económico, este ordenador ofrece buen color y configuraciones propias de máquinas más caras

El teclado del Tandy MC-10
El teclado es de los de botón, pero es mejor que muchos otros. Las teclas son de plástico duro, con inscripciones grabadas, que tardan más tiempo en gastarse, y cuenta con una adecuada barra espaciadora. Lamentablemente hay una sola tecla SHIFT, situada a la derecha, y el gran botón de la izquierda corresponde a la tecla de CONTROL, localizada más convenientemente. La sensación táctil de las teclas es cómoda, pero no son aptas para la escritura rápida

Mando de borrado
Por ser grande y de color rojo, el mando de borrado es más fácil de hallar que los de otras máquinas. Al utilizar el MC-10, cuide de no golpear la parte posterior de la máquina en este lugar

Interface para cassette
Las patillas 1 y 3 de este enchufe DIN de cinco patillas proporcionan control remoto. La entrada de señales está en la patilla 4, la salida en la 5 y la señal a tierra en la patilla 2

Interface RS232
Es también un enchufe DIN, pero consta sólo de 4 patillas. La detección de señales portadoras está en la patilla 1, los datos de recepción en la 2, los de señal a tierra en la 3 y los datos de transmisión en la 4

Bus del sistema
No está definido en el manual, pero es obvio que está destinado a ser utilizado con algunas unidades de ampliación aún no especificadas. Hay, no obstante, líneas suficientes para manipular algunos dispositivos complejos

ROM
Está soldada firmemente al tablero y por ello no es probable que se pueda reemplazar por otras versiones. El BASIC Microsoft está almacenado en los 8 Kbytes de ROM

CPU
Insólitamente, el Tandy MC-10 utiliza un procesador 6803 en lugar de uno de los tipos de procesadores más populares. Éste es miembro de una de las familias más antiguas y no es tan conocido como el 6502 o el Z80. Sin embargo, es un útil chip de 8 bits con un razonable conjunto de instrucciones

RAM estática
Los 4 Kbytes nominales de la RAM del usuario están retenidos en estos dos chips de RAM estática de 21 Kbytes × 8 bits, así como la RAM de pantalla y algunas variables del sistema

6847 VDP
Al igual que muchas otras máquinas, la pantalla se controla mediante un chip especial, que en este caso es el 6847 Video Display Processor (procesador de visualización en video). Este chip (al menos en teoría) puede programarse para formatos de pantalla diferentes. No obstante, en la práctica esto se hace en muy raras ocasiones

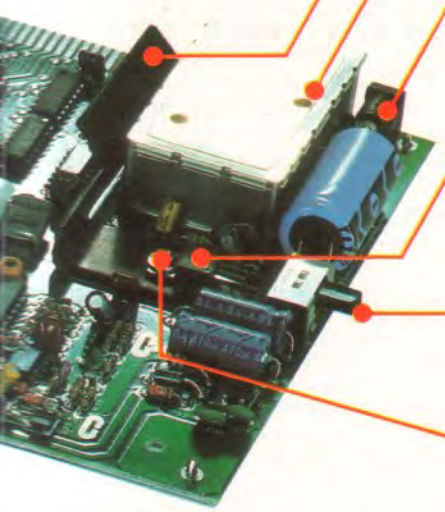
El Tandy MC-10 es una máquina pequeña y compacta que consigue mucho utilizando unos pocos y sofisticados chips. El teclado es del tipo botón, si bien ligeramente más grande que otros de esa misma clase, y posee una barra espaciadora adecuada. Otras configuraciones hacen que la máquina resulte bastante fácil de utilizar. La entrada de palabras clave en BASIC mediante una única tecla, por ejemplo, se consigue manteniendo pulsada la tecla CONTROL mientras se pulsa la tecla de la función deseada. La máquina se coloca en modalidad "de letras mayúsculas" al encenderse, y la modalidad de letras minúsculas se activa pulsando SHIFT 0 y se desactiva volviendo a pulsar las mismas teclas.

La visualización en pantalla es más pequeña que la de la mayoría de los otros ordenadores personales. Existen sólo 16 líneas de 32 caracteres y sólo se pueden obtener gráficos de una resolución bastante baja. La visualización adolece, asimismo, de otros defectos, incluyendo unas posibilidades de color más bien limitadas, si bien la calidad del color es muy alta. Lo más sorprendente de todo es que no visualiza caracteres en minúsculas, que sí se reconocen pero que en cambio se muestran como letras en mayúscula. El texto sólo puede ser verde sobre negro o viceversa: aunque los símbolos del bloque de gráficos se hallen en la modalidad de cualquiera de los nueve colores disponibles, la letra o el fondo

Disipador
El transistor regulador de potencia Triac se calienta mucho al estar en funcionamiento y esta gran pieza de metal disipa ese calor

Modulador de TV
Convierte el flujo de datos producido por el sistema de circuitos de video en una señal de TV de canal 36, pero sin sonido en la señal de TV. Ésta es la única salida de pantalla y la máquina no dispone de conector para monitor

Enchufe red
Se trata de un conector coaxial normal de poco voltaje. Al igual que otras máquinas de este tipo, el Tandy MC-10 toma su energía de un pequeño transformador de poco voltaje conectado a un enchufe



Regulador de potencia
Este gran transistor, junto con otros componentes cercanos a él, estabiliza la potencia transformada pero no regulada

Interruptor de potencia
Dado que el MC-10 posee un mando de borrado, no es necesario utilizarlo como alternativa, como ocurre con otras máquinas

Cristal
El reloj maestro genera una frecuencia de 4,4 MHz; está subdividido en impulsos más lentos y se lo utiliza para toda la máquina

Chris Stevens

TANDY MC-10

DIMENSIONES
210 x178 x51 mm

CPU
6803

VELOCIDAD DEL RELOJ
4,4 MHz

MEMORIA
8 Kbytes de ROM
4 Kbytes de RAM

VISUALIZACION EN VIDEO
16 líneas de 32 caracteres, 9 colores, pudiéndose determinar sólo el fondo. 75 caracteres predefinidos

INTERFACES
Interface RS232 en serie, cassette

LENGUAJE SUMINISTRADO
BASIC

OTROS LENGUAJES DISPONIBLES
Ninguno

VIENE CON
Manual de funcionamiento y manual de BASIC, cable para TV

TECLADO
48 teclas tipo botón

DOCUMENTACION
Clara, adecuada y bien diseñada, pero algo carente de información técnica. El único fallo importante es la ausencia de índice. Se incluye una ficha de referencia rápida, que da del BASIC suficientes detalles para que una persona experimentada pueda empezar a trabajar con la máquina enseguida

han de ser siempre negros. Por consiguiente, no se puede producir una forma azul sobre un fondo rojo, ¡ni siquiera en la modalidad de gráficos!

La función de sonido también tiene limitaciones. Sólo hay un canal disponible, que permite solamente variaciones mínimas de altura y de duración. Las facilidades de input/output son para cassette (incluyendo control remoto), televisión y una puerta en serie RS232. La puerta en serie se puede utilizar como línea para transferir datos hacia y desde otros ordenadores o, alternativamente, para activar una impresora. También se puede utilizar para crear una red con otros ordenadores Tandy MC-10.

Los diseñadores de la máquina no parecen haberse ocupado de manera especial de los juegos, ya que no han dotado al ordenador de conexiones para palanca de mando o mando de raqueta ni de ninguno de los chips especiales controladores de sonido y de gráficos existentes en otras máquinas más idóneas para juegos.

No obstante, para el futuro están claramente perfiladas algunas posibilidades de ampliación, dado que en un conector marginal hay una terminación de bus de sistema muy misteriosa, tapada con una placa atornillada. Aparte de afirmar que "esta ranura está reservada para futuras unidades de ampliación de memoria", el manual no dice nada más acerca de ella y no proporciona ninguna pista en cuanto a cuáles serán los accesorios que se podrán enchufar en tal ranura.

La documentación del Tandy MC-10 es la clásica documentación de las otras máquinas Tandy: texto bastante sólido, con pocas imprecisiones.

Por su precio asequible, vale la pena tener en cuenta este ordenador, pero al leer las especificaciones recuerde que por más que posea 4 Kbytes nominales de RAM, sólo son 3,142 bytes los que están disponibles para el usuario, porque de allí se toman también la RAM de pantalla y algunas variables del sistema.

Nuevos mandos

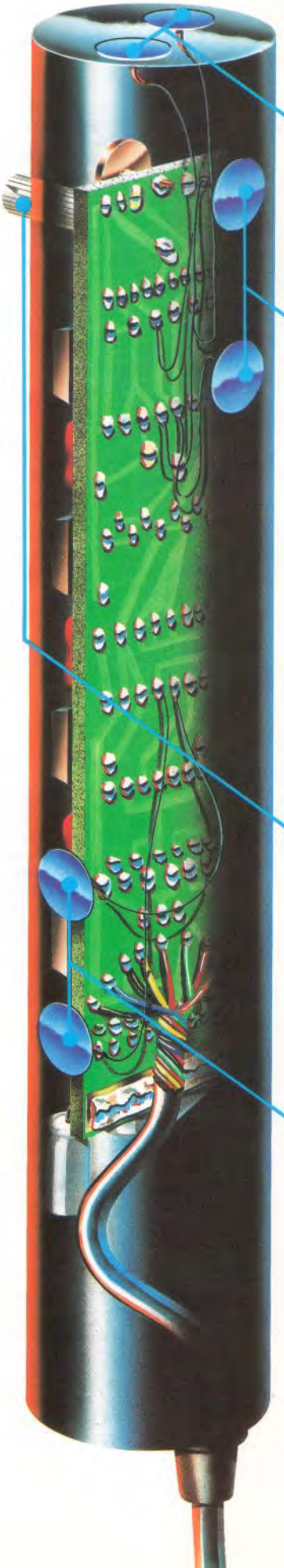
Dos innovadores tipos de palanca de mando. Una utiliza interruptores de mercurio; la otra capta las señales electromagnéticas del cuerpo humano

La industria del ordenador personal se ha habituado a un desarrollo tecnológico rápido, y los cambios no se limitan a los ordenadores: los periféricos y los accesorios están también sujetos a constantes refinamientos. Por ejemplo, en el breve tiempo transcurrido desde que analizáramos por primera vez el mecanismo de una palanca de mando (véase p. 56), han aparecido en el mercado dos tipos completamente nuevos de estos dispositivos.

Un dispositivo denominado *Le Stik* ha sido la primera palanca de mando analógica que ha dejado de lado los mecanismos de señalización habituales. Consiste en una empuñadura contorneada, provista de un pulsador de disparo montado en la parte superior y un control de pausa dispuesto en uno de los lados. La palanca de mando se sostiene en el aire y se inclina respecto a la vertical en la dirección requerida, y en la pantalla la imagen correspondiente se mueve de acuerdo con ella.

Manos arriba
La Trickstick se basa en el "zumbido" de la red, que es la radiación electromagnética que emite en todas las casas la red eléctrica. El cuerpo humano actúa como si fuera la antena de este "zumbido", y los sensores de la palanca captan los diferentes niveles de "zumbido" de acuerdo con la presión que se aplique con los dedos

Trickstick



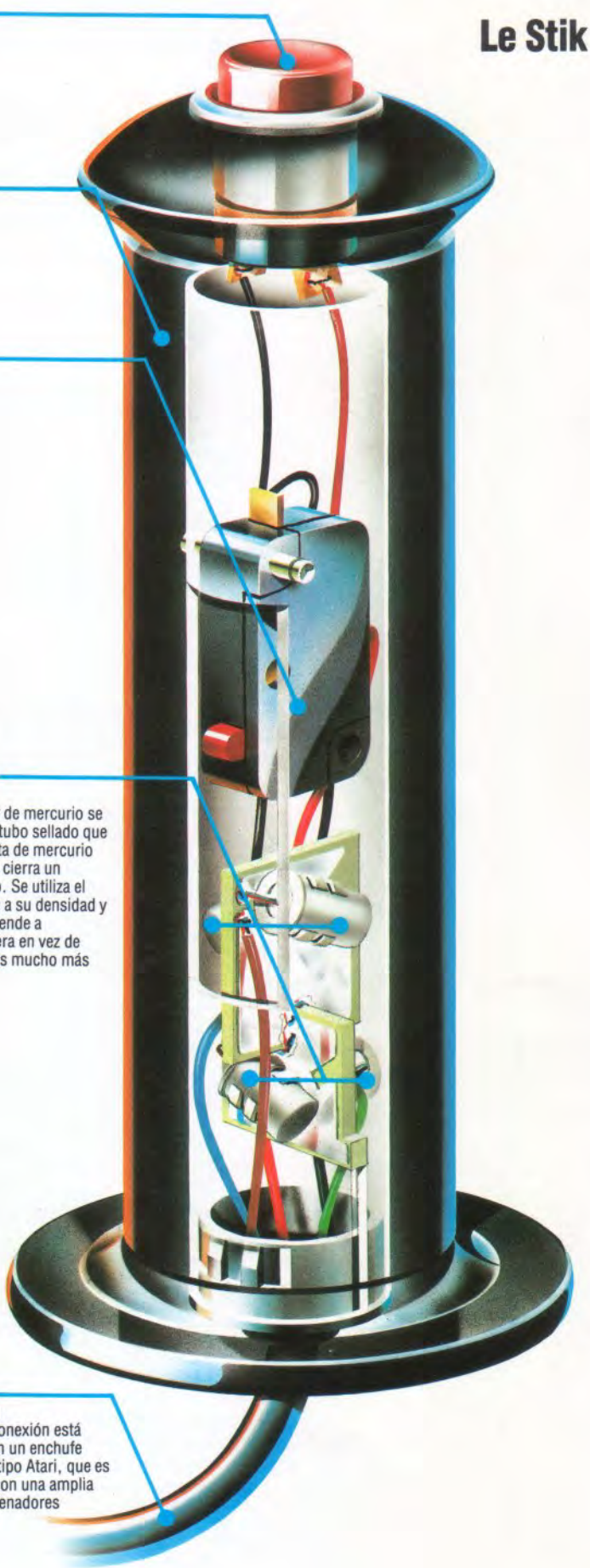
Controles de movimiento horizontal
Balanceando el pulgar sobre estos dos sensores se puede controlar el movimiento hacia atrás y hacia adelante

Controles de movimiento vertical
El sensor superior controla el movimiento hacia arriba; el inferior controla el movimiento hacia abajo

Control de sensibilidad
Permite ajustar la Trickstick al nivel de sensibilidad "emitida" por el usuario

Pulsadores de disparo
Cada uno genera una señal independiente, de modo que con un pulsador se podrían arrojar bombas mientras con el otro se disparan rayos láser

Le Stik



Pulsador de disparo
Está colocado en un lugar ideal para la acción de los juegos rápidos

Empuñadura
Le Stik es una de las pocas palancas de mando que poseen una empuñadura contorneada apta tanto para los usuarios diestros como para los zurdos

Botón de pausa
El botón de pausa, acoplado al microinterruptor de la empuñadura, le permite al jugador tomarse un respiro durante la acción, sólo con presionar el mando

Interruptores de mercurio
Cada interruptor de mercurio se compone de un tubo sellado que contiene una gota de mercurio que al inclinarse cierra un circuito eléctrico. Se utiliza el mercurio debido a su densidad y porque la gota tiende a permanecer entera en vez de dividirse en gotas mucho más pequeñas

Cable de conexión
El cable de conexión está equipado con un enchufe estándar de tipo Atari, que es compatible con una amplia gama de ordenadores personales

El mecanismo esencial de *Le Stik* se compone de cuatro tubos sellados llenos de mercurio. A medida que la palanca de mando se inclina respecto a la vertical, el mercurio fluye en la dirección elegida y hace uno o varios contactos eléctricos, como si se hubiera cerrado un interruptor. Al mover la empuñadura para que recupere la posición vertical, el mercurio vuelve a fluir en los tubos, cesando, de este modo, el contacto. La respuesta del sistema es mejor que la de las palancas de mando anteriores.

Por su parte, la *Trickstick*, diseñada por la East London Robotics', utiliza el más reciente procedimiento para transformar los movimientos manuales en señales que pueda comprender un ordenador. En efecto, esta palanca de mando constituye algo único en cuanto al efecto eléctrico que emplea: utiliza al cuerpo humano como una antena para captar el "zumbido" de la red eléctrica (la radiación electromagnética inocua que emite en cualquier habitación la red eléctrica). La *Trickstick* consiste en un tubo sellado dentro de una carcasa plástica, que se sostiene verticalmente entre ambas manos. En la superficie del tubo hay tres pares de sensores: un par controla el movimiento hacia adelante y hacia atrás; otro par controla el movimiento hacia arriba y hacia abajo y el par restante corresponde a los pulsadores de disparo.

El "zumbido" que capta el cuerpo humano se transmite a través de estos sensores al sistema de circuitos sensible, donde los impulsos se convierten en señales que le proporcionan al ordenador información direccional. Las señales también se pueden analizar para mostrar hasta dónde ha de llegar el movimiento. Cuanto más fuerte se pulse, más potente será la señal y más rápida la salida al ordenador. De este modo, la *Trickstick* combina el control proporcional de la palanca de mando analógica con el rápido control digital directo de una unidad basada en interruptor. Dado que diferentes personas afectarán a los circuitos de modo diverso, la *Trickstick* se ha de ajustar a la sensibilidad de cada usuario. Esto se realiza por medio de una perilla montada en uno de los extremos del dispositivo.

La idea de la *Trickstick* es ciertamente interesante, y los fabricantes se han apresurado a solicitar patente para esta técnica. No obstante, su fiabilidad y rendimiento aún están por ver.

Sistemas de sonido

Un segundo análisis de la capacidad de sonido del Vic-20

La última vez que analizamos el Vic-20 en esta sección, vimos de qué manera se pueden controlar los tres osciladores de la máquina "empujando" (POKE) las localizaciones de memoria, cómo establecer los niveles de volumen y cómo controlar la duración de una nota. Investigamos cómo se podían determinar la duración de las notas y las pausas entre ellas mediante la utilización de bucles FOR...NEXT o, con mayor eficacia, a través del empleo del reloj del Vic-20 para contar en *jiffys* (sesentavos de segundo). El manejo de estos tres elementos musicales (frecuencia, volumen y tiempo) le permite al usuario crear melodías sencillas en el Vic-20 y producir efectos sonoros.

Tocando melodías

Para crear una melodía, primero debe ensamblar las notas requeridas. Éstas podrían ser, por ejemplo, las notas de la primera línea de *Oh, I do like to be beside the seaside* ("Oh, quiero estar a la orilla del mar"). Por el orden correcto, éstas se podrían seleccionar como:

Re# Mi Fa Re# Do La# Sol# Sol Sol# Re# Re#

Utilizando las técnicas explicadas en la página 284, la duración de las notas y las pausas se podría establecer utilizando la configuración TI. Nuestra melodía, por lo tanto, se puede tocar ejecutando (RUN) el siguiente programa (observe la utilización de las variables para simplificar la selección de órdenes POKE):

```
10 V = 36878
20 FOR I = 1 TO 11
30 READ N: REM *NOTA*
40 POKE V,7:P = TI: REM *VOL ON*
50 IF TI-P<15 THEN 50: REM *PAUSA*
60 POKE V-3,N:D = TI: REM *TOCAR NOTA*
70 IF TI-D<20 THEN 70: REM *DURACION*
80 POKE V-3,0: REM *STOP NOTA*
90 NEXT I
100 DATA 203,207,209,203: REM *VALORES
    NOTAS*
110 DATA 195,187,179,175
120 DATA 179,203,203
130 POKE V,0: REM *VOL OFF*
140 END
```

Programa de luz

Primeros pasos con los sofisticados gráficos del BBC

El BBC Micro puede proporcionar unos efectos gráficos verdaderamente impresionantes tan sólo con unas pocas líneas en BASIC.

En el BASIC del BBC existen varias órdenes de alta resolución, incluyendo instrucciones para trazar líneas rectas, colocar puntos y trazar y rellenar triángulos. Esta última función se emplea para colorear el interior de las formas mediante una serie de triángulos pequeños, ya que este ordenador no dispone de ninguna orden del tipo PAINT (pintar). Carece, asimismo, de orden en BASIC para dibujar círculos y elipses, y carece de configuraciones para la programación de sprites. No obstante, posee características interesantes y poco frecuentes de las que carecen la mayoría de sus competidores. Éstas incluyen la posibilidad de mezclar texto y gráficos en la pantalla, cursores de texto y de gráficos que se pueden controlar por separado, y el acceso a la parte del sistema operativo de la máquina que controla la visualización en pantalla, desde dentro de un programa en BASIC. Esto se consigue mediante el conjunto de órdenes de pantalla o VDU. Las "ventanas" de texto y de gráficos también se pueden definir en la pantalla, lo que le permite al usuario dividir la visualización en sectores separados para gráficos y para texto.

Modalidades de visualización

El BBC Micro posee ocho modalidades para gráficos, tres de las cuales admiten sólo visualizaciones de texto. Se puede escoger entre 20, 40 u 80 caracteres a lo ancho de la pantalla, según la modalidad que se haya seleccionado. Los colores disponibles son dos, cuatro o dieciséis, dependiendo su número, nuevamente, de la modalidad seleccionada; no obstante, esta modalidad de color limitado posee una característica muy interesante: dos o cuatro colores de los que se pueden utilizar no son fijos y pueden ser seleccionados por el programador de entre los 16 normalmente disponibles.

MODE 7 (modalidad 7) se diferencia de todas las demás en que en ella no se emplean el juego estándar de caracteres ASCII y los códigos relacionados con ellos. En cambio, la visualización se construye con caracteres de teletexto. Las órdenes para gráficos normales, como PLOT y DRAW, no funcionan en MODE 7.

La siguiente tabla muestra las opciones de color y de resolución según la modalidad elegida:

Modalidad	Texto	Gráficos	Colores
0	80 ×32	640 ×256	2
1	40 ×32	320 × 256	4
2	20 ×32	160 ×256	16
3	80 ×25		2 (blanco y negro)
4	40 ×32	320 × 256	2
5	20 ×32	160 ×256	4
6	40 ×25		2 (blanco y negro)
7	40 ×25		Teletexto

Este programa simplemente toca las notas en la secuencia correcta con pausas y duraciones iguales. Por consiguiente, la melodía resultante es algo artificial. Si continúa experimentando, podrá crear programas más complejos que proporcionen intervalos y duraciones diferentes para cada una de las notas.

Efectos sonoros

Utilizando dos o tres osciladores se pueden tocar acordes sencillos. El programa que reproducimos abajo ejecuta el acorde en *re mayor* (*fa#, la* y *re*), empezando con el *fa#* solo y agregando el *la* y el *re* después de unas demoras establecidas en un segundo cada una. Después el acorde continúa durante dos segundos más.

```
10 POKE 36878,7
20 POKE 36874,233:D = TI
30 IF TI-D<60 THEN 30
40 POKE 36875,219:D = TI
50 IF TI-D<60 THEN 50
60 POKE 36875,147:D = TI
70 IF TI-D<120 THEN 70
80 POKE 36878,0:POKE 36874,0
90 POKE 36875,0:POKE 36876,0
100 END
```

No obstante, existen varios recursos para hacer más atractivo el tono de estos sonidos. Por ejemplo, el volumen se puede variar durante la duración de una nota, haciéndolo subir o bajar de acuerdo a una variable. Por ejemplo:

```
100 V = 36878
110 FOR I = 1 TO 12
120 POKE V,I
130 NEXT I
140 POKE V,0
```

Esto hace que el volumen vaya aumentando gradualmente desde 1 hasta alcanzar a 12 —la escala completa va desde 0 (apagado) hasta 15 (alto)—. Asimismo, el volumen se puede hacer "vibrar" alternando un nivel de volumen alto con uno bajo. La frecuencia se puede variar para "curvar" una nota modificando como sigue la línea 120 anterior:

```
POKE V-3,203+I
```

También vale la pena probar distintas combinaciones de ruido, frecuencias de oscilador y volúmenes. Esto a menudo resulta en un timbre más agradable. Tanto al hacer música como al agregarles efectos sonoros a los juegos, el objetivo de la informática es el de incentivar el interés del usuario, evitando la repetición constante de notas monótonas.

Hemos mostrado cómo se pueden manipular las sencillas configuraciones de sonido del Vic-20 para producir interesantes tonos y secuencias de notas. El principal problema es la falta de órdenes de sonido, lo que obliga a utilizar complicadas sentencias en BASIC para efectuar tareas relativamente sencillas. Esto desemboca en largas rutinas de programa que impiden que el intérprete de BASIC pueda procesar el código entre las notas con suficiente rapidez. La única forma sencilla de evitar este inconveniente consiste en invertir en uno de los muchos paquetes de programas comerciales que proporcionan órdenes extras para la programación de música. El cartucho Super Expander de Commodore proporciona una útil gama de órdenes para sonido, así como un recurso para almacenar melodías escritas con la ayuda del cartucho. No obstante, si usted busca un ordenador que posea configuraciones que le permitan ir más allá de la creación de música o sonidos elementales, sería conveniente que probara otros modelos, como el BBC Micro, el Commodore 64, el Dragon 32 o el Oric-1.

La pantalla de alta resolución se define con su origen en la esquina inferior izquierda de la pantalla, independientemente de la modalidad seleccionada. Los valores verticales oscilan entre 0 y 1023, y los horizontales entre 0 y 1279. Este procedimiento de delinear el mapa de la pantalla resulta muy conveniente cuando el usuario decide cambiar la visualización de una modalidad a otra. Conviene agregar que si en el curso de un programa se cambiara la modalidad de visualización, entonces la pantalla se limpiaría automáticamente.

Los colores del fondo, de los textos y de los gráficos se establecen utilizando las órdenes COLOUR y GCOL. El BBC Micro emplea la interesante idea de los colores lógicos y reales para permitir que el usuario seleccione un juego limitado de colores de entre los 16 disponibles. Para ilustrar esto, lo mejor es utilizar el ejemplo de emplear color en MODE 0, en la que sólo se pueden especificar dos colores. A los dos colores posibles para el primer plano se les dan los números de color lógicos 0 y 1 y, a menos que se instruya al ordenador en otro sentido, éste considera que 0 es negro y 1 blanco. La orden COLOUR selecciona el color de primer plano del texto. COLOUR 1 selecciona el número de color lógico 1 como el color del texto, pero se puede borrar el color de texto lógico utilizando una de las órdenes VDU. VDU19 define el color lógico. Para establecer el color lógico 1 en verde (número de color real 2) se necesita la siguiente orden:

```
VDU19, 1, 2, 0, 0, 0,
```

Los tres ceros a la derecha no tienen valor y están allí para una futura ampliación del sistema.

La orden GCOL tiene dos números relacionados con ella. El segundo es el número de color lógico para la visualización de gráficos; el primero se relaciona con la forma en que ese color se utiliza en la pantalla. Para la orden GCOL a,b los valores de a pueden oscilar entre 0 y 4, permitiéndole al usuario especificar si el punto o la línea se ha de visualizar en el color de primer plano lógico, si se ha de combinar mediante AND, OR o, a través de un OR exclusivo, con el color ya existente, o si se ha de invertir el color original.

En un futuro capítulo volveremos a ocuparnos del BBC Micro y explicaremos sus posibilidades de alta resolución, cómo definir caracteres y analizaremos con mayor profundidad el conjunto de órdenes VDU.

Cambiando de lugar

Tras analizar cómo insertar registros nuevos, seguimos con las formas de recuperarlos. Como ya dijimos, nos encontramos primero con el problema de hallar un emparejamiento exacto

El capítulo anterior finalizó con un ejercicio en el que se debía escribir un programa de tipo base de datos que permitiera dar entrada en ella a información. Analicemos algunos de los pasos que implica dar entrada a un registro nuevo como forma de continuar nuestro estudio de lo que entraña la etapa INICIALIZACION de nuestro programa principal. Daremos por sentado que existen los campos siguientes y las correspondientes matrices:

CAMPO	MATRIZ
1 campo del NOMBRE	NOMCAM$
2 campo del NOMBRE MODIFICADO	MODCAM$
3 campo de la CALLE	CALLECAM$
4 campo de la CIUDAD	CIUCAM$
5 campo de la PROVINCIA	PROVCAM$
6 campo del NUMERO DE TELEFONO	TELCAM$
7 campo del INDICE	INDCAM$

El significado de la mayoría de estos campos debería estar bastante claro, quizá con la excepción de los campos 2 y 7. Consideremos en primer lugar el campo del NOMBRE MODIFICADO. Cuando analizamos por primera vez el problema del formato de los datos para el nombre, dudamos entre dos opciones: darle el formato del nombre estrictamente especificado (rígido) o especificado con flexibilidad (libre), y optamos por esta última. Dado que la forma en que se puede dar entrada a un nombre es sumamente variable, un formato rígido hubiera dificultado en gran medida las rutinas de búsqueda y clasificación. Para obviar este inconveniente decidimos convertir todos los nombres a un formato estandarizado: todas las letras irían de mayúscula, todos los caracteres no alfabéticos (como espacios, puntos, apóstrofos, etc.) se suprimirían y sólo habría un único espacio (en caso de haber alguno) entre el nombre de pila y el apellido.

La necesidad de estandarizar los nombres de esta manera se plantea porque las rutinas de clasificación y de búsqueda han de tener alguna forma de comparar las semejanzas de unos con otros. Por otra parte, cuando recuperamos un nombre y una dirección de la base de datos, queremos que los datos se presenten en la misma forma en que se les dio entrada originalmente. Existen dos maneras de manejar este problema: o cada uno de los nombres archivados se convierte a la forma estándar sólo cuando se están llevando a cabo clasificaciones y búsquedas, o bien el campo del nombre se puede convertir a la forma estándar y almacenar como un campo separado, de modo que las rutinas de clasificación y búsqueda puedan tener acceso instantáneo a los nombres estandarizados.

El otro campo que podría inducir a confusión es el del INDICE. En realidad éste se incluye como un campo separado para permitir la futura ampliación o modificación de la base de datos sin que haya necesidad de reescribir una porción considerable del programa. Su inclusión introduce el tema del *encadenamiento* (término que se refiere a la determinación de las relaciones de los datos y el procesamiento). Todos los campos o elementos de cada uno de los registros están encadenados porque todos poseen el mismo índice (el mismo número de elemento o subíndice en sus respectivas matrices), y porque todos los campos de un registro se almacenarán juntos en un archivo. Esto puede hacer que, en una etapa ulterior, agregar nuevos tipos de datos o de relaciones resulte una tarea difícil que posiblemente implique la reorganización total de la estructura de archivos y la reescritura de gran parte del programa. La incorporación del campo del INDICE en esta etapa simplificará muchísimo la futura incorporación de modificaciones al programa.

Antes de intentar agregar un registro nuevo a la base de datos enunciaremos algunos supuestos relativos a la estructura de los archivos. En primer lugar limitaremos el número de registros a 50. Asimismo, vamos a suponer que ya se han transferido todos los datos a las matrices, como parte del procedimiento INICIALIZACION.

Cuando se añade un registro nuevo, lo más sencillo es agregarlo al final del archivo (es decir, en el primer elemento libre de cada matriz). Existen muchas probabilidades de que el registro nuevo esté desordenado en relación a los otros, pero éste es un problema que podremos investigar más adelante. Lo primero que se habrá de hacer, por lo tanto, será averiguar cuán grande es la matriz. Dado que ésta es una información que probablemente nos será de utilidad en muchas partes del programa, el mejor lugar para efectuarla es en INICIALIZACION. Éste es un caso en el que la necesidad de una variable global es muy clara (es decir, una variable que se pueda emplear en cualquier parte del programa). La denominaremos TAMAÑO. Otra variable global que probablemente tenga utilidad es el índice del registro en curso. Puesto que no habrá ningún registro en curso cuando se ejecute el programa por primera vez, para asignarle a CURSO un valor inicial habrá que esperar hasta que el programa haga algo con los datos. No obstante, CURSO se puede inicializar en 0 en el procedimiento INICIALIZACION. En BASIC no es estrictamente necesario inicializar una variable en cero, porque esto se realiza de forma automática. A pesar de ello es una buena costumbre y se debería hacer siempre con las variables locales para evitar los "efectos colaterales" derivados de la utilización de la misma variable en algún otro lugar del programa.

Cuando se ejecute el programa por primera vez, se producirán varios tipos de inicializaciones y los datos se habrán de cargar desde el disco o la cinta y transferir a variables alfanuméricas. Entonces se presentará el menú ELECCION. Si el usuario escoge

la opción 6 (agregar un registro en el archivo), se devolverá el valor 6 de la variable OPCION, que llamará al subprograma INCLREG. INCLREG supondrá que a TAMAÑO ya se le habrá asignado un valor y que, por lo tanto, puede empezar a solicitar entradas (también supone que INICIALIZACION ha DIMensionado correctamente las matrices necesarias).

Agregar un registro nuevo significa, asimismo, que ahora el archivo está, al menos en potencia, desordenado. Puesto que una clasificación podría ocupar cierto tiempo, tal vez fuera innecesario clasificar los registros después de efectuar cada adición; pero ésta es una decisión que por el momento vamos a posponer. En cambio, estableceremos una bandera para indicar que se ha agregado un registro nuevo.

Ahora estamos en condiciones de empezar a elaborar una lista provisional de las posibles matrices, variables y banderas que se podrían necesitar en el programa.

MATRICES

NOMCAM$	(campo del nombre)
MODCAM$	(campo del nombre modificado)
CALLECAM$	(campo de la calle)
CIUCAM$	(campo de la ciudad)
PROVCAM$	(campo de la provincia)
TELCAM$	(campo del número de teléfono)
INDCAM$	(campo del índice)

VARIABLES

TAMAÑO	(tamaño normal del archivo)
CURSO	(índice del registro en curso)

BANDERAS

RADD	(agregado registro nuevo)
SORT	(clasificado desde modificación del registro)
SAVE	(guardado desde modificación del registro)
RMOD	(efectuada modificación desde que se guardara por última vez)

Es probable que mientras se desarrolle el programa se presente la necesidad de incluir algunas matrices más. Ciertamente, será indispensable utilizar más variables. En cuanto a las banderas, es evidente que si bien se necesitarán otras, puede que no se requieran las cuatro que hemos reseñado arriba. No habrá necesidad ni de guardar ni de clasificar el archivo (se supone que ya está guardado y clasificado) a menos que se haya efectuado una modificación, de modo que quizá la única bandera realmente necesaria sea RMOD. Pero si decidimos emplear las cuatro banderas, el subprograma de INICIALIZACION las habría de establecer a todas en sus valores apropiados. Como ejercicio adicional en cuanto al refinamiento de la programación *top-down*, veamos lo sencillo que es codificar *INCLREG*

I 4 (EJECUCION) 6 (INCLREG)

```
EMPEZAR
   Localizar tamaño habitual del archivo
   Solicitar entradas
   Asignar las entradas a los finales de las matrices
   Establecer bandera RMOD
FIN
```

II 4 (EJECUCION) 6 (INCLREG)

```
EMPEZAR
   (el tamaño del archivo es TAMAÑO)
   (solicitud de entradas)
      Limpiar pantalla
      Imprimir mensaje solicitando la primera matriz (TAMAÑO)
      Dar entrada a los datos para la matriz (TAMAÑO)
      (solicitud y entrada para todas las matrices)
      Establecer RMOD en 1
FIN
```

Todo esto es directo y no incluye bucles ni ninguna otra estructura complicada. El paso siguiente puede ser la codificación directa en BASIC. Los únicos puntos que debemos considerar son que TAMAÑO es una variable que se establece durante la ejecución de INICIALIZACION y que no necesita codificarse como parte de esta sección.

III 4 (EJECUCION) 6 (INCLREG) CODIFICACION EN BASIC

```
CLS: REM O UTILIZAR PRINT CHR$(24) ETC
   PARA LIMPIAR PANTALLA
INPUT "DE ENTRADA AL
   NOMBRE";NOMCAM$(TAMAÑO)
INPUT "DE ENTRADA A LA
   CALLE";CALLECAM$(TAMAÑO)
INPUT "DE ENTRADA A LA
   CIUDAD";CIUCAM$(TAMAÑO)
INPUT "DE ENTRADA A LA
   PROVINCIA";PROVCAM$(TAMAÑO)
INPUT "DE ENTRADA AL NUMERO DE
   TELEFONO";TELCAM$(TAMAÑO)
LET RMOD = 1
LET INDCAM$ = CALLE$(TAMAÑO)
GOSUB *MODNOMBRE*
RETURN
```

La antepenúltima línea establece el campo INDCAM$ en el valor de TAMAÑO (convertido por CALLE$ en una serie), de modo que en una etapa ulterior pueda actuar como un índice. La subrutina *MODNOMBRE*, a la que se llama justo antes de que termine el programa, no es otra que el programa descrito con todo detalle en la página 254. A ese programa será necesario efectuarle algunas ligeras modificaciones, pero estas sólo son detalles. Esta subrutina tiene la función de tomar la entrada corriente (libre) del nombre y convertirla en una forma estándar. La salida de esta subrutina será un elemento (TAMAÑO) de una matriz denominada MODCAM$. Ahora todas las búsquedas y clasificaciones de nombres se pueden dirigir hacia los elementos de MODCAM$ y, dado que el elemento tendrá el mismo índice que los otros campos del registro, será fácil visualizar el nombre y la dirección tal como se les dio entrada originalmente. En otras palabras, la búsqueda se llevará a cabo en MODCAM$, pero la visualización provendrá de NOMCAM$.

Y esto es todo cuanto implica agregar un registro nuevo en el archivo, si bien no hemos dado cabida a ninguna verificación de error, ni hemos tomado medidas para la eventualidad de que no quedara más espacio en la matriz. Puesto que todos nuestros programas los estamos escribiendo en forma modular, las modificaciones y las mejoras de este tipo se podrán efectuar fácilmente después, sin tener que volver a escribir todo el programa.

Los subprogramas MODREG y BORREG (para modificar y borrar registros, respectivamente) son bastante similares a INCLREG, excepto en que antes de que se puedan ejecutar primero debemos localizar el registro que deseamos modificar. Por consiguiente, estos dos subprogramas comenzarán llamando a

ENCREG. Este subprograma se basa en una rutina de búsqueda similar a la descrita en la página 273. Esta vez la diferencia principal estriba en que (con toda probabilidad) no habrá ningún dato que sea idéntico a otro, porque muy pocas personas tienen exactamente el mismo nombre.

Una búsqueda se puede llevar de dos formas. Una consiste en buscar en una pila desordenada, lo cual impide que la indagación sea tan expedita como sería de desear. En el peor de los casos, puede que la rutina haya de buscar a través de toda la información antes de localizar el dato buscado. Sin embargo, buscar en una pila desordenada tiene la ventaja de que no se necesitan rutinas de clasificación cada vez que se agrega, se borra o se modifica un registro.

Si la información está ordenada de alguna manera (ya sea numérica o alfabéticamente, por ejemplo) el programa sólo habrá de buscar a través de una pequeña fracción de los datos de la lista. Cuanto más larga sea la lista, tanto más eficaz resultará la búsqueda binaria en comparación con la búsqueda a través de una pila desordenada. En realidad, si en el archivo hay datos suficientes como para garantizarlo, la clasificación de los registros después de una modificación se puede acelerar realizando una búsqueda preliminar para localizar la primera y la última aparición en la matriz de la letra inicial del apellido en el registro en cuestión.

Otra forma de acelerar la rutina de clasificación podría ser mantener una tabla de búsqueda de las localizaciones en la matriz donde conste la primera aparición de cada una de las letras del alfabeto. Sin embargo, esta tabla debería mantenerse (actualizarse) con sumo cuidado cada vez que se produjeran cambios en los datos.

El tema de la búsqueda y la clasificación constituye una de las áreas más extensas de la programación, y a él se han dedicado libros enteros. Nosotros no vamos a intentar descubrir la solución óptima para nuestro programa de agenda de direcciones, porque éste depende de una gran cantidad de factores, incluyendo el número de registros del archivo y también la posibilidad de disponer o no de unidades de disco.

A continuación ofrecemos un programa en seudolenguaje para efectuar una búsqueda a través de los elementos de la matriz MODCAM$. La variable alfanumérica CLY$ es la clave para la búsqueda. En este contexto, la palabra "clave" significa el grupo de caracteres de identificación utilizados para especificar qué registro (o registros) se necesita.

```
Solicitar el nombre a buscar
LET CLV$ = nombre (a buscar)
LET BMT = 1
LET BUSCA = 0
LET TOP = TAMAÑO
LOOP mientras (BTM< = TOP) AND (BUSCA = 0)
  LET MID = INT((BTM+TOP)/2)
  IF CLV$ = MODCAM$(MID)
    THEN
      PRINT NOMCAM$(MID)
      PRINT CALLECAM$(MID)
      PRINT CIUCAM$(MID)
      PRINT PROVCAM$(MID)
      PRINT TELCAM$(MID)
      LET BUSCA = 1
    ELSE
      IF CLV$>MODCAM$(MID)
        THEN LET BTM = MID+1
        ELSE LET TOP = MID-1
      ENDIF
  ENDIF
ENDLOOP
IF BUSCA = 0 THEN PRINT "REGISTRO NO
  HALLADO"
END
```

Este trozo de seudolenguaje se basa mucho en el programa utilizado para buscar los marcadores de fútbol de la página 275 de *Mi Computer*, pero verá que posee una salida adecuada si no se consigue hallar el registro (la última sentencia PRINT), que se ejecutará sólo si el bucle fracasa en localizar un emparejamiento exacto entre CLV$ y MODCAM$(MID).

Lamentablemente, es bastante improbable que se llegue a hallar un emparejamiento exacto, aun cuando el nombre y el número de teléfono que usted desee se encuentren en la base de datos. Esto se debe a que la sentencia IF CLV$ = MODCAM$ es totalmente inflexible; no permite que exista la más mínima diferencia entre la serie de caracteres a que el usuario da entrada en respuesta a la solicitud y la serie de caracteres almacenada en MODCAM$(MID). En una agenda de direcciones corriente, el ojo humano puede ir explorando la página de arriba a abajo y es capaz de tener en cuenta todo tipo de pequeñas diferencias que puedan existir entre la representación verdadera del registro y lo que se está buscando. El ordenador no puede hacer lo mismo.

Existen, sin embargo, algunas formas de evitar esto, aunque éstas suelen implicar un esfuerzo de programación extra y su ejecución lleva algo más de tiempo. La primera mejora sería verificar primero solamente el apellido, y por esta causa resulta coherente que el nombre almacenado en MODCAM$ esté en la forma de APELLIDO (espacio) NOMBRE DE PILA. Previamente, en nuestro curso de programación BASIC (véase el recuadro "Complementos al BASIC"), desarrollamos una rutina para invertir el orden de un nombre; ésta se puede incorporar como una subrutina dentro de la rutina INCLREG cuando se cree el campo MODCAM$.

Habiendo conseguido localizar la primera aparición del apellido requerido, la rutina ENCREG debería entonces verificar la parte del nombre de pila de ese elemento para ver si es idéntica a la entrada del nombre (CLV$). Si lo es, no hay ningún problema: se ha encontrado el registro. Sin embargo, si no es idéntica el problema empieza a complicarse, por lo cual debemos planificar nuestra estrategia con sumo cuidado. Podríamos, por ejemplo, buscar a través de todos los nombres de pila y, de no hallarse ningún emparejamiento exacto, empezar a buscar una pareja aproximada. La dificultad será: ¿qué es, exactamente, una pareja aproximada?

En el programa anterior, en lugar del mensaje "REGISTRO NO HALLADO" podría ser mejor dar un mensaje como "NO HALLADA PAREJA EXACTA, ¿PRUEBO CON UNA PAREJA CERCANA? (¿S/N?)". ¿Y qué significan las palabras "pareja cercana"? ¿Es Paco una pareja cercana de Francisco? ¿Y qué diría de Francis? Estas dos últimas representan posibles entradas para el programa ENCREG. Intentemos definir a qué nos referimos con la expresión una "pareja cercana" y empecemos luego a desarrollar un programa en BASIC para hallar la pareja más cercana para una entrada.

Supongamos que la serie de la memoria era FRANCISCO. ¿Cuál de las siguientes parejas es la más cercana: FRAN o FRNCS? La segunda extrae cinco letras de un total de nueve, mientras que la primera extrae sólo cuatro de un total de nueve. Por otra parte, la primera posee cuatro letras en la secuencia correcta, mientras que la segunda sólo posee dos en dos ocasiones.

Se trata de una elección en gran parte arbitraria. Nosotros optaremos por dar prioridad a una pareja exacta entre CLV$ y una subserie del nombre de la memoria. Si no se hallara una pareja exacta de ninguna subserie, el programa intentaría obtener el mayor número de letras en común. Enunciado en términos de entrada y salida, el programa es:

INPUT
Una serie de caracteres
OUTPUT
La pareja más cercana de la serie entrada

El siguiente programa, en un seudolenguaje bastante aproximado al BASIC, buscará a través de las series de una matriz y examinará las primeras "n" letras de cada una, siendo "n" el número de letras de la clave (CLV$). De no haber ninguna pareja, se imprimirá un mensaje en ese sentido:

```
DIM MAT$(4)
FOR L = 1 TO 4
READ MAT$(L)
NEXT L
DATA "FRANCISCO", "FERNANDO", "FRANÇOISE",
   "FRANCISCA"
LET CLV$ = "FAR"
LET LCLV = LEN(CLV$)
LET BUSCA = 0
LOOP FOR INDEX = 1 TO 4
   IF CLV$ = LEFT$(MAT$(INDEX),LCLV)
      THEN PRINT "LA PAREJA ES   ";MAT$(INDEX)
         LET BUSCA = INDEX
   ENDIF
ENDLOOP
IF BUSCA = 0
   THEN PRINT CLV$; "NO ES PAREJA EXACTA
        DE NINGUNO"
      PRINT "PRIMEROS";LCLV;"CARACTERES"
```

Después de esto, el programa podría seguir adelante mirando los grupos de caracteres de LCLV de longitud, empezando por el segundo carácter de cada serie. Si ninguno de ellos concordara, se podrían buscar los grupos que empezaran por el tercer carácter, y así sucesivamente. Finalmente, si ninguna de las ternas de caracteres de las series concordaran, el programa podría intentar averiguar qué serie tenía el mayor número de letras en común con CLV$. Y esto lo dejamos como un ejercicio para el lector.

En realidad podríamos escribir páginas y páginas acerca del tema de las parejas "libres" y de las diferentes técnicas empleadas en los paquetes de bases de datos comerciales. La mayoría de éstos ofrecen la posibilidad de buscar en algunos de los primeros caracteres del campo, como la codificación que acabamos de desarrollar más arriba. Otros recuperarán un registro si en algún lugar del campo aparece la secuencia de caracteres especificada, o incluso si apareciera en cualquier lugar del registro. La capacidad de "extravagancia" es particularmente útil: al especificar J?S nos encontraríamos con JESUS o JOSE, pero no con JUAN.

Complementos al BASIC

SPECTRUM

Éste es el listado del programa para invertir el orden de nombre de pila y apellido, que se publicara por primera vez en la página 136:

```
100 CLS
200 PRINT "INTRODUCIR NOMBRE EN LA
    FORMA"
300 PRINT "NOMBRE-DE-PILA APELLIDO"
400 PRINT "P. EJ. MARIA PEREZ"
500 INPUT "DE ENTRADA AL NOMBRE";N$
600 GOSUB 9500
700 PRINT "EL NOMBRE EN LA FORMA
    ESTANDAR ES"
800 PRINT N$
1000 STOP
9500 REM S/R PARA INVERTIR ORDEN DEL
    NOMBRE
9520 GOSUB 9600
9540 IF P = 0 THEN RETURN
9560 LET N$ = AS+ ",  "+P$
9580 RETURN
9600 REM S/R PARA CORTAR N$ POR UN
    ESPACIO
9620 LET N = LEN(N$)
9630 LET P = 0
9640 FOR K = 1 TO N
9650 IF FN M$(N$,K,1) = "  " THEN LET
    P = K: LET K = N
9660 NEXT K
9670 IF P = 0 THEN RETURN
9680 LET P$ = FN L$(N$,P-1)
9700 LET A$ = FN R$(N$,N-P)
9720 RETURN
9900 REM FUNCIONES EN SERIE
    DEFINIDAS POR EL USUARIO
9990 DEF FN M$(X$,P,N) = X$(P TO
    P+N-1)
9991 DEF FN L$(X$,N) = X$(  TO N)
9992 DEF FN R$(X$,N) = X$
    (LEN X$-N+1 TO  )
```

LEFT$

En el Commodore 64, Vic-20, Oric-1 y Lynx, reemplazar desde la línea 9600 a la 9720 del listado para el Spectrum por las líneas siguientes:

```
9600 REM S/R PARA CORTAR N$ POR UN
    ESPACIO
9620 LET N = LEN(N$)
9630 LET P = 0
9640 FOR K = 1 TO N
9650 IF MID$(N$,K,1) = "  " THEN LET
    P = K: LET K = N
9660 NEXT K
9670 IF P = 0 THEN RETURN
9680 LET P$ = LEFT$(N$,P-1)
9700 LET A$ = RIGHT$(N$,N-P)
9720 RETURN
```

RIGHT$

y suprimir desde la línea 9900 hasta la 9992.

MID$

En el Dragon 32 y en el BBC Micro, reemplazar desde la línea 9600 hasta la 9720 del listado para el Spectrum por las líneas siguientes:

```
9600 REM S/R PARA CORTAR N$ POR UN
    ESPACIO
9620 LET N = LEN(N$)
9640 LET P = INSTR(N$,"  ")
9670 IF P = 0 THEN RETURN
9680 LET P$ = LEFT$(N$,P-1)
9700 LET A$ = RIGHT$(N$,N-P)
9720 RETURN
```

y eliminar desde la línea 9900 a la 9992.

INSTR

Como ya hemos mencionado anteriormente, INSTR es una función muy útil, en especial al tratar con aplicaciones de tipo base de datos como lo es ésta. Si su máquina dispone de INSTR, tal vez se decida a intentar una forma más sofisticada de emparejamiento "libre".

INPUT

En el BBC Micro, reemplazar la línea 500 del listado para el Spectrum por:

```
500 INPUT "DE ENTRADA AL NOMBRE",N$
```

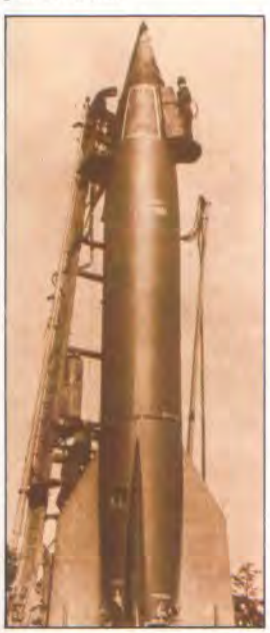
Tony Lodge

Konrad Zuse

Cortesía del Siemens-Museum, Munich

Zuse logró en Alemania resultados similares a los obtenidos por Von Neumann en Estados Unidos

La informática en la guerra
Los ordenadores de Zuse se desarrollaron para sustituir a los equipos de técnicos que efectuaban cálculos aeronáuticos trabajando con reglas de cálculo. Se aplicaron especialmente en el diseño de las bombas volantes V1 y V2 (en la fotografía), que utilizaría Alemania al final de la segunda guerra mundial



© The Imperial War Museum

Con frecuencia en distintos lugares del mundo se producen inventos simultáneamente a partir de ideas que surgieron y se desarrollaron independientemente una de otra. En la década de los cuarenta, mientras en Estados Unidos se estaba construyendo el primer ordenador de válvulas (el ENIAC), un ingeniero alemán, Konrad Zuse, trabajaba en una calculadora programable, que se considera como el primer ordenador de la historia.

Zuse nació en Berlín el 22 de junio de 1910. Después de estudiar en la Universidad Técnica de la ciudad, trabajó como ingeniero aeronáutico para la Henschel Aircraft Company, dedicándose al diseño de alas. Los principios matemáticos básicos aplicados al refuerzo de las alas de los aviones para resistir las tensiones del vuelo a alta velocidad ya se habían establecido en la década de 1920. No obstante, los cálculos individuales necesarios para producir cada par de alas requerían equipos de personas trabajando con máquinas de calcular mecánicas y reglas de cálculo. Zuse comprendió muy pronto la necesidad de contar con una máquina que pudiera efectuar con rapidez este trabajo que ocupaba tanto tiempo. Por las tardes, en compañía de otros amigos, emprendió, en el piso de sus padres, la labor de construir un ordenador que pudiera realizar esta tarea.

Su primera máquina, el Z1, era un dispositivo mecánico que podía efectuar las cuatro operaciones aritméticas elementales, calcular raíces cuadradas y convertir números decimales a notación binaria y viceversa. Aunque no estaba enterado de los logros de Charles Babbage (véase p. 220), cuyo ingenio diferencial se había creado para efectuar los laboriosos cálculos que requerían las tablas náuticas, Zuse había llegado a conclusiones muy similares y a otras que eran mucho más avanzadas. El descubrimiento más sensacional de Zuse se produjo al comprobar que una palanca era un interruptor que se podía colocar en una de dos posiciones (encendido o apagado) y que, por consiguiente, se podía utilizar ya como medio para almacenar datos, ya como dispositivo de control.

Zuse pretendía representar tanto los datos como las instrucciones en forma binaria, y el año 1941 inició la construcción de un ordenador electromagnético, al que él llamó Z2. Dedicado de lleno al esfuerzo de la guerra, el gobierno alemán se mostró poco interesado, al principio, en el invento. No obstante, finalmente acabó por reconocer el potencial interés militar del aparato y le proporcionó fondos a Zuse para desarrollar el Z3. Éste habría de ser un ordenador eléctrico, con un tendido de cables eléctricos, que posibilitaron un diseño más compacto y elegante, en lugar de los enlaces mecánicos que utilizó en las máquinas anteriores.

Zuse construyó el Z3 pese a no pocos contratiempos. Los bombardeos de Berlín por los aliados le obligaron a trasladar su taller en diversas ocasiones. Dos veces lo llamaron a filas, sólo para mandarlo de vuelta desde el frente oriental para que continuara su trabajo. La escasez de materiales durante la guerra le forzó a improvisar, obligándolo a servirse de piezas extraídas de los engranajes de conmutación telefónicos y a utilizar copias de antiguas películas, perforadas con códigos de ocho agujeros por fotograma, en lugar de cinta de papel.

El Z3 podía almacenar 64 palabras, cada una de ellas de 22 bits de longitud. A la información se le daba entrada a través de un teclado y los resultados se exhibían visualmente en un conjunto ordenado de lámparas montadas sobre un tablero. Lamentablemente, el Z3, al igual que todos los ordenadores anteriores de Zuse, fue destruido en 1945 durante un bombardeo de Berlín.

Uno de los ordenadores lo adaptó la Henschel Aircraft Company para ayudar en la construcción de la bomba volante HS-293. Se trataba de un avión no tripulado que se lanzaba desde un bombardero y se guiaba hasta su objetivo por radio.

El último ordenador que produjo Zuse durante la guerra, el Z4, había incrementado la longitud de sus palabras a 32 bits. Cuando los aliados se acercaban a Berlín, la máquina fue trasladada a Gotinga. Finalmente quedó instalada en Basilea (Suiza), donde estuvo en funcionamiento hasta 1954.

Zuse no consiguió fabricar ordenadores en la Alemania de la posguerra, por lo cual dedicó sus esfuerzos a la teoría informática. Desarrolló un sofisticado lenguaje denominado Plankalkül que podía tratar lógicamente tanto con matemáticas como con información más general. Cuando pudo volver al campo de la creación de ordenadores, fundó la Zuse Company, que fue la fábrica de ordenadores más importante de Alemania hasta 1969, en que fue absorbida por Siemens Corporation.

Transporte informatizado

Cortesía de YRM Arquitectos y Urbanistas

El "People Mover"
Siguiendo el ejemplo de muchos aeropuertos de Estados Unidos, se ha instalado en el aeropuerto de Gatwick, al sur de Londres, un revolucionario sistema de transporte interterminal; combina la estabilidad direccional y la capacidad de funcionar automáticamente, sin conductor, de los ferrocarriles ligeros, con la comodidad del autobús y el coche. Diseñado y construido por la Westinghouse Corporation, empresa famosa por sus sistemas de señalización de vías, el *People Mover* ("transportador de personas") puede trasladar hasta 100 pasajeros en cada viaje

En un mundo cada vez más poblado, transportar personas y bienes es una labor compleja. La utilización de ordenadores contribuye a que todo marche mejor

Hace 150 años la travesía desde Europa a Australia duraba tres meses. En la década de los ochenta, ese mismo viaje se podría efectuar en menos de doce horas. Sin embargo, este milagro tecnológico no habría sido posible de no existir métodos de control computerizado.

El viaje aéreo es la forma de transporte que plantea los problemas más serios. En el aeropuerto de Heathrow, en Londres, por ejemplo, se contabilizan más de un millar de vuelos diarios, con 120 desplazamientos de aviones por hora durante los períodos punta. Sin el auxilio de los ordenadores para controlar esta actividad, el sistema sencillamente no podría funcionar.

Consideremos el caso de un viajero que desee trasladarse de Londres a Nueva York. Desde la agencia de viajes, donde compraría el billete y se le reservaría una plaza (utilizando el Prestel como puerta para el ordenador de reservas de la propia compañía aérea), hasta que tocara tierra en el aeropuerto Kennedy, es posible que participen directamente en el viaje hasta 15 ordenadores diferentes. Analicemos con mayor detalle el papel del ordenador en un viaje aéreo.

Veamos primero el aparato. Los aviones de pasajeros modernos valen una enorme cantidad de dinero. Con el fin de garantizar al máximo su inversión, los operadores deben mantener al avión "como nuevo" en la mayor medida posible, y la clave para ello es el "mantenimiento planificado". Después de un número preestablecido de horas de vuelo, el aparato regresará a su base de ingeniería, donde el equipo de personal tendrá acceso a registros computerizados del historial completo del avión desde el día en que fue construido, pasando por el número de serie de cada pieza, tanto de las que están actualmente en uso como de las reemplazadas. Los registros detallarán todas las operaciones de ingeniería a las que haya sido sometido el avión; informes de los mecánicos de a bordo y de otros miembros de la tripulación acerca de su rendimiento; cifras del consumo de combustible, y cualquier otro tipo de dato que pueda considerarse de interés.

El avión sólo volverá a entrar en servicio cuando el plan de mantenimiento esté completo y actualizado. Y aun entonces se integrará a otro sistema informático: el sistema de control operativo de las líneas aéreas. Éste le señala al avión las rutas, distribuye pedidos de combustible en diversos puntos a través de esas rutas y se ocupa de la tripulación, las comidas, la atención y distracción de los pasajeros durante el vuelo y del gran número de medidas necesarias para transportar a trescientas o cuatrocientas personas a través de medio mundo.

Otro sistema de control computerizado opera dentro del propio aeropuerto, donde los funcionarios han de hacer frente a la inmensa demanda de sus recursos, a veces limitados, por parte de las compañías aéreas, para las que unos pocos minutos de retraso pueden representar una considerable pérdida de dinero. Esta operación sólo se puede

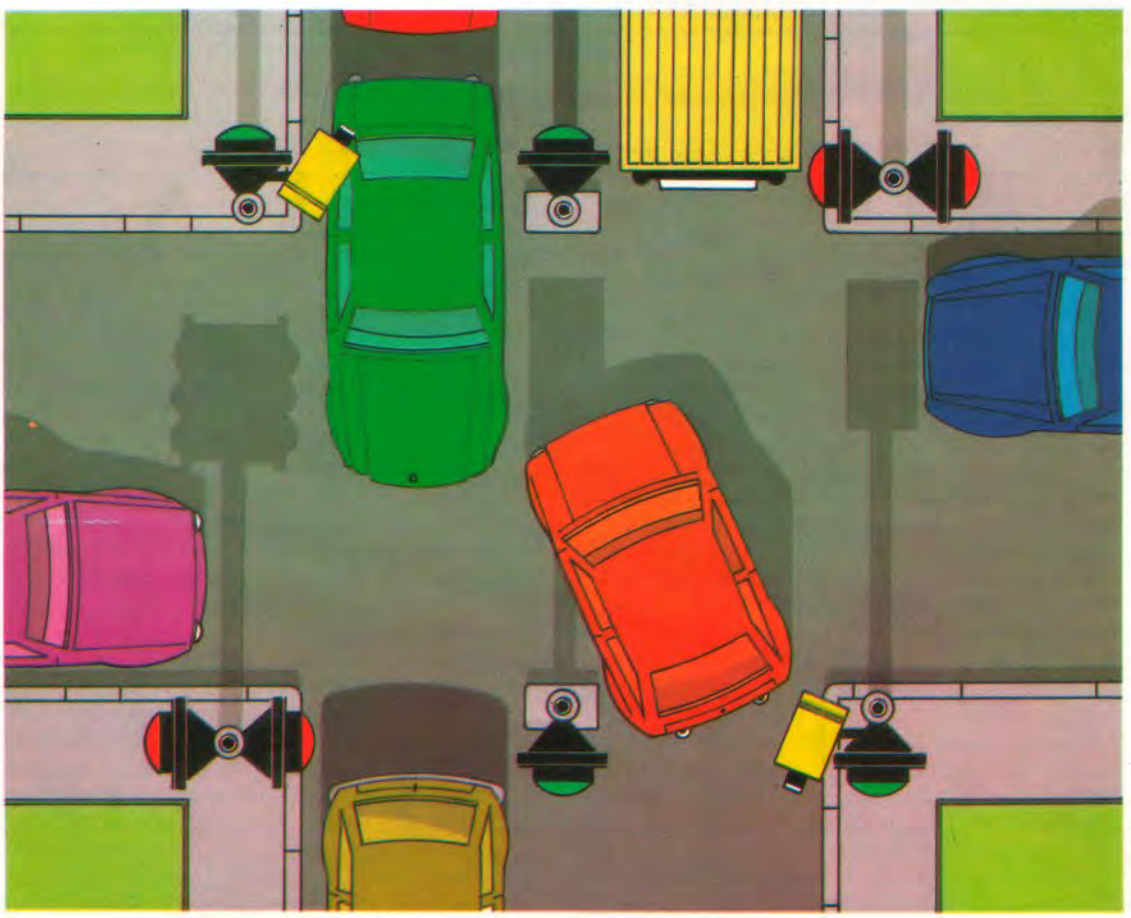
llevar a cabo con total éxito gracias a la planificación computerizada. El sistema informático del aeropuerto también se ocupará de llamar a los pasajeros para verificar las listas de vuelo y de visualizar la información relativa a salidas y llegadas.

Antes de que los pasajeros hayan siquiera llegado al aeropuerto, el piloto habrá archivado un detallado plan de vuelo con el Control de Tráfico Aéreo. De modo sorprendente, la tarea del CTA se efectúa sólo parcialmente asistida por ordenador. Gracias a los sistemas de radiofaro de respuesta por radar, los controladores ya no tienen que depender de que el piloto comunique su posición verbalmente. Cada blip que ellos ven en sus pantallas de radar se identifica mediante un número de vuelo, junto con la lectura de altitud interpretada por ordenador y el código de destino que se transmiten automáticamente desde el avión. El controlador dispone aún de otro nivel de asistencia por ordenador en la forma de líneas impresas, cada una de las cuales cubre un segmento de la ruta planificada, derivada del plan de vuelo del piloto. Estas líneas, con información relativa a la trayectoria de vuelo, la altura, carga útil y tipo de avión, ayudan al controlador a guiar el vuelo a través de su zona de la forma más rápida y más económica.

En el transporte por ferrocarril también está muy difundida la utilización de ordenadores. El trabajo del guardavía ferroviario, si bien no es tan complejo como el de los controladores de tráfico aéreo, tiene varios rasgos en común. Su labor consiste, igualmente, en guiar el tránsito de pasajeros y de mercancías a través de su zona con absoluta seguridad y con el menor costo posible. En Gran Bretaña, la British Rail viene utilizando sistemas de control por ordenador desde mediados de los años setenta, siguiendo el ejemplo que puso en práctica la Southern Pacific Railroad en Estados Unidos. Su TOPS (*Total Operations Processing System*: sistema de procesamiento total de operaciones) cubre todos los aspectos del control de carga, desde llevar el inventario exacto y actualizado al minuto de cada locomotora y de cada componente de material rodante, hasta ensamblar estas unidades para formar trenes completos y determinar sus rutas.

Cada vagón de mercancías posee un número de identificación exclusivo que el ordenador del TOPS graba junto con su localización. Cuando en otro lugar se necesita un vagón de esa clase, éste es asignado a un tren determinado y se apunta su destino. Habiendo cumplido su cometido, el TOPS toma nota de la nueva posición del vagón y queda en condiciones de volver a asignarlo. Si se considera el volumen del tráfico de mercancías que maneja la British Rail (a finales de la década de los setenta había 185 000 vagones distribuidos en casi 2 000 trenes diarios), se comprende la necesidad de utilizar estos sistemas de control por ordenador.

En las estaciones de ferrocarril se presentan varios de los problemas de operación que existen en los aeropuertos, y en ellas se han puesto en práctica muchas soluciones similares. No obstante, en el caso de los ferrocarriles hay que tener en cuenta otro adelanto: la creciente utilización de estaciones sin personal. Se han llevado a cabo experimentos con microordenadores que respondían a las preguntas de los viajeros (y, en algunos casos, anunciaban las salidas y llegadas mediante síntesis de voz). La informática también ha hecho posible la existencia de trenes sin maquinista. En Londres, por ejem-



Luz verde y luz roja
Los ingenieros de caminos han venido utilizando durante algún tiempo el ajuste de fase de las señales de tráfico con el fin de regular el flujo de vehículos, especialmente en las calles principales de las ciudades. En la actualidad, los semáforos pueden controlar de manera individual la densidad del tráfico en sus inmediaciones mediante detectores de radar doppler y transferir esta información a un sistema de ordenador. La frecuencia del cambio de luz de los semáforos se puede adecuar, entonces, a las condiciones de cada momento

Roy Ingram

Cortesía de British Airports Authority

"Se anuncia la salida..."
Quizá sea el tablero de salidas y llegadas el aspecto del sistema informático operativo del aeropuerto que más trasciende al público. Estos dispositivos electromecánicos los actualiza constantemente el ordenador central a medida que va recibiendo nueva información

plo, el metro de la línea Victoria, de la London Transports, dispone de esta capacidad, si bien no se emplea en la práctica debido al recelo que despiertan en el público los trenes sin conductor.

Las representaciones a escala más sofisticadas de redes de ferrocarriles se aproximan mucho a lo que es un sistema de tren sin conductor. Cada locomotora representada posee un código identificador exclusivo, denominado "número de dispositivo", almacenado en un pequeño microordenador basado en un solo chip. El controlador le envía las señales a cada tren modulando la energía conducida a lo largo de las vías, de modo tal que sólo una locomotora será capaz de decodificarla. De este modo, un elevado número de trenes puede circular simultáneamente por el mismo trazado bajo el control directo del microordenador central.

El tren sin conductor es viable porque circula sobre rieles, pero es poco probable que el concepto se utilice para los vehículos de carretera. No obstante, ya se emplean ordenadores en el transporte por carretera, básicamente para el transporte público y las operaciones de carga. En este caso contribuyen a planificar y trazar las rutas de los vehículos, así como a confeccionar horarios (lo cual es una tarea compleja en una gran ciudad, donde es necesario integrar los servicios de autobuses, trenes y metro en un único sistema de transporte público). El problema consiste en mantener en funcionamiento justo la cantidad de vehículos necesarios para que los pasajeros no sufran demoras inaceptables y, al mismo tiempo, no perjudicar la rentabilidad de la empresa. Desde el punto de vista estadístico esto constituye un espinoso problema, pero fue para resolver problemas de esta índole que se desarrollaron por primera vez los ordenadores. Otra tarea que se puede realizar con la aplicación de métodos estadísticos es el trazado de las rutas de los camiones de reparto para minimizar la distancia a cubrir entre cada entrega y la asignación de consignas a cada camión. En este sentido, una variante interesante se observa en el envío de vehículos, especialmente de taxis y de coches policiales, que van "viajando" en vez de regresar a su punto de salida. A la localización de cada vehículo se le da entrada como el nombre de una calle, que el ordenador convierte en la referencia de una cuadrícula. A cada solicitud de envío se le da entrada de la misma forma, y la tarea de relacionar los recursos con las llamadas es una simple cuestión de comparar los dos de acuerdo a reglas predeterminadas.

El sistema denominado *dial-a-bus* ("llame un autobús"), actualmente en funcionamiento en la periferia de Hannover (Alemania), representa uno de los experimentos más interesantes llevados a cabo con ordenadores en el área del transporte público. Basado en una flota de miniautobuses que no siguen ninguna ruta preestablecida, el sistema permite que el usuario telefonee desde la parada del autobús a la estación de control central, indicando su punto de destino. La miniterminal (que tiene el aspecto de una terminal de cajero automático) imprime entonces la hora a la cual llegará el autobús (el margen de tiempo nunca supera los cinco minutos), la duración del viaje y el precio del billete.

El transporte de pasajeros y de mercancías supone el 20 % de todo el comercio internacional. En este campo la utilización del ordenador está más avanzada que en otros y sin duda alguna ha contribuido significativamente a su crecimiento. Si bien la mayoría de los ejemplos que hemos citado se realizan con miniordenadores u ordenadores de unidad principal, muchos de ellos también se pueden efectuar con microordenadores personales.

Carga por ordenador
La industria naviera utiliza los ordenadores en mucha menor medida que otros medios de transporte; una importante área de aplicaciones es la de los servicios de embalaje de carga en contenedores. Los microordenadores juegan un papel muy importante en la "consolidación" (grupo de empresas fletadoras que comparten un contenedor), organización y arreglo de la terminal de contenedores y en la carga del buque contenedor con el fin de asegurar una distribución uniforme del peso

Cortesía de «The Motor Ship»

Hablando idiomas

El BASIC posee una construcción muy familiar y por ello resulta sencillo de aprender, pero no es tan eficaz como otros lenguajes

A menos que su ordenador personal sea un Jupiter Ace (véase p. 150), es casi seguro que el lenguaje de programación que lleva incorporado su aparato es el BASIC. Pero esto no significa que usted esté limitado a esa opción; en efecto, a pesar de que el BASIC está reconocido como un lenguaje particularmente fácil de aprender, hay otros que resultan muchísimo más apropiados para escribir aplicaciones específicas. Para disponer de estos lenguajes en su ordenador le será necesario cambiar las unidades ROM que contienen el intérprete de BASIC, o bien cargar en RAM el nuevo lenguaje; en este último caso, necesitará una máquina que posea una capacidad de memoria razonable, para que quede suficiente RAM libre para contener sus programas. Algunos ordenadores personales, como el Sharp MZ-711, se han anticipado a este problema cargando el intérprete de BASIC también en cassette.

PASCAL-CASTELLANO

CASTELLANO-PASCAL

El PASCAL se desarrolló a principios de la década de los setenta con la intención de convertirlo en el sucesor del BASIC. Su gama de estructuras de datos y de control deriva del grupo de lenguajes FORTRAN/ALGOL, y están pensadas para alentar al estudiante a enfocar la programación de forma sistemática y a escribir códigos bien estructurados y fácilmente comprensibles. Se trata de un lenguaje de programación muy conveniente para desarrollar una buena técnica de programación, aunque esto conlleve que las primeras etapas del aprendizaje de la programación puedan resultarle más difíciles al principiante absoluto, en parte debido a la disciplina que impone el lenguaje y también porque generalmente se compila en vez de interpretarse. No obstante, los programas en PASCAL tienden a ser elegantes, se desarrollan con relativa rapidez y son mucho más fáciles de comprender.

Éste es el equivalente en PASCAL del programa en BASIC:

```
VAR
NAME          :PACKED
              ARRAY (1..30)
              OF CHAR;

AGE, COUNT    :INTEGER;
ANSWER        :PACKED
              ARRAY (1..3)
              OF CHAR;

RUNNING       :BOOLEAN;

BEGIN
RUNNING: = TRUE;
WHILE RUNNING DO
  BEGIN
  WRITE('¿Cuál es su nombre?');
  READLN(NAME);
  WRITE ('¿y cuántos años tiene?');
  READLN(AGE);
  FOR COUNT: = 1 TO AGE DO
    WRITE(COUNT,'Hola',10,NAME);
  WRITE('¿Quiere otra pasada?');
  READLN(ANSWER);
  IF ANSWER(1) = 'N'
  THEN RUNNING: = FALSE;
  END;
WRITELN('Adiós', NAME)
END.
```

BASIC-CASTELLANO

CASTELLANO-BASIC

El BASIC, un lenguaje muy difamado, se desarrolló a partir del FORTRAN (que fue uno de los primeros lenguajes de programación de alto nivel y que sigue siendo uno de los lenguajes más populares para aplicaciones científicas y de ingeniería), destinado a los estudiantes universitarios como una forma de introducción a la programación. Dado que fue concebido como método docente, el BASIC por lo general no se compila sino que se interpreta (véase p. 184), esta característica ha sido la causa principal de que se convirtiera en el lenguaje incorporado de casi todos los ordenadores personales. A los lenguajes interpretados no resulta caro ponerlos en práctica, utilizan comparativamente menos memoria del ordenador y son muy adecuados para desarrollar programas.

Ahora el BASIC es un lenguaje eficaz por derecho propio, pero se encuentra limitado por poseer demasiadas versiones no estandarizadas (el BASIC de cada ordenador es exclusivo de esa máquina) y por su carencia de datos especializados y de estructuras de control. A causa de estas desventajas es posible que los programadores autodidactas desarrollen malas técnicas de programación y que una buena técnica de programación pueda ser difícil de aplicar a unos problemas específicos en BASIC.

En este corto programa se pueden apreciar el atractivo y el carácter general de este lenguaje, pero también sus limitaciones:

```
100 INPUT ¿Cuál es su nombre?";N$
200 INPUT "¿y cuántos años tiene?";A
300 FOR K = 1 TO A
400 PRINT K,"Hola  ";N$
500 NEXT K
600 INPUT "¿Quiere otra pasada?";R$
700 IF LEFT$(R$,1) = "S" THEN GOTO 100
800 PRINT "Adiós  ";N$
900 END
```

COMAL-CASTELLANO

CASTELLANO-COMAL

El COMAL se desarrolló para combinar la accesibilidad del BASIC con las eficaces estructuras y el enfoque disciplinado del PASCAL. Por lo tanto se asemeja a ambos y puede que haya sido tomado como modelo para crear el BASIC del BBC Micro, que casi se ha desarrollado como un nuevo lenguaje.

El COMAL ha obtenido un gran éxito en la informática escolar y en los países escandinavos (donde se originó), pero no parece probable que llegue a desplazar a sus dos antecesores como el lenguaje de introducción a la programación.

Éste es el programa "Hola" en COMAL:

```
100 RUNNING: = TRUE
200 WHILE RUNNING DO
300   INPUT"Cuál es su nombre?":N$
400   INPUT"¿y cuántos años tiene?":A
500   REPEAT
600     FOR K: = 1 TO A DO
700       PRINT K,"Hola";N$
800     NEXT K
900   INPUT"Quiere otra pasada";R$
1000  IF R$ = "N" THEN RUNNING: = FALSE
1100 ENDWHILE
1200 PRINT"Adiós";N$
```

LISP-CASTELLANO
CASTELLANO-LISP

El LISP se desarrolló a principios de los años sesenta como un lenguaje para tratamiento de listas y desde entonces se ha venido utilizando ampliamente en el campo de la inteligencia artificial, que implica la continua búsqueda y comparación de listas de datos, relaciones y respuestas. A diferencia del BASIC, que tiende a resaltar la importancia del flujo del programa a través de una secuencia de instrucciones y procedimientos, el LISP es un lenguaje "funcional", en el cual el juego de órdenes elementales se puede construir para conformar funciones más sofisticadas con nombres determinados por el programador. Por ejemplo:

```
(SETQ LISTA1 '(4 7 2 5 1))
```

crea una lista denominada LISTA1 cuyos elementos son los números (4 7 2 5 1).

```
(CAR LISTA1)
```

da el primer elemento de la lista LISTA1 (4, en este caso).

```
(CDR LISTA1)
```

da la lista LISTA1 eliminando el primer elemento:(7 2 5 1), en este caso.

```
(SETQ LISTA1(CDR LISTA1))
```

convertirá la lista de LISTA1 en una copia de sí misma excluyendo su primer elemento.
El LISP también se presta para las aplicaciones "repetitivas": problemas que, después de aplicárseles una función simple, quedan reducidos a un problema menor pero idéntico.

FORTH-CASTELLANO
CASTELLANO-FORTH

El FORTH se parece al LOGO en cuanto que es un lenguaje funcional interactivo, con la importante diferencia de haber sido el primer lenguaje, aparte del BASIC, que se incorporó a un ordenador personal (el Jupiter Ace). El lenguaje consta de varias funciones definidas, denominadas "primitivas", y posee la capacidad de definir funciones nuevas desde el punto de vista de éstas. En FORTH las operaciones matemáticas son "orientadas en pila", lo que significa que a la memoria del ordenador se la trata como una lista de datos que se expande y se contrae, lo que da por resultado que la última operación siempre esté al comienzo de la lista. Otra consecuencia de la "orientación en pila" es que no se utiliza la notación algebraica. En vez de escribir (12 + 4)/2 para hallar la media de 12 y 4, en FORTH usted debería escribir 12 4 + 2/, que es la misma suma que antes expresamos en notación algebraica, expresada ahora en notación polaca invertida.
Todo esto hace que el FORTH sea un tipo de lenguaje muy diferente, que obliga a aplicar un criterio muy distinto para solucionar los problemas y para efectuar los procesos informáticos. En la jerarquía de los lenguajes de alto nivel, se encuentra casi un paso más abajo.
Este fragmento en FORTH define dos palabras nuevas denominadas SHOUT y CHORUS:

```
:SHOUT (imprime "SHAZAM!")
   ." SHAZAM!";
:CHORUS (utiliza SHOUT en un bucle)
   0 DO SHOUT LOOP;
```

Ahora digitar "n" CHORUS hará que SHAZAM! se imprima "n" veces en la pantalla.

LOGO-CASTELLANO
CASTELLANO-LOGO

El LOGO lo desarrolló un psicólogo que se hallaba trabajando en "inteligencia artificial" mientras impartía clases a sus alumnos. Se asemeja al FORTH tanto en su interactividad como en la utilización de funciones "primitivas" que se pueden incorporar a las funciones definidas por el usuario. Pero encarna el principio fundamental de que la forma de aprender algo consiste en enseñarle a otro (en este caso al ordenador) cómo hacerlo. Se considera un lenguaje innovador que creará una forma completamente nueva de enseñarles a pensar a los niños.
Al LOGO por lo general se lo denomina lenguaje de "tortuga", porque se lo suele utilizar para controlar a un pequeño robot con ruedas denominado "tortuga" (véase p. 34).
El siguiente fragmento en LOGO dibuja una casa simbólica como un cuadrado de dimensiones especificadas con un triángulo encima:

```
TO TRIANGLE LENGTH
  REPEAT 3(FORWARD:LENGTH RIGHT 120)
END
TO SQUARE LENGTH
  REPEAT 4(FORWARD:LENGTH RIGHT 90)
END
TO HOUSE LENGTH
  RIGHT 30
  TRIANGLE:LENGTH
  LEFT 90
  SQUARE:LENGTH
END
```

Ahora al digitar HOUSE 15 se dibujará una "casa" que tendrá una longitud lateral de 15 unidades.

En estas páginas le ofrecemos un panorama de los lenguajes de programación más comunes disponibles para ordenadores personales. Tal como ocurre con los idiomas corrientes, mientras más lenguajes de programación domine, más fácil le resultará aprender uno nuevo.

Hacia el firmamento

En astronomía los ordenadores cumplen principalmente dos funciones: almacenar datos de los objetos observados y calcular su posición para facilitar la alineación precisa del telescopio

Marcus Wilson-Smith

Luz guía
La astronomía óptica se simplifica considerablemente cuando se puede formular una predicción exacta de la situación de una determinada estrella o planeta en un día determinado. Un ordenador personal puede datar las localizaciones de los astros y efectuar con rapidez los cálculos necesarios. Con la adición de los motores a impulsos, el ordenador puede incluso determinar la posición del telescopio

Cuando mira el cielo nocturno y observa la estrella más cercana, en realidad la está viendo tal como era hace cuatro años, porque ése es el tiempo que tarda en llegar a nosotros la luz desde Alfa de Centauro. Durante esos cuatro años el microordenador ha pasado de ser una novedad extraña y cara a ser un periférico habitual en los hogares del mundo. La astronomía es una ciencia que utiliza al máximo el potencial del ordenador personal para manipular datos, efectuar cálculos y para la robótica.

Al estudiar el cielo nocturno el astrónomo se enfrenta con tres tipos principales de problema: la observación inicial del objeto celeste, la manipulación de los datos obtenidos a través de la observación y el análisis significativo de esos datos. El ordenador constituye un medio auxiliar muy útil en cada una de estas áreas.

Comencemos por considerar cómo puede ayudar un ordenador personal a efectuar los cálculos necesarios para construir la herramienta básica del astrónomo, el telescopio. La disposición y el diseño de las lentes y los espejos de un telescopio son decisivos para la calidad de la imagen final y se pueden calcular matemáticamente para obtener los mejores resultados posibles. A los astrónomos aficionados a menudo les gusta construir sus propios sistemas ópticos, pero antes de que los ordenadores personales tuvieran una difusión tan amplia, con frecuencia era mucho más rápido y más sencillo armar un diseño experimental que efectuar los laboriosos cálculos necesarios para los diagramas de rayos de luz. Con un ordenador ahora se pueden realizar en unos pocos minutos cálculos que antes podrían haber tardado una semana.

También el ordenador nos puede ser de gran ayuda para localizar una estrella en el firmamento. Las estrellas no son objetos fijos: siguen trayectorias a través del cielo en el curso de la noche (efecto provocado por la rotación de la Tierra) y sus posiciones en el cielo están sujetas a las variaciones de las estaciones. El procedimiento que se sigue para localizar una estrella es similar al sistema de latitud y longitud que se emplea en la geografía terrestre. Si imaginamos un sistema de coordenadas proyectadas a través de la superficie interior del cielo nocturno, cada objeto celeste se puede localizar mediante dos coordenadas denominadas *declinación* y *ascensión recta*.

Todos los objetos se pueden señalar después en un mapa estelar de acuerdo a las coordenadas de cada uno, y se pueden combinar los diversos mapas para elaborar un atlas del firmamento. Este tipo de atlas permite observar los planetas y otros objetos que se mueven en contraste con el fondo de las estrellas fijas, o bien descubrir objetos celestes completamente nuevos, como los cometas. Estos atlas estelares en la actualidad están siendo incluidos en bases de datos para ordenadores personales. Estos programas de bases de datos también incluyen información relativa a la brillantez o luminosidad de cada uno de los cuerpos celestes, la calidad espectral de la luz que emiten (obtenida cuando se la hace pasar a través de un prisma o se la analiza con un espectrómetro) y el tipo y edad de la estrella. Toda esta información se puede visualizar en el televisor o el monitor del ordenador, en forma de mapas que muestran el cielo desde cualquier latitud en el momento que se determine.

Debido a la rotación de la Tierra, las estrellas desaparecen de la vista en unos pocos minutos, incluso para los telescopios de gran campo visual. Si

se utiliza un telescopio que pueda enfocar sólo un área estrecha del cielo, resulta esencial desviar el telescopio de manera continua para compensar el movimiento de la Tierra. Durante muchos años se han empleado motores de acción mecánica, pero recientemente el aficionado ha tenido acceso a los sistemas controlados por ordenador. El telescopio está montado sobre un eje "ecuatorial" que señala el norte verdadero, y el motor hace rotar el eje exactamente a la velocidad a la que se mueve la Tierra, con lo que se consigue mantener al objeto dentro del campo visual. Acoplados al eje y al telescopio hay codificadores y digitalizadores axiales, que le envían al ordenador una señal digital para las coordenadas celestes y controlan la actividad del motor de actuación. De esta forma, se puede dejar que el telescopio, bajo el control del ordenador, siga durante toda la noche la pista de una estrella tenue, mientras su imagen se va componiendo lentamente en una placa fotográfica. En Estados Unidos existe un adaptador llamado Celestial Navigator Mk II, que conecta un telescopio con el ordenador personal a través de interfaces de ampliación.

En ese mismo país, algunos astrónomos aficionados han llegado incluso a adaptar las más recientes tecnologías de reconocimiento y de síntesis de voz al campo de la astronomía. Al ordenador se lo programa para que reconozca ciertas palabras-órdenes; de este modo, si un observador penetra en el edificio y dice "abrir", la cúpula se desliza y se abre; con una ulterior orden ("hacer rotar la cúpula") se consigue que los motores la hagan girar sobre sus ruedas. La síntesis de voz también es muy útil en la oscuridad de un observatorio para hacer que el ordenador produzca una salida de información. Podría, por ejemplo, contar en voz alta las señales de tiempo para ayudar al observador a tomar exposiciones fotográficas exactas.

Los astrónomos profesionales utilizan telescopios que registran parte del espectro electromagnético además de la luz, como ondas de radio o rayos X. Con el aumento de tamaño de la apertura del telescopio y el consiguiente incremento en el peso, los problemas de ingeniería se volvieron muy críticos. En 1964, el Jodrell Bank Mark II, disco elíptico de 38 por 25 metros situado cerca de Manchester, se constituyó en el primer telescopio que utilizó un ordenador digital para convertir las coordenadas celestes en las instrucciones que necesitaban los diversos motores de actuación; el proceso se efectuaba continuamente cuatro veces por segundo.

Los ordenadores también ayudan al telescopio al tener en cuenta aquellos cambios atmosféricos que, como las variaciones de temperatura y humedad, refractan y desvían la luz mientras ésta atraviesa la atmósfera. Asimismo, pueden compensar y corregir las distorsiones de la imagen recibida a través del telescopio, en virtud de varias técnicas destinadas a mejorar la imagen.

La astronomía no sólo se sirvió de los ordenadores sino que también ha contribuido al desarrollo de la ciencia informática. El lenguaje FORTH lo creó un astrónomo, Charles H. Moore, en 1971 en el observatorio de Kitt Peak (Arizona) para controlar los radiotelescopios y procesar datos.

En la actualidad están apareciendo libros sobre programación personal destinados a los entusiastas de la astronomía e incluso se edita una revista, *Apex*, en la cual los usuarios de ordenadores personales publican e intercambian programas. Se han escrito programas para calcular las fechas de la Pascua de Resurrección, la conversión de las fechas históricas al calendario juliano, la salida y la puesta de la luna y tablas diarias para determinar el sitio exacto del firmamento donde los astrónomos esperan ver el regreso del cometa Halley en 1986.

Los mejores, los de una escuela de humanidades
Los alumnos y el personal docente de la Kettering Grammar School, de Northamptonshire (Gran Bretaña), gozan desde hace tiempo de gran estima por el trabajo que realizan en el campo de la astronomía, especialmente en lo que se refiere al seguimiento de satélites. En varias ocasiones, la KGS ha sido la primera estación de observación en detectar la presencia de un nuevo satélite en órbita

John Drisdale/Colorific

Estrellas con radio
A raíz del descubrimiento de la presencia de objetos de galaxias remotas que emiten radio, se construyó este gigantesco radiotelescopio en Jodrell Bank, al suroeste de Manchester. Los radiotelescopios, que operan a modo de inmensas antenas, pueden detectar objetos invisibles hasta para los telescopios ópticos más potentes

Paso a paso

Cuando hay que medir el movimiento lineal o angular por medio de un sensor óptico, la codificación binaria no es fiable: por ello se inventó el código de Gray

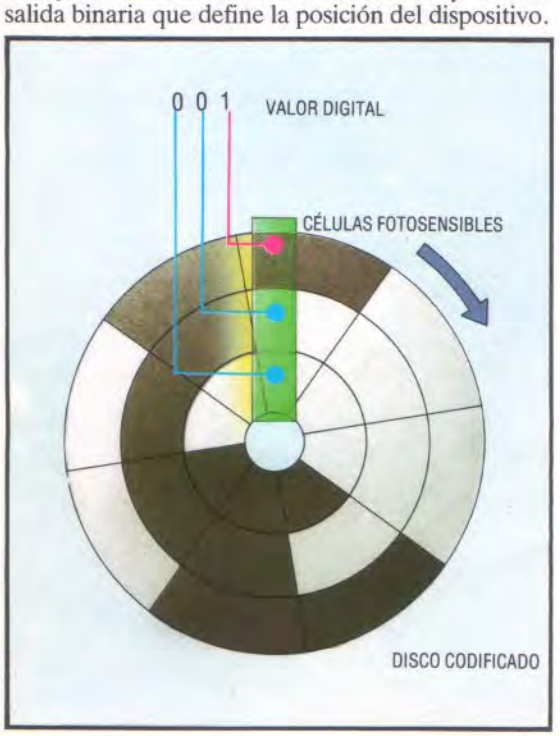
Extraños fuera
Esta tabla muestra los equivalentes en código de Gray de los decimales del 0 al 15.

Decimal	Binario	Cód. Gray
0	0000	0000
1	0001	0001
1	0010	0011
3	0011	0010
4	0100	0110
5	0101	0111
6	0110	0101
7	0111	0100
8	1000	1100
9	1001	1101
10	1010	1111
11	1011	1110
12	1100	1010
13	1101	1011
14	1110	1001
15	1111	1000

Hay muchas tareas en las que se ha de determinar y transmitir con gran precisión al ordenador la posición física de un objeto móvil. En robótica, por ejemplo, el ordenador ha de estar enterado de las posiciones y las orientaciones de todos sus miembros; y en la manipulación de herramientas por máquinas controladas por ordenador, se debe determinar exactamente la posición de la mesa fresadora. ¿Cómo se puede convertir una posición en un valor binario para ser procesado por ordenador?

Uno de los métodos consiste en utilizar un sistema analógico. Éste puede implicar una conexión del dispositivo móvil a una resistencia variable y el paso del voltaje resultante a un conversor de analógico a digital (o directamente a la puerta analógica, en caso de que su ordenador poseyera una). Sin embargo, este sistema no es muy preciso y los componentes mecánicos del mismo son susceptibles de gastarse y romperse.

La alternativa consiste en imprimir un código binario en el dispositivo móvil y leérselo directamente al ordenador. El código generalmente se imprime como un patrón de bloques negros y blancos a lo largo de la parte superior del dispositivo y se lee mediante una fuente luminosa que brilla en un patrón junto con una línea de células fotosensibles, cada una de las cuales es responsable de un dígito del patrón binario. A medida que se mueve el objeto sobre el que se está trabajando, el patrón situado debajo de las células luminosas cambia y da una salida binaria que define la posición del dispositivo.



Kevin Jones

Ángulo visual
La posición angular de un dispositivo se le puede leer al ordenador por medio de un disco que lleva impreso un código. En el dispositivo hay una luz que brilla y una línea de células fotosensibles detecta el patrón. Se produce una señal digital en paralelo que cambia a medida que se mueve el dispositivo. El problema de utilizar como código el binario, es que si el disco se detiene en la juntura entre dos valores se puede producir un resultado sin ningún significado. El código de Gray da solución a este problema

Además de las disposiciones lineales, también se utilizan patrones radiales para codificar el movimiento angular, como el de la articulación de un brazo robot.

Surgen problemas, sin embargo, cuando el dispositivo se mueve desde un código binario a otro y, especialmente, si tiene que descansar a medio camino entre ambos. La precisión de la impresión tiene una tolerancia finita, y cuando el dispositivo se detiene en una juntura entre dos códigos, las células luminosas podrían optar por leer cualquiera de los dos. Si la pieza de trabajo llegara al punto de descanso con las células luminosas brillando en la juntura entre la posición binaria 11 (1011) y 12 (1100), por ejemplo, entonces sólo se podría confiar en el bit más significativo (es decir, el situado más hacia la izquierda), para convenir la salida correcta, mientras que la lectura de los valores de las otras tres células luminosas sería conflictiva. En algunas situaciones cambian todos los bits, como en la juntura entre el siete (0111) y el ocho (1000) binarios, de manera que hasta la más insignificante imprecisión de impresión produciría lecturas incorrectas en todas las células. El resultado podría ser un valor totalmente falso para la posición y el ordenador no tendría posibilidad alguna de saber que esto había sucedido. Las consecuencias podrían ser desastrosas.

Lo que se necesita, por consiguiente, es un sistema de cuenta alternativo al binario, en el que a cada incremento sólo se modifique un bit. Esto significaría que en una juntura cualquiera sólo un bit podría ser dudoso, y que la salida sería errónea en no más de una posición. Este código alternativo se denomina código de Gray y está determinado exclusivamente por estos requerimientos: moviéndose de un valor al valor siguiente, cambia sólo un único bit y éste habría de ser el bit situado más hacia la derecha, lo cual daría un único patrón. De manera que si empezamos con 0000, como en binario, el número uno se representará por 0001. Sin embargo, para representar dos, debemos cambiar el segundo bit para obtener 0011. Para tres, ahora se puede cambiar el primer bit, obteniendo 0010. Observe la diferencia con la secuencia binaria para los mismos números: 0000, 0001, 0010, 0011.

El panel muestra este proceso ampliado hasta el equivalente del decimal 15, junto con los equivalentes en binario a modo de referencia. Como ejercicio, usted puede calcular los valores superiores en código de Gray.

Los ordenadores se podrían diseñar para que internamente efectuaran la aritmética y las funciones en código de Gray, pero ello sería ineficaz e innecesario. Por lo tanto, se necesita algún medio para convertir de Gray a binario, que se pueda llevar a cabo en hardware o en software.

Apple IIe

Muchos se preguntan la razón de su prolongada vigencia; se debe a que el Apple II es aún uno de los microordenadores más versátiles y, además, dispone de más accesorios que ningún otro

En muchos sentidos, el Apple II puede ser considerado como un pionero; si bien no fue el primer microordenador que apareció en el mercado, fue a raíz de su comercialización que capacidades como color, gráficos de alta resolución y sonido incorporado se hicieron accesibles al usuario de ordenadores personales. Sin embargo, aunque estas configuraciones del Apple son importantes, las realmente significativas, que le permitieron (y le permiten) alcanzar tan extraordinarias cotas de popularidad, resultan mucho menos patentes a primera vista.

La cualidad más manifiesta fue el alto nivel de su documentación, elaborada con el propósito de que los usuarios se hicieran una idea lo más clara posible de todos los aspectos de la máquina. Este hecho contradecía abiertamente la actitud de algunas empresas informáticas de entonces, que mantenían (y en algunos casos siguen manteniendo) un silencio estricto acerca del interior de sus productos. Como resultado directo de esta información de tan fácil acceso, se difundió otra importante configuración del Apple II.

Esta segunda capacidad era la fila de "conectores de ampliación" que ocupa la sección trasera del tablero de la máquina. Es precisamente la adaptabilidad de estas ranuras lo que ha hecho posible la amplia gama de accesorios para la máquina que se encuentran en el mercado. A su vez, esta diversidad le ha proporcionado al Apple una versatilidad extraordinaria.

Muchos ordenadores, por supuesto, poseen alguna forma de conector para ampliación (por lo general uno o dos), pero a menudo con una zona de memoria bastante pequeña. El Apple posee siete en la última versión (el IIe) y ocho en los modelos anteriores (II y II+), cada uno de los cuales aparece para la CPU como dos secciones de memoria (una bastante pequeña, y la otra, muy útil, de 2 Kbytes).

Por consiguiente, una ficha para periféricos que se enchufe en el Apple podrá tener 2 048 bytes para controlar el programa. Ello hace que utilizar una ficha de este tipo resulte muy sencillo, puesto que no es necesario conectar programas activadores especiales ni reescribir el sistema operativo.

Actualmente existe una extensa selección de fichas para el Apple II, que van desde interfaces para input/output relativamente sencillas hasta unidades muy avanzadas, como lápices ópticos y grandes fichas de RAM que pueden ampliar la memoria a un Megabyte o más. No obstante, dado que el 6502 puede direccionar sólo 64 Kbytes, la memoria está dispuesta en "bancos" de 64 Kbytes, que se seleccionan uno cada vez. Incluso se pueden enchufar ordenadores con avanzados chips de CPU de 16 o 32 bits, para funcionar en paralelo con el procesador 6502 de la máquina, con lo que se puede crear un sistema que posea un potencial igual (o superior) al de un miniordenador.

Naturalmente, se ha desarrollado una amplísima gama de software para aprovechar esta riqueza de hardware, y en la actualidad la biblioteca de programas del Apple debe de ser, con toda probabilidad, la más grande del mundo, mayor incluso que la lista de programas que utilizan el sistema operativo estándar CP/M. De hecho, la biblioteca de CP/M se puede incluir en la lista del Apple, porque aunque este ordenador posea un procesador 6502, puede ejecutar programas CP/M con la ayuda de uno de los accesorios más populares: una ficha Z80. (El CP/M funciona solamente con un microprocesador Z80.) Una vez enchufada en una ranura del Apple, la máquina funcionará como un ordenador CP/M normal.

Aunque bastante pequeño y de aspecto modesto, el Apple II es una máquina que suele ser objeto de superlativos. Es el ordenador personal que posee mayor número de sistemas operativos diferentes (11), de lenguajes (al menos 27) y editores de texto (una docena o más) que ninguna otra máquina.

La historia de este ordenador está lejos de haber terminado; en el primer semestre de este año se lanzará al mercado un sistema operativo para la máquina completamente nuevo. Se llamará ProDOS y posee muchas configuraciones que harán que el Apple II supere en rendimiento a algunos sistemas más caros.

Chris Stevens

El teclado del Apple
Ha sido diseñado de conformidad con los estándares más elevados, teniendo presente el tratamiento de textos. Las teclas están adecuadamente esculpidas y el corte transversal forma un arco para comodidad de uso. Las dos teclas marcadas con el logotipo de Apple a un lado de la barra espaciadora son teclas de control que se utilizan en programas aplicativos. A primera vista, la tecla RESET (restablecer) podría parecer que está demasiado cercana a la tecla DELETE (borrar). No obstante, para restaurar el ordenador se debe mantener pulsada CTRL y después apretar RESET

Monitor
El Apple IIe funciona con cualquier monitor especializado, pero la propia unidad de Apple es la que mejor armoniza con su diseño físico. Puede visualizar 80 columnas de texto y está equipado con un filtro antideslumbrante para reducir la fatiga visual

Chris Stevens

CPU 6502

Conector para altavoz

Conector para ampliación auxiliar
En el Apple IIe, la ranura 3 se duplica mediante esta ranura ligeramente más grande, que posee todas las señales de bus normales, así como algunas extras para manipular el segundo banco de 64 Kbytes que se puede agregar

RAM
El Apple IIe posee una RAM de 64 Kbytes lineales, retenidos en ocho chips 4164, pero el trazado de mapas es bastante más complejo, porque las direcciones de los chips para periféricos están situadas en el medio de la RAM

ROM de video
Los caracteres que se imprimen en la pantalla los genera esta ROM, que existe en diversos juegos de lenguajes diferentes

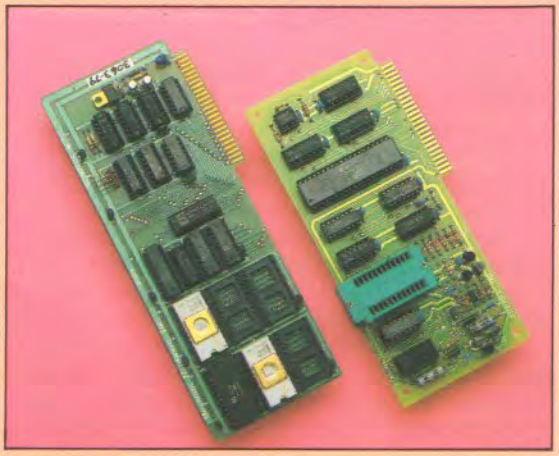
ROM del teclado
El trazado del mapa del teclado está controlado por esta ROM, que se sustituye para satisfacer las necesidades de los diferentes países; Francia, por ejemplo, puede tener teclados AZERTY

Conector para relleno del teclado
El Apple II se puede equipar con un relleno de teclado numérico para la entrada de datos, que se enchufa aquí

Conector para el teclado

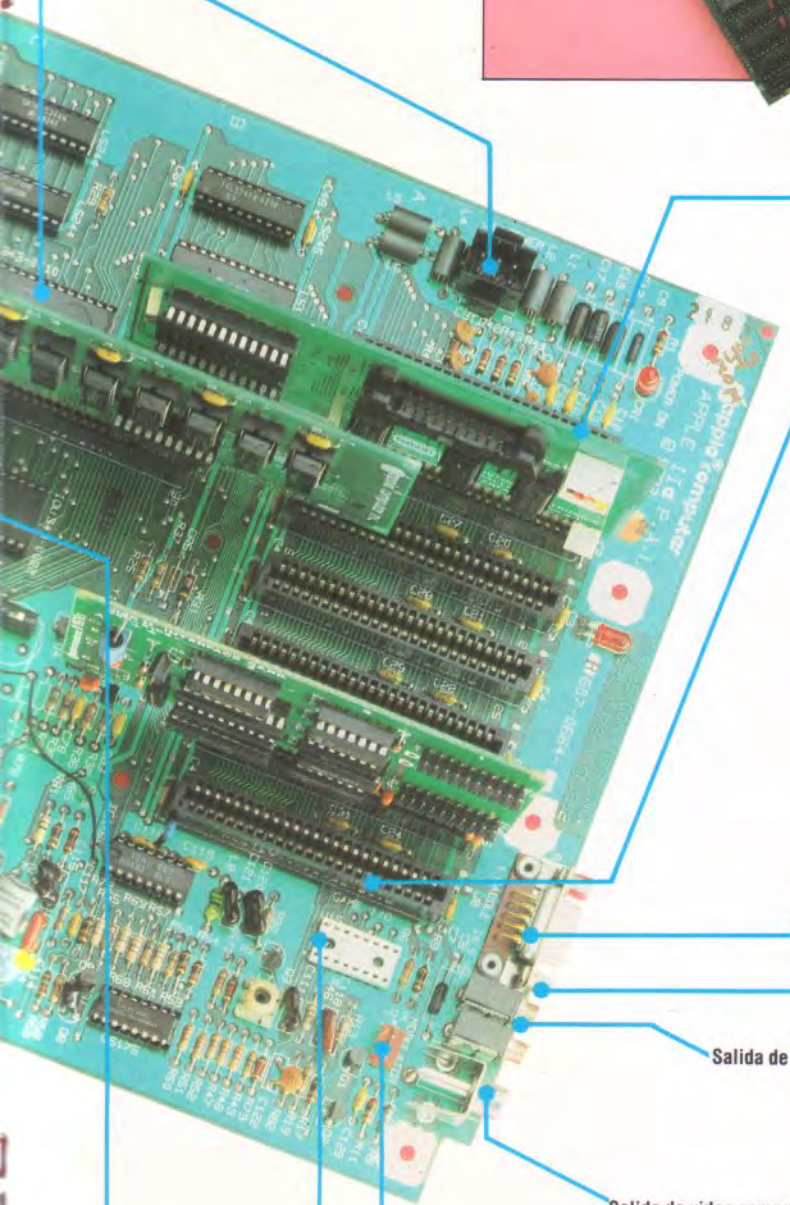
Unidad de disco
Los discos de Apple no son inteligentes, de modo que se debe acoplar una ficha controladora en el tablero principal del ordenador. Hoy, los 143 Kbytes de capacidad son muy pocos, pero son suficientes para muchas aplicaciones

Blower EPROM y Ficha ROM
La amplia gama de herramientas de desarrollo disponibles para el Apple lo hacen ideal para utilizarlo como un sistema de desarrollo para nuevos programas. El blower EPROM se puede emplear para producir unidades EPROM, que después se pueden insertar en una ficha ROM. Cuando se haya perfeccionado el programa, se lo colocará en forma de ROM si es que se va a vender en gran número

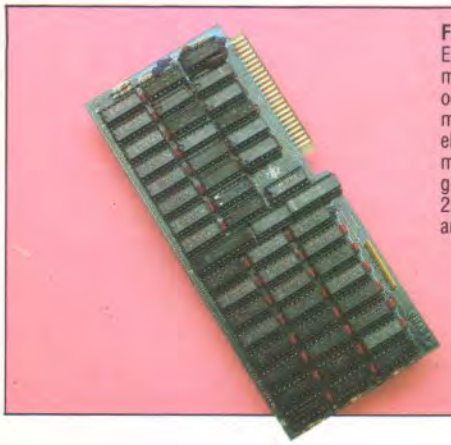


Unidad de administración de memoria
Una de las dos ULA, que constituye la principal diferencia entre el Apple II y el Apple IIe. Ésta controla la pantalla de 80 caracteres así como la segunda RAM o banco de 64 Kbytes

Conector de potencia



Ficha RAM de 256 Kbytes
El Apple posee un avanzado mapa de direcciones, lo que ocasiona un acceso ligeramente más complicado. Sin embargo, ello permite utilizar en la máquina bloques de RAM muy grandes. Esta ficha transporta 256 Kbytes, ¡pero se puede ampliar hasta 1 Mbyte!



Ranura 1
Normalmente está ocupada por una interface para impresora en paralelo

Ranura 7
Las señales especiales de video sólo están disponibles en esta ranura, de modo que aquí se suelen encontrar las fichas relacionadas con el video, como lápices ópticos y moduladores de color

Enchufe DIL para juegos
Una de las configuraciones más innovadoras del Apple fue la conexión para juegos, que daba una forma de entrada analógica mínima pero útil

Entrada de cassette

Salida de cassette

Salida de video compuesto

Conector auxiliar para video

Unidad de input/output
La ULA que manipula el direccionamiento del conector auxiliar

Enchufe-D para juegos
El enchufe normal de 16 patillas para conexión de juegos es demasiado frágil para utilizarlo a diario, por lo cual el Apple IIe está equipado con un pequeño enchufe-D en paralelo

APPLE IIe

DIMENSIONES
460 x 385 x 115 mm

VELOCIDAD DEL RELOJ
1 MHz

MEMORIA
ROM de 16 Kbytes
RAM de 64 Kbytes
Ampliables a 128 Kbytes o más

VISUALIZACION EN VIDEO
24 líneas de 40 caracteres, sólo monocromática. Gráficos de baja resolución de 48 x 40 en 16 colores. Gráficos de alta resolución de 192 × 280 en 6 colores

INTERFACES
Cassette, video compuesto, 7 ranuras de ampliación, conexión para juegos

LENGUAJE SUMINISTRADO
BASIC Applesoft

OTROS LENGUAJES DISPONIBLES
La mayoría de los lenguajes alternativos comunes, así como algunos menos conocidos

VIENE CON
Manuales de instalación y de BASIC, cable para TV

TECLADO
62 teclas de gran calidad

DOCUMENTACION
La documentación que acompaña a la máquina es de un nivel muy elevado, si bien el material avanzado necesario para una comprensión absoluta de la máquina se ha de adquirir por separado y es bastante caro. Existe gran número de libros que satisfacen todos los niveles de interés y, en este sentido, probablemente se trate de la máquina que ha sido estudiada con mayor profundidad

Ficha de I/O para fines generales
En algunas ocasiones una ficha puede tener tantas posibles funciones que una ROM-controladora limitaría al usuario y se convertiría en un riesgo. Esta ficha, que posee dos versátiles adaptadores para interface 6522, constituye un ejemplo de ello. Posee 40 líneas de I/O controlables independientemente, dos registros de cambio que se utilizan para convertir datos de modalidad en paralelo a en serie, y cuatro cronometradores de 16 bits

Chris Stevens

Discos Winchester

Estos discos rígidos requieren para su funcionamiento unas condiciones de máxima limpieza. Sellados en el interior de su carcasa, proporcionan una gran capacidad y un rápido acceso

Aunque en el transcurso de los últimos años los ordenadores personales han asimilado muchas de las configuraciones de las pequeñas máquinas de gestión, hay un área de la tecnología de los ordenadores en la que carecen relativamente de sofisticación: el espacio del almacenamiento en disco. Mientras que el usuario de un ordenador personal estaría muy satisfecho de tener un único disco flexible capaz de retener 100 Kbytes de datos, los requerimientos de las máquinas contables exigen un espacio considerablemente mayor. Una pila de discos flexibles, con toda la información empresarial diseminada entre ellos, no constituye ninguna solución para esta gran necesidad de almacenamiento; y, por consiguiente, se desarrollaron discos rígidos, capaces de almacenar cantidades mucho mayores de datos. La IBM, en la década de los sesenta, inició los primeros trabajos para crear estos discos rígidos. Dado que los discos originales poseían una capacidad de almacenamiento de 30 Megabytes en cada unidad, se los apodó "30/30" y a partir de allí, por analogía con el rifle, discos "Winchester".

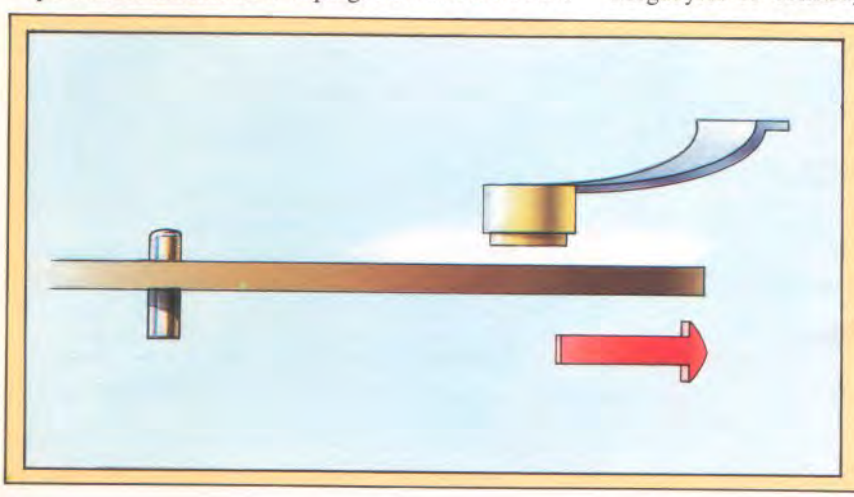
Los Winchester utilizan discos rígidos en lugar de los flexibles de plástico empleados en los discos flexibles. Ello permite incrementar el número de pistas del disco desde un máximo operativo típico de 96 pistas por pulgada de los discos flexibles a varios centenares en los Winchester. Con la creciente sofisticación de la tecnología del disco se ha hecho algo común disponer de cinco, 10 o incluso 20 Megabytes de almacenamiento en una caja de las mismas dimensiones que una unidad de disco flexible de 5 ¼ pulgadas.

Ahora los usuarios de ordenadores personales están comenzando a beneficiarse de la tendencia a emplear los discos rígidos gracias a la disponibilidad en el mercado de los microfloppies Sony de 3 ½" y los Hitachi de 3". Éstos son discos semirrígidos y pueden almacenar tanta información como cualquier disco flexible de 5 ¼ pulgadas. El BBC Micro y el Oric-1 son dos de las primeras máquinas que dispusieron de este tipo de dispositivos como accesorios, mientras que ordenadores como el Apricot de ACT los incluyen ya como accesorios estándar.

Este incremento de la densidad de almacenamiento ha creado, sin embargo, otros problemas. La precisión que exige el mecanismo de posicionamiento de la cabeza, por ejemplo, requería un método completamente nuevo para hacerla mover. La solución a este problema se halló en la industria de audio: con frecuencia las cabezas Winchester se colocan en posición mediante una bobina electromagnética del tipo de las que llevan los altavoces. Al hacer pasar una corriente eléctrica a través de una bobina se genera un campo magnético que a su vez hace que un "brazo" de hierro maleable situado en el centro de la bobina se mueva una distancia precisa. Acoplando la cabeza al extremo de este "brazo" (por supuesto, convenientemente aislada de todo efecto magnético), se consigue que ésta se desplace sobre la superficie del disco con rapidez y precisión.

La cabeza "vuela" a través de la superficie del disco sobre un colchón de aire, sin llegar a tocarla nunca. Ello reduce significativamente el desgaste y el deterioro de los discos, pero implica que éstos han de estar sellados en cajas herméticas para evitar el contacto del polvo y otros cuerpos extraños. Por lo general, se hallan fijos dentro de la unidad y no se pueden retirar, si bien en la actualidad están empezando a aparecer discos Winchester con cartuchos separables. Estos cartuchos suelen ser autosoldables y se abren sólo para permitir que la cabeza acceda a la superficie del disco cuando se inserta el cartucho en la unidad. Para evitar que entre el polvo, se suele mantener la presión del aire dentro del cartucho por sobre la presión exterior, y todo el aire que se bombea en la unidad se filtra antes.

Otra ventaja de los discos rígidos es que se pueden utilizar discos múltiples. Un Winchester de 10 Megabytes se construye sencillamente colocando



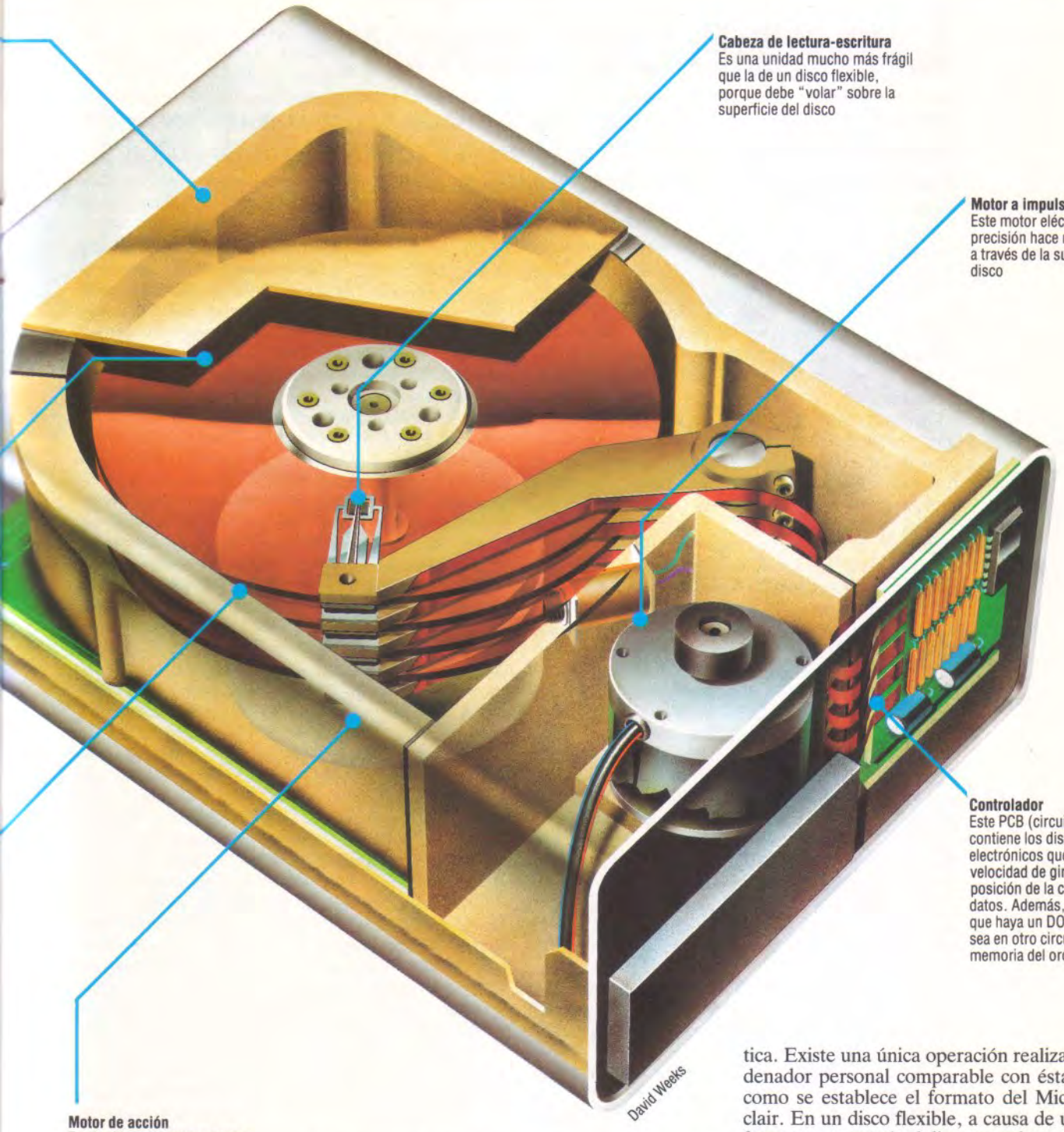
Cabezas flotantes
A diferencia de la unidad de disco flexible, en la cual la cabeza hace contacto con la superficie de grabación, en una unidad de disco Winchester la cabeza "flota" muy cerca de la superficie. El disco gira tan rápidamente que el "efecto Kelvin" crea un colchón de aire sobre el cual se apoya la cabeza. Si ésta "aterrizara" contra el disco, probablemente eliminaría la superficie de grabación magnética

Kevin Jones

Carcasa
La mayoría de las unidades Winchester vienen dentro de una carcasa de aleación colada, a la cual deben gran parte de su peso. Esta carcasa es necesaria para mantener a los componentes alineados con total precisión

Discos
Los Winchester de mayor capacidad sencillamente incorporan más discos en el mismo eje. El disco que aquí ilustramos posee cinco, pero la mayoría poseen dos o tres. Las cabezas de lectura-escritura están conectadas entre sí, de modo que sólo se puede leer un bloque cada vez

Sellado hermético
La maquinaria de la unidad está herméticamente sellada para impedir que partículas de polvo o humo, provenientes del exterior, "choquen" contra la cabeza



Cabeza de lectura-escritura
Es una unidad mucho más frágil que la de un disco flexible, porque debe "volar" sobre la superficie del disco

Motor a impulsos
Este motor eléctrico de gran precisión hace mover la cabeza a través de la superficie del disco

Controlador
Este PCB (circuito impreso) contiene los dispositivos electrónicos que controlan la velocidad de giro, determinan la posición de la cabeza y leen los datos. Además, es necesario que haya un DOS sofisticado, ya sea en otro circuito o bien en la memoria del ordenador

Motor de acción
En el mismo eje, que a su vez hace girar el disco, hay montados un motor de CD (corriente directa) y un pequeño generador. La salida del generador es una medida de la velocidad del motor y se alimenta de un circuito de control especial. Así es como se logra determinar con tanta precisión la velocidad

David Weeks

en la misma caja dos discos de cinco Megabytes. Se consigue administrar todo este espacio de almacenamiento dividiéndolo en una gran cantidad de secciones. Para mantener la compatibilidad con el software existente, que en el momento de ser diseñado estaba previsto que funcionara sólo con discos flexibles, estas secciones generalmente se aproximan a la capacidad de un floppy normal. Un disco Winchester se parece, al menos para el ordenador, a gran cantidad de unidades de disco flexibles separadas.

Cuando se le da el formato a un disco Winchester (es decir, cuando se marcan por primera vez las pistas y sectores), el DOS (véase p. 324) ha de ser capaz de saltarse los "sectores malos", los que posean defectos en la superficie de grabación magnética. Existe una única operación realizada en un ordenador personal comparable con ésta: la manera como se establece el formato del Microdrive Sinclair. En un disco flexible, a causa de un sector defectuoso, se rayaría el disco completo; mientras que en un Winchester, el programa para formato tan sólo apunta que hay un sector determinado inutilizable e impide su ulterior uso. Al fin y al cabo, si se dispone de cinco millones de bytes de espacio, ¿quién puede echar de menos unos pocos centenares?

Siguiendo las tendencias marcadas por los avances que se están produciendo en las otras áreas de la industria de los ordenadores, el disco Winchester se está volviendo más pequeño en tamaño, y ahora existe un microdisco Winchester de 5 Megabytes. Con el acelerado perfeccionamiento que están experimentando las unidades de almacenamiento en disco, la posibilidad de que un ordenador personal corriente configure un almacenamiento de datos de 10 Megabytes no está demasiado lejana.

Líneas de unión

Ahora ya podemos unir los subprogramas que procesarán nuestra agenda de direcciones computerizada y analizar la forma de lograr que el programa sea lo más sencillo posible

Aunque aún debamos dar forma final a muchos detalles del programa para la agenda de direcciones computerizada, la estructura global ya debería estar bastante clara. Llegados a este punto del desarrollo de un programa de cualquier dimensión, resulta útil dibujar un diagrama de bloques del programa y pensar en el flujo de actividades del mismo.

En este punto, el programador debería considerar también los aspectos del programa relacionados con la "interface humana" y la "imagen para el usuario". Pocos fabricantes de software dedican la debida atención a estos dos importantes conceptos.

La "interface humana", definida en términos sencillos, es la "ergonómica" del software, o el nivel de sencillez de su utilización. La "imagen para el usuario" se refiere a la forma en que éste percibe el programa que utiliza. Vamos a analizar estos conceptos en relación a nuestro programa tal como lo hemos desarrollado hasta el momento y estableceremos hasta qué punto podremos aplicarlos.

La tabla muestra los principales bloques del programa que hemos realizado hasta ahora. Convencionalmente, hemos utilizado nombres de seis caracteres para los procedimientos o subrutinas, de siete caracteres (incluyendo $) para las matrices, de cuatro caracteres para las variables numéricas simples y de cinco caracteres (incluyendo $) para las variables alfanuméricas simples. Para las variables locales (dentro de bucles, p. ej.) por lo general hemos empleado una única letra.

Cada uno de los grandes bloques del programa de la segunda columna necesita volver a dividirse en subunidades, y éstas se habrán de volver a refinar hasta que obtengamos todos los detalles suficientes para escribir el código verdadero en BASIC. Los procesos implicados en esta forma de "refinamiento por pasos" ya los hemos ilustrado en muchos de los bloques en capítulos anteriores de nuestro curso.

BLOQUES PRINCIPALES DEL PROGRAMA

	CREMAT	(crea matrices e inicia variables)
INICIL	LEARCH	(lee los archivos desde cinta o disco)
	ESBAND	(establece banderas y modifica variables)
PRESEN		(imprime mensaje presentación)
ELECCN	IMMENU	(imprime menú de opciones)
	ASOPCN	(asigna opción a OPCN)
	ENCREG	(localiza e imprime un registro)
	ENCNOM	(localiza nombres a partir de entradas parciales)
	ENCIUD	(localiza registros de una ciudad específica)
	ENCINI	(localiza nombres a partir de iniciales)
EJECUT	LISREG	(lista de todos los registros)
	INCREG	(agrega un registro nuevo)
	MODREG	(modifica un registro existente)
	BORREG	(borra un registro)
	SAPROG	(guarda un archivo y da salida al programa)

Suponiendo que todos los módulos del programa, o la mayoría de ellos, ya se han elaborado, codificado en BASIC y verificado de forma individual, ¿cómo se pueden unir entre sí para conformar un programa completo? La mejor manera de afrontar este problema consiste en guardar cada módulo en cinta o en disco, otorgándoles el mismo nombre de archivo que empleáramos en las notas sobre el desarrollo del programa. De modo que INCREG se puede escribir y verificar en la mayor medida posible, guardándola después bajo el nombre de archivo INCREG. Normalmente, cuando se carga un programa desde cinta o disco, utilizamos la orden LOAD, seguida del nombre del archivo, como en LOAD "INCREG". Sin embargo, esto tiene el efecto de limpiar toda memoria, de modo que si cargáramos INCREG y posteriormente cargáramos MODREG, todo el programa INCREG desaparecería.

Por fortuna, existe una solución parcial. La orden MERGE permite cargar un programa desde cinta o disco sin borrar ningún otro programa que estuviera ya en la memoria. Pero para ello existe una condición importante. Si cualquiera de los números de línea del programa cargado mediante MERGE fueran iguales a los números de línea del programa que ya estuviera en la memoria, los nuevos se escribirían sobre los antiguos y se produciría un caos. Las versiones de BASIC que disponen de la orden RENUM pueden obviar este inconveniente volviendo a numerar las líneas de un módulo de programa antes de guardarlo, de modo que cuando se utilice MERGE no se plantee ningún problema.

Lamentablemente, en los ordenadores personales muchas versiones de BASIC no poseen la orden RENUM y, por lo tanto, será necesario efectuar una cuidadosa planificación de los números de línea desde el principio. Cuando se haya elaborado un diagrama de todos los módulos principales del programa (como hemos hecho parcialmente en la tabla), se podrán asignar los números de línea con los que se inicia cada módulo. En aquellas partes del programa en que con toda seguridad se introducirán cambios o modificaciones, como son el programa principal o las partes del programa donde se manipula el archivo, se deben efectuar incrementos de 50, o incluso de 100, en la numeración de las líneas, con el fin de dejar abundante espacio para los agregados. En los módulos del programa menos susceptibles de sufrir modificaciones, como la rutina PRESEN, pueden realizarse incrementos de 10 en la numeración de las líneas. Introducir un buen número de líneas REM en blanco en el programa no sólo contribuye a que éste resulte más fácil de leer, sino que también permite que posteriormente se

puedan agregar sentencias adicionales o llamadas a subrutinas. Si el BASIC de su ordenador tampoco dispusiera de MERGE, entonces tendrá que digitar los diversos módulos tal como están escritos y guardarlos juntos.

Los bloques del programa de la tabla se han fusionado (MERGE) como una "ejecución de prueba" en el listado impreso aquí, con el fin de ilustrar los peligros latentes del enfoque "pruebe y vea" que fomentan lenguajes como el BASIC. No tiene ningún sentido digitar el programa entero sólo para descubrir que no funcionará, pero si usted ha guardado todas las rutinas de anteriores capítulos, y si el BASIC de su ordenador posee la orden RENUM, puede volver a numerarlas (RENUM) y después fusionarlas (MERGE) para producir un listado similar.

El primer bloque del programa principal es INICIL, del cual se espera que inicialice variables, dimensione matrices, lea archivos, les asigne los datos a las matrices, establezca banderas, etc. La subrutina *INICIL* se divide en *CREMAT* (para crear matrices, *LEARCH* (para leer archivos y asignarles los datos de las matrices correspondientes) y *ESBAND* (para establecer banderas, etc.).

Cuando se ha efectuado todo esto, el programa pasa a *PRESEN*, una subrutina que imprime en la pantalla un mensaje de presentación. La última parte de esta rutina espera a que el usuario pulse la barra espaciadora para que el programa continúe.

El programa prosigue luego con *ELECCN*. Ésta consta de dos partes: la primera presenta un menú de las opciones que ofrece el programa de la agenda de direcciones; la segunda acepta la entrada de la opción desde el teclado y le asigna el valor (numérico) a una variable denominada OPCN.

El valor de esta variable lo utiliza el siguiente bloque del programa, EJECUT, para seleccionar uno de entre nueve bloques del programa. Todos ellos, a excepción de SAPROG, necesitarán retornar a *ELECCN* después de haberse ejecutado, para que el usuario tenga la oportunidad de seleccionar otra opción. Ello no será necesario en caso de seleccionarse 9 (SAPROG), porque se supone que esta opción pondrá fin a la operación del programa.

El principal problema de este programa, tal como está, es que el flujo de control no es correcto. INICIL insiste en que leamos un archivo con todos los datos, exista o no este archivo. Si el programa se está ejecutando por primera vez, no se habrá dado entrada a ningún registro y en la cinta o el disco no habrá archivos de datos. Todo intento de abrir y de leer un archivo inexistente tendrá como resultado un mensaje de error y el programa no funcionará.

Lo que se necesita es que la rutina *LEARCH* sea llamada sólo por uno de los módulos EJECUT, y entonces sólo una vez en cada ocasión se ejecutará el programa. Esto sugiere que debería haber una bandera ARCH, establecida originalmente en 0, que se establezca en 1 una vez que se haya leído el archivo. Si se hubiera establecido en 1, impediría todo intento ulterior de leerlo. INCREG entonces buscaría siempre a través de todas las matrices para localizar el primer elemento vacío y escribiría allí la información. Casi con toda certeza este registro no estaría entonces en la secuencia de clasificación correcta, de modo que debería haber una bandera RMOD que durante la ejecución se estableciera en 1. La bandera RMOD también se habría de establecer en 1 si se ejecutaran MODREG o BORREG. Puede intentar escribir el código correspondiente para conseguir esto; o, si simplemente desea ejecutar el programa, cambie la línea 1310 para que retorne (RETURN).

Tanto agregar como borrar o modificar un registro significa que es probable que la secuencia de registros esté desordenada, de modo que cualquier módulo (ENCREG, por ejemplo) debería primero verificar RMOD para ver si se han realizado cambios. De ser así, podríamos insistir en una clasificación antes de efectuar alguna búsqueda, o bien emprender una búsqueda estéril en una pila. SAPROG verificará RMOD y llamará a la rutina de clasificación si estuviera establecida (en 1) antes de guardar los datos del archivo en cinta o en disco.

Los aspectos de "interface humana" del programa, que hemos mencionado anteriormente, se pueden dividir en las siguientes categorías:

Interface para el usuario
Imagen para el usuario
Recuperación de errores
Seguridad
Adaptabilidad

Al hablar de *interface para el usuario* nos referimos a la forma en que el usuario del programa se comunica con éste. Hemos optado por utilizar menús (en lugar de órdenes). Muchas personas prefieren las órdenes, pero lo que importa es que, cualquiera sea la forma de intercomunicación empleada, ésta sea coherente. En el curso de un programa no debe haber órdenes similares que realicen cosas diferentes en distintas partes del mismo. De ser así, el usuario habría de leer atentamente cada menú antes de efectuar alguna selección y entonces no se podrían desarrollar "reflejos".

Tal como está conformado, nuestro programa es pobre en este sentido: el mensaje de presentación se termina pulsando la barra espaciadora; el menú de opciones se termina automáticamente digitando cualquiera de las teclas de 1 a 9; y, en el caso de INCREG, la entrada de datos se termina (para cada campo) pulsando la tecla RETURN. Esta clase de incoherencia podría ser aceptable en un programa "casero", pero no lo sería en el software comercial.

La *imagen para el usuario* se refiere a la forma en que el usuario percibe el funcionamiento del programa. En ella influye de manera decisiva la calidad de la interface para el usuario. La mayoría de las operaciones que se llevan a cabo en el interior del ordenador están totalmente ocultas al operador de la máquina. El único medio de que dispone para reconocerlas es la pantalla. Para el programa que estamos desarrollando, la imagen para el usuario más adecuada sería la representación de una agenda de direcciones verdadera. Del mismo modo, la imagen para el usuario más apropiada en el caso de un programa de tratamiento de textos sería la de un trozo de "papel" (en la pantalla) en el que se pudiera mecanografiar. En este caso, lo ideal sería que las palabras mecanografiadas en negrita aparecieran en negrita en la pantalla, que el texto subrayado se viera realzado y que los párrafos justificados se observaran así al visualizarlos.

La perfecta imagen para el usuario sólo se consigue en muy raras ocasiones (ninguna agenda de direcciones verdadera esperaría que el usuario pulsara 1 ("PRESS 1") para hallar un nombre. No obstante, una buena imagen para el usuario implica unos trazados de pantalla bien diseñados y un pa-

trón coherente para las operaciones. Los avisos siempre deberían aparecer en la misma posición en la pantalla (algunos procesadores de textos muy conocidos, por ejemplo, visualizan algunos avisos en la línea superior de la pantalla y otros en la línea inferior, aparentemente al azar). Un programa con una buena imagen para el usuario también debería informarle a éste, en cualquier momento que lo deseara, en qué punto del programa se halla. Si usted estuviera en la modalidad INCREG, por ejemplo, debería haber un mensaje siempre visible que se lo indicara. Si acabara de dar entrada a un campo (para agregarlo a un nuevo registro), debería haber un mensaje que le indicara, por ejemplo, PULSE RETURN SI LA ENTRADA ES CORRECTA, DE LO CONTRARIO PULSE ESCAPE (lo que nos lleva al importante tema de la recuperación e información de errores, que trataremos luego).

Lo ideal sería que todo el formato apareciera en la pantalla, de modo que, por ejemplo, el registro visualizado fuera del mismo formato que un registro impreso por la impresora. Muchos programas incorporan "menús de ayuda" que le indican cómo debe proseguir cuando no está seguro de ello.

Lo deseable es que la imagen para el usuario de un programa sea lo más concreta posible (un papel para mecanografiar o una tarifa, por ejemplo), evitando recurrir a objetos como "subarchivos", "buffers", etc. En este sentido, muchos programas comerciales de base de datos no son todo lo eficientes que cabría esperar; el usuario ha de tener presente en todo momento que ciertas informaciones se encuentran en archivos o en campos ocultos y temporales. Estos factores tienden a transformar la utilización de un programa de estas características en un arduo esfuerzo intelectual.

La *recuperación de errores* es también un tema importante. ¿Qué sucedería, por ejemplo, si acabara de dar entrada al nombre de alguien y de inmediato se percatara de que ha cometido un error de digitación? ¿Tendría que seguir adelante y después llamar a MODREG para corregirlo, o el programa le ofrecerá la posibilidad de "abandonar" antes de ir más lejos? La mayoría de las versiones de BASIC informarán de los errores al dar entrada a un programa, ya sea cuando se da entrada a la línea errónea o cuando se ejecuta el programa. Sin embargo, esto no forma parte de la "interface para el usuario". El BASIC incluye varios mensajes que vuelven a solicitarle al usuario una entrada correcta en caso de que se hubiera realizado una errónea (p. ej., el aviso REDO —rehacer— cuando se hace una entrada inadecuada en respuesta a una sentencia INPUT).

La manipulación de errores posee dos facetas: la información sobre el error y la recuperación del error. Un programa muy conocido de tratamiento de textos, por ejemplo, posee una buena información de errores pero una pobre recuperación de ellos; si después de crear un largo documento usted intentara guardarlo en un disco que estuviera casi lleno, el programa le proporcionaría un útil mensaje: AGOTADO ESPACIO DISCO. Lamentablemente, no le permite al usuario recuperarse de este error: ¡no se puede dar formato a un disco nuevo sin destruir primero el texto que quizá le haya costado horas digitar!

Toda operación que pueda efectuar el usuario y que pudiera producir pérdida de datos (MODREG, por ejemplo) debería siempre ser objeto de alguna indagación antes de ser ejecutada. Siempre se deberían proporcionar mensajes como ESTO DESTRUIRA EL REGISTRO, ¿ESTA USTED SEGURO? (S/N). En el caso de un procesador de textos, un mensaje de esta índole sería: LA ORDEN "SAVE" NO CONSERVARA UNA COPIA DEL DOCUMENTO ANTIGUO. ¿LE PARECE BIEN? (S/N).

La manipulación de errores (detectarlos e informar de que se han producido) se debe tener en cuenta al diseñar un programa siempre que exista la posibilidad de una entrada de datos incorrecta, de una opción del menú equivocada, de órdenes erróneas y siempre que la información se haya de modificar o de guardar, especialmente si guardarla implica tener que escribir sobre datos antiguos.

El usuario también debe prestar atención a la *seguridad*: ¿qué sucedería con el programa o con los datos si se produjera un grave contratiempo (p. ej., un corte del fluido eléctrico)? Un procesador de textos bastante sofisticado incorpora un programa denominado RECOVER (recuperar), que, en el caso de que se produzca un grave percance (p. ej., un corte del fluido eléctrico, o que el usuario apague el ordenador sin antes haber guardado el documento), no permite que se pierda casi nada. No obstante, este tipo de técnicas avanzadas de programación se encuentran más allá de las metas de nuestro curso. Lo importante es lograr que sus programas sean lo más seguros posible, anticipándose a los errores garrafales en que se pueda incurrir —neutralizándolos de manera racional— y escribiendo rutinas concebidas para hacerles frente.

La *adaptabilidad*, es decir, la facilidad con que el usuario se puede familiarizar con el programa, también es importante. Por norma general, siempre se debe dejar abundante espacio entre los números de línea (en BASIC) e incorporar muchas líneas REM en blanco, donde posteriormente, en caso de ser necesario, se puedan incluir sentencias y GOSUB. Al crear matrices, se debe incorporar al menos una matriz innecesaria para posibilitar una futura ampliación. Una regla fundamental de la escritura de programas es que las exigencias futuras no se pueden anticipar. Lo único seguro es que un buen programa siempre se puede mejorar.

Complementos al BASIC

SPECTRUM

Véase el programa de la página 318 para la versión del Spectrum desde la línea 1300 a la 1370. El bucle del programa principal entre las líneas 3750 y 3780 funcionará en el Spectrum, pero podría crear problemas debido a que las teclas de este ordenador se repiten si se mantienen pulsadas durante más de una fracción de segundo. El manual del Spectrum recomienda que, para evitar este inconveniente, este tipo de bucle INKEY$ incluya una línea extra:

```
3755 IF INKEY$ <> "" THEN GOTO 3755
```

El Spectrum admite la función VAL(A$), pero romperá el programa si el primer carácter de A$ no es numérico; en este programa el problema se puede obviar mediante:

```
3790 LET OPCN = CODE A$ - 48
```

pero esta solución no es perfecta, funciona sólo cuando A$ es un carácter único (como debe ser en el programa). El Spectrum no admite ON...GOSUB, pero sí le permite escribir GOSUB (expresión numérica) así como sencillamente GOSUB (número de línea); la línea 4010 se puede reemplazar por:

4010 GOSUB (290+OPCN*20)

El Spectrum no admite RENUMBER

MERGE

No la admiten el Oric-1, el Commodore 64, el Vic-20, el Lynx, el Dragon 32 y el BBC Micro. No obstante, a menudo suele haber una forma de imitar la orden MERGE, que le podrían revelar algún usuario o alguna publicación especializada. El Lynx posee la orden APPEND, que le permite LOAD (cargar) un programa en el final de un programa de la memoria; esta orden puede reemplazar a MERGE siempre que el primer número de línea del programa en cinta sea mayor que el último número de línea del programa de la memoria.

RENUMBER

No la admiten el Spectrum, los Commodore y el Oric-1; en estas máquinas la renumeración se puede realizar línea a línea utilizando la instrucción EDIT.

INKEY$

Véase "Complementos al BASIC" de página 175.

OPEN CLOSE

Véase "Complementos al BASIC" de página 319.

PRINT CHR$(12)

Reemplazarla por CLS, excepto en el Commodore 64 y el Vic-20; en estas máquinas digite PRINT "shiftkey+CLRkey", que hará que entre las comillas aparezca un corazón de campo invertido o una S mayúscula.

ON... GOSUB

En la línea 4010, ON...GOSUB hará que el control pase a líneas inexistentes (310, 330, 350, etc.), lo que provocará la ruptura del programa. Estas líneas posteriormente se ampliarán para incluir las subrutinas para ejecución del menú; mientras tanto, dé entrada en el programa a lo siguiente, a modo de líneas "ficticias":

```
310 RETURN
330 RETURN
```

etc.

STR$

El Lynx no admite esta orden, de modo que reemplace la línea 9080 por:

```
9075 N = TAMA
9077 GOSUB 9500
9080 INDCAM$ (TAMA) = N$
```

e inserte esta subrutina:

```
9500 REM Para convertir N en N$, donde N
     es un entero
9510 LET N$ = " "
9520 IF N<0 THEN LET N$="-"
9530 N=ABS(N)
9540 X=10
9550 I=1
9560 FOR K=1 TO 8
9570 I=I*X
9580 R=K
9590 If I>N THEN K=8
9600 NEXT K
9610 FOR K=1 TO R
9620 I=I/X
9630 Z=INT(N/I)
9640 LET N=N-Z*I
9650 N$=N$+CHR$(48+INT(Z))
9660 NEXT K
9670 RETURN
```

```
10 REM "PROPRI"
20 REM *INICIL*
30 GOSUB 1000
40 REM *PRESEN*
50 GOSUB 3000
60 REM *ELECCN*
70 GOSUB 3500
80 REM *EJECUT*
90 GOSUB 4000
100 END
1000 REM SUBRRUTINA *INIC*
1010 GOSUB 1100: REM  SUBRRUTINA *CREMAT* (CREAR MATRICES)
1020 GOSUB 1300: REM  SUBRUTINA *LEARCH* (LEER ARCHIVOS)
1030 GOSUB 1380: REM  SUBRUTINA *ESBAND* (ESTARLECER BANDERAS)
1040 REM
1050 REM
1060 REM
1070 REM
1080 REM
1090 RETURN
1100 REM SUBRUTINA *CREMAT* (CREAR MATRICES)
1110 DIM NOMCAM$(50)
1120 DIM MODCAM$(50)
1125 DIM CLLCAM$(50)
1130 DIM CIUCAM$(50)
1140 DIM PROCAM$(50)
1150 DIM TELCAM$(50)
1160 DIM INDCAM$(50)
1170 REM
1180 REM
1190 REM
1200 REM
1210 LET TAMA = 0
1220 LET RMOD = 0
1230 LET SVED = 0
1240 LET CURS = 0
1250 REM
1260 REM
1270 REM
1280 REM
1290 RETURN
1300 REM SUBRUTINA *LEARCH* -- VEASE RECUADRO "COMPLEMENTOS"
1310 ON ERROR GOTO 1370
1320 OPEN "I",#1,"DAT.AGCO"
1330 FOR L = 1 TO 50
1340 INPUT #1 NOMCAM$(L),CLLCAM$(L),CIUCAM$(L),PROCAM$(L), TELCAM$(L)
1350 NEXT L
1360 CLOSE #1
1370 RETURN
1380 REM RUTINA *ESBAND* FICTICIA
1390 RETURN
3000 REM SUBRUTINA *PRESN*
3010 PRINT
3020 PRINT
3030 PRINT
3040 PRINT
3050 PRINT TAB(11);"*BIEN VENIDO A LA *"
3060 PRINT TAB(9);"*AGENDA COMPUTERIZADA*"
3070 PRINT TAB(12);"*DE MI COMPUTER*"
3080 PRINT
3090 PRINT TAB(0);"(PULSE BARRA ESPACIADORA PARA CONTINUAR)"
3100 FOR L = 1 TO 1
3110 IF INKEY$ <> " " THEN L = 0
3120 NEXT L
3130 PRINT CHR$(12)
3140 RETURN
3500 REM SUBRUTINA *ELECCN*
3510 REM
3520 REM "IMMENU"
3530 PRINT CHR$(12)
3540 PRINT "SELECCIONE UNA DE LAS SIGUIENTES OPCIONES"
3550 PRINT
3560 PRINT
3570 PRINT
3580 PRINT "1. HALLAR REGISTRO (DE NOMBRE)"
3590 PRINT "2. HALLAR NOMBRES (DE NOMBRE INCOMPLETO)"
3600 PRINT "3. HALLAR REGISTRO (DE CIUDAD)"
3610 PRINT "4. HALLAR REGISTRO (DE INICIAL)"
3620 PRINT "5. LISTAR TODOS LOS REGISTROS"
3630 PRINT "6. AGREGAR REGISTRO NUEVO"
3640 PRINT "7. MODIFICAR REGISTRO"
3650 PRINT "8. BORRAR REGISTRO"
3660 PRINT "9. SALIR Y GUARDAR"
3670 PRINT
3680 PRINT
3690 REM  "ASOPCN"
3700 REM
3710 LET L = 0
3720 LET I = 0
3730 FOR L = 1 TO 1
3740 PRINT "DE ENTRADA A OPCION (1 - 9)"
3750 FOR I = 1 TO 1
3760 LET A$ = INKEY$
3770 IF A$ = "" THEN I = 0
3780 NEXT I
3790 LET OPCN = VAL(A$)
3800 IF OPCN < 1 THEN L = 0
3810 IF OPCN > 9 THEN L = 0
3820 NEXT L
3830 RETURN
4000 REM SUBRUTINA *EJECUT* -- VEASE RECUADRO "COMPLEMENTOS"
4010 ON OPCN GOSUB 310, 330, 350, 370, 390, 410, 430, 450, 470
4020 RETURN
9000 REM SUBRUTINA *INCREG*
9010 PRINT CHR$ (12)
9020 INPUT "DE ENTRADA AL NOMBRE";NOMCAM$(TAMA)
9030 INPUT "DE ENTRADA A LA CALLE";CLLCAM$(TAMA)
9040 INPUT "DE ENTRADA A LA CIUDAD" CIUCAM$(TAMA)
9050 INPUT "DE ENTRADA A LA PROVINCIA" PROCAM$(TAMA)
9060 INPUT "DE ENTRADA AL NUMERO DE TELEFONO";TELCAM$(TAMA)
9070 LET RMOD = 1
9080 LET INDCAM$(TAMA) = STR$(TAMA)
9090 GOSUB *MODNOM*
9100 RETURN
```

Tony Lodge

N O P Q R S T U V X Z

Un buen sonido

La generación de sonido en el BBC Modelo B

Unas excepcionales configuraciones para sonido y amplias órdenes en BASIC para controlarlas hacen que el BBC Micro sea uno de los mejores ordenadores para aquellos usuarios que se interesen, sobre todo, por el sonido. Se proporcionan como estándar tres osciladores de onda cuadrada independientes, ocho tipos de ruido y cuatro envolventes independientes de ADSR (arranque, demora, sostenido, liberación) y tono. Ello significa que se pueden construir frases musicales de hasta tres voces en armonía; además, órdenes especiales permiten que las notas seleccionadas para formar un acorde suenen exactamente al mismo tiempo.

Creación del sonido

En su forma más simple, el sonido se puede crear mediante la orden SOUND:

```
SOUND C,V,P,D
```

En este caso:

C = Número del canal u oscilador (0-3)
V = Volumen
P = Tono de la nota
D = Duración o longitud de la nota

Cualquiera de los tres osciladores (1, 2 y 3) puede tocar una nota; 0 corresponde a ruido. El volumen adopta un valor entre 0 (apagado) y −15 (fuerte). El tono se define en intervalos de un cuarto de nota (medio semitono) entre 0 (*la #* a 116,5 Hz) y 255 (*re* a 4698,64 Hz), ofreciendo una escala de cinco octavas y media de extensión. El do_3 (o central) se establece mediante el valor 28. Si se ha especificado el canal de ruido, existen ocho tipos disponibles, que determinan los siguientes números de tono:

Número	Tipo de ruido
0	Trémolo de tono agudo
1	Trémolo de tono medio
2	Trémolo de tono grave
3	Trémolo. El tono varía con el tono del canal 1
4	Tono agudo
5	Tono medio
6	Tono grave
7	El tono varía con el tono del canal 1

Por último, la duración de la nota se controla en intervalos de veinteavos de segundo entre 1 y 255, es decir, una nota puede durar hasta 12,75 segundos. Una orden para tocar el la_3 en el canal 1 a un volumen medio de −7 durante medio segundo se construye de la siguiente forma:

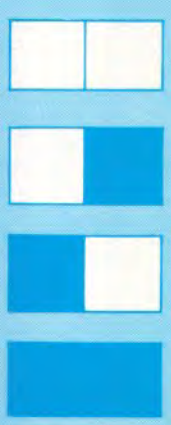
```
SOUND 1,−7,46,10
```

Si el ordenador recibiera una segunda orden SOUND antes de haber completado la primera, la nota se colocaría al final del canal.

Además de las funciones sencillas que hemos re-

Construcción de luz

Las facilidades para gráficos del Commodore 64



El ordenador Commodore 64 proporciona gran cantidad de posibilidades al programador de gráficos. No obstante, adolece de un inconveniente, que comparte con el Vic-20: su muy limitado juego de órdenes estándar en BASIC. Mediante las instrucciones POKE y PEEK el programador consigue acceder a todas las configuraciones de la máquina, pero algunos de los procedimientos necesarios pueden resultar bastante confusos. Asimismo, al igual que en el Vic-20, existen varios caminos para abrirse paso a través de este laberinto. Y éstos adoptan la forma de varios paquetes de programas producidos por fabricantes independientes que simplifican muchísimo la creación de sprites o de caracteres definidos por el usuario. Además de este software, Commodore fabrica su propio cartucho enchufable, denominado *Simon's* BASIC.

El Commodore 64 ofrece sus juegos de caracteres estándar en mayúscula y minúscula en modalidades de visualización normal o invertida. También se encuentran a la venta los caracteres especiales para gráficos que originariamente se desarrollaron para el PET y que también se utilizan con el Vic-20. La visualización en pantalla consta de 40 columnas y 25 filas y los caracteres aparecen en ella imprimiéndolos (PRINT) en cualquier color elegido o colocando (POKE) los códigos correspondientes en la memoria de pantalla y de color detalladas en el manual del usuario. El juego de caracteres especiales para gráficos del Commodore constituye un instrumento muy adaptable, con el cual se pueden unir atractivas visualizaciones en baja resolución. En esta tarea es posible emplear más de 60 caracteres especiales, que se puede visualizar en modalidad normal o invertida, dando al usuario más de 120 posibilidades de elección al diseñar una figura.

Si no se consigue hallar el carácter exacto, entonces se pueden definir caracteres nuevos para utilizar en la visualización. Esto, lamentablemente, no es muy sencillo de hacer. Para definir un carácter el programador debe, primero, copiar, de la memoria ROM a la RAM, todos los caracteres normales que se hayan de emplear, antes de colocar (POKE) los números necesarios en la serie de posiciones que habrán de retener el carácter nuevo.

señado, la capacidad de la orden SOUND se puede ampliar modificando su formato:

SOUND & HSFC,V,P,D

En este caso el signo & instruye al ordenador para que trate HSFC como un número hexadecimal de cuatro dígitos (véase p. 179). Sin embargo, para comprender cómo se debe utilizar la orden modificada únicamente es necesario analizar la función de cada dígito. Sólo tres de éstos son significativos, ya que C es simplemente el número de canal normal descrito en SOUND.

H puede estar sin especificar (0) o bien establecida en 1. H = 1 le permite a la nota anterior del mismo canal continuar hasta el final. Por otra parte, si se construyera una nota con una fase de liberación larga (apagándose lentamente), el ordenador supondría erróneamente que la nota ha finalizado mientras todavía está sonando y comenzaría de manera abrupta la nota siguiente en mitad de ella. H establecida en 1 obliga al ordenador a esperar que se complete. Cuando se utiliza el parámetro H, las restantes partes de la orden SOUND & se ignoran, a excepción del número de canal.

S le permite al usuario especificar un número de notas a ejecutar al mismo tiempo en distintos canales, creando un acorde. S = 0 deja la orden en su estado normal. S se establece en 1 si se requiere que al mismo tiempo se toque una nota en alguno de los otros canales. Si se establece en 2, se tocarán notas en los otros dos canales. En efecto, cuando el ordenador se encuentra en un valor S retiene la nota correspondiente hasta que puede dar cuenta de la otra nota o las otras dos notas relacionadas del acorde que están indicadas con el mismo número S en los canales restantes. Entonces ejecuta juntas todas las notas especificadas.

F se establece en 0 o en 1. Cero no tiene ningún efecto, pero 1 hace que el ordenador descarte todas las notas que esperan ser ejecutadas al final del canal, interrumpe la ejecución de la nota en curso y toca inmediatamente la nota contenida en su propia orden.

La mejor forma de ilustrar SOUND & es mediante un ejemplo. Este programa toca la primera línea de *Happy birthday to you* (Cumpleaños feliz) en la tonalidad de *fa#* (notas inmediatamente superiores al do_3):

fa# fa# sol# fa# si la#:

También incluye un acorde mayor de tres notas para el *la#* final compuesto por *re, fa* y *la#*

```
10 SOUND 1,-7,40,20: REM *1ER FA#*
20 SOUND 1,-7,40,10: REM *2DO FA# BREVE*
30 FOR I = 1 TO 3: READ N
40 SOUND 1,-7,N,20: NEXT I: REM *SOL#
   FA# SI*
50 SOUND &201,-7,32,30: REM *RE*
60 SOUND &202,-7,38,30: REM *FA*
70 SOUND &203,-7,48,30: REM *LA#*
80 DATA 44,40,50
90 END
```

Las capacidades de sonido que proporciona la orden ENVELOPE las analizaremos en un próximo capítulo.

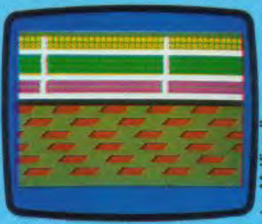
Cada celda de caracteres se compone de líneas de puntos pixel de ocho por ocho, que se representan en forma binaria como grupos de unos y ceros. Normalmente 1 se interpreta como un pixel establecido para el color del primer plano o del carácter, y 0 se interpreta como un pixel establecido para el color del fondo o de la pantalla. El Commodore 64 proporciona la posibilidad de representar hasta cuatro colores dentro de cada celda de caracteres. Esto se conoce como modalidad multicolor. Cuando el ordenador se coloca en esta modalidad, utiliza dos bits para representar el color de cada pixel.

Para cualquier par de bits existen cuatro combinaciones posibles, que se emplean para representar cuatro colores diferentes en el interior de la celda de caracteres. Cada una de las celdas de caracteres de la pantalla se puede establecer para que se la interprete de forma normal o bien como un carácter multicolor, pero en este último caso la elección de colores disponibles se reduce de 16 a 8. Existen otros inconvenientes: los dos bits multicolores han de ser los mismos para todos los caracteres de la pantalla; además, en la modalidad multicolor la resolución horizontal se reduce a la mitad.

En el BASIC de Commodore no existen órdenes para gráficos de alta resolución. No obstante, se pueden crear visualizaciones de igual resolución utilizando la técnica conocida como *confección de mapas de bits*. La visualización en pantalla del Commodore 64 consta de 64 000 pixels. La confección de mapas de bits se efectúa posibilitando que el usuario encienda o apague cada uno de los pixels de forma individual. Se trata de un procedimiento bastante complicado como para que lo pueda realizar el usuario corriente; por otra parte, al utilizar el BASIC de esta manera, la imagen en alta resolución se va creando con notable lentitud. En la práctica, existen dos alternativas para quienes deseen crear gráficos de alta resolución: la primera es comprar el cartucho *Simon's* BASIC de Commodore; la segunda, aprender a programar en código de lenguaje máquina. La modalidad estándar de alta resolución del Commodore 64 divide la pantalla en 200 líneas de 320 pixels. También se puede adquirir modalidad multicolor de alta resolución.

A continuación transcribimos un breve programa que utiliza el juego de caracteres Commodore para crear una imagen de un supermercado. En un próximo capítulo analizaremos los sprites y el *Simon's* BASIC.

Primer paso
Esta escena estática de supermercado se creó utilizando gráficos de baja resolución. En un próximo capítulo de esta sección agregaremos, utilizando gráficos sprite, un comprador con un carrito móvil



Ian McKinnell

```
3000 REM ** SUPERMARKET **
3010 PRINT"■"
3020 POKE53280,6:POKE53281,12
3030 PRINT"■●●●●■ ■●●●●●●●●●●●●●●●●●●●●●●■ ■●●●●●●●●●●●●●"
3040 PRINT"■ooooo■ ■oooooooooooooooooooooo■ ■oooooooooooooo"
3050 PRINT"■                                        ■"
3060 PRINT"■◆◆◆◆■ ■◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆■ ■◆◆◆◆◆◆◆◆◆◆◆◆◆"
3070 PRINT"■◆◆◆◆■ ■◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆■ ■◆◆◆◆◆◆◆◆◆◆◆◆◆"
3080 PRINT"■◆◆◆◆■ ■◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆■ ■◆◆◆◆◆◆◆◆◆◆◆◆◆"
3090 PRINT"■                                        ■"
3100 PRINT"■♥♥♥♥■ ■♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥■ ■♥♥♥♥♥♥♥♥♥♥♥♥♥"
3110 PRINT"■♥♥♥♥■ ■♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥■ ■♥♥♥♥♥♥♥♥♥♥♥♥♥"
3120 PRINT"■                                        ■"
3130 PRINT"■                                        ■"
3140 PRINT"■  ■   ■   ■   ■   ■   ■   ■   ■   ■"
3150 PRINT
3160 PRINT" ■   ■   ■   ■   ■   ■   ■   ■   ■   ■"
3170 PRINT
3180 PRINT"■   ■   ■   ■   ■   ■   ■   ■   ■   ■ "
3190 PRINT
3200 PRINT"■ ■   ■   ■   ■   ■   ■   ■   ■   ■   ■
3210 PRINT
3220 PRINT"■   ■   ■   ■   ■   ■   ■   ■   ■   ■"
3230 PRINT
3240 PRINT" ■   ■   ■   ■   ■   ■   ■   ■   ■   ■"
3250 FORI=1063TO2023STEP40
3260 POKEI,160:POKE54272+I,6:NEXT
3270 GOTO3270
```

Torres Quevedo

Además de construir dirigibles y transbordadores, este científico español contribuyó decisivamente al desarrollo de la informática

Leonardo Torres Quevedo, el científico que aplicó por primera vez la aritmética de coma flotante a los ordenadores, nació en Santa Cruz (Cantabria) el 28 de diciembre de 1852. Estudió en el Instituto de Bilbao y en la Facultad de Ingeniería de Madrid.

Su genio inventor alcanzó su punto culminante en los últimos años de su vida. A él se le debe la construcción del transbordador funicular aéreo junto a las cataratas del Niágara, que en la actualidad continúa en uso; además diseñó un dirigible dotado de una armadura funicular que reunía las propiedades de los dirigibles rígidos y flexibles. Pero, básicamente, Torres era un hijo de su época y su principal interés se centraba en los dispositivos electromecánicos. En 1906, en Bilbao, exhibió ante el rey de España, Alfonso XIII, el prototipo de un barco dirigido a distancia por medio de las ondas hertzianas, y en 1911 inventó el primer jugador de ajedrez autómata. Para mover las piezas la máquina utilizaba unos electroimanes situados debajo de la mesa; estaba programada para ganarle una partida sencilla a un oponente humano.

El interés de Torres por los autómatas nació a raíz de su experiencia en las líneas de montaje de las plantas industriales de la Europa de comienzos del siglo xx; y a lo largo de toda su vida se esforzó por separar los tipos de problema que requerían pensamiento humano para su resolución de aquellos que se podían resolver automáticamente.

Científico polifacético
Al igual que muchos de sus contemporáneos, Torres Quevedo fue un científico que dedicó sus esfuerzos a la investigación en diferentes campos. Sus intereses abarcaron desde el diseño y construcción de dispositivos mecánicos como el dirigible que vemos en la fotografía hasta la creación de máquinas de calcular electromecánicas, ya dentro de los dominios de la matemática pura

Cortesía de la Royal Aeronautical Society

En 1914 publicó un estudio en el que demostraba la viabilidad de la construcción del ingenio analítico de Babbage (véase p. 220) empleando técnicas electromecánicas, y fue en este documento donde sugirió por primera vez la utilización de la aritmética de coma flotante para todo futuro ordenador. En 1920 construyó una calculadora electromecánica que utilizaba una máquina de escribir adaptada para dar entrada a los números e imprimía los resultados automáticamente. La máquina de escribir estaba conectada a la calculadora mediante cables telefónicos, y Torres previó la posibilidad de acoplar muchos terminales a una calculadora central (o unidad de proceso).

Torres Quevedo fue condecorado por la Academia Francesa de Ciencias, y con posterioridad sería nombrado presidente de honor vitalicio de la Academia de Ciencias de Madrid. Falleció en Madrid el 18 de diciembre de 1936.

La aritmética de coma flotante

Una caja registradora visualiza los totales en pesetas y céntimos (12,50, por ejemplo) y en una máquina de este tipo sólo son necesarios dos espacios después de la coma decimal. Pero en un ordenador, con frecuencia se requiere mayor exactitud y se permite que el número de lugares decimales varíe o "flote" de acuerdo a las exigencias del problema. Esto se conoce como "aritmética de coma flotante".

Cualquier número se puede escribir de diversas maneras. Por ejemplo, 0,8752 metros se puede expresar como 875,2 milímetros, o $0{,}8752 \times 1000$ milímetros, o simplemente $0{,}8752 \times 10^3$. Este último procedimiento se presta para cumplir las funciones de codificación en un ordenador. Si éste sólo tiene asignados seis espacios digitales para representar a cada número (y, en aras de la claridad, se utiliza el sistema decimal en lugar del binario), entonces el número anterior se puede almacenar como 875203: donde los dos últimos dígitos de la derecha se denominan *índice* y representan la potencia de 10 (en este caso, 3) y los cuatro primeros dígitos reciben el nombre de *mantisa*. Para dar otro ejemplo para este ordenador: el número 418302 representa $0{,}4183 \times 10^2$, o 41,83.

La mantisa y el índice generalmente se "normalizan" para eliminar de la mantisa los primeros ceros. Por ejemplo, el número 41,83 se podría escribir como 004104, pero se normalizaría en 418302 (incluyendo, por lo tanto, más dígitos significativos en la mantisa).

La forma índice-mantisa de la aritmética de coma flotante ofrece la ventaja de que se puede representar una amplia escala de números. En el caso del ordenador que mencionábamos anteriormente, que dispone de sólo dos dígitos para el índice, puede tratar con un número tan elevado como $0{,}9999 \times 10^{99}$, o con un número tan pequeño que posea 98 ceros después de la coma decimal y antes del primer dígito distinto a cero.

No obstante, la exactitud de este sistema permanece limitada a los dígitos que se destinen a la mantisa. Por consiguiente, algunos números sólo se pueden representar aproximadamente y para evitar que se produzcan errores las técnicas de la programación aritmética se han de asumir con gran cuidado. Ésta es la razón por la cual, en algunos ordenadores, (1/3)*3 dará como resultado 0,9999999, en vez de la respuesta verdadera, que es 1.

Juegos de inteligencia

Los programas de ajedrez son difíciles de escribir, pero hasta un principiante puede hacer un programa sencillo e "inteligente"

Cortesía de Milton Bradley Ltd

Ian McKinnell

La mano invisible
Las máquinas exclusivas para jugar al ajedrez contienen los mismos componentes que los ordenadores personales: una CPU, RAM y el programa en ROM y sólo difieren en el método de entrada y salida. El Phantom, que vemos en la fotografía, utiliza un servomecanismo e imanes que capacitan al ordenador para mover las piezas de ajedrez automáticamente. Cuando, por ejemplo, un caballo salta sobre otra pieza, se emplea un sofisticado algoritmo que quita cualquier obstáculo y que lo vuelve a restituir a su sitio después de efectuada la jugada

Muchas personas, cuando empiezan a escribir sus propios programas para ordenador, sueñan con el día en que sabrán lo suficiente como para poder hacer un programa que juegue al ajedrez. Y esto no se debe, por supuesto, a que no se haya creado software para este juego. Existe gran abundancia de estos programas, tanto en forma de paquetes para ordenadores personales como en forma de máquinas dedicadas exclusivamente a jugarlo. Pero escribir programas de ajedrez se puede llegar a convertir en una obsesión, aun para aquellos programadores que no estén particularmente interesados en él como juego. En este sentido, una posible causa de este interés podría ser el hecho de que nosotros consideramos este juego como un pasatiempo de alto nivel intelectual y, en consecuencia, el ordenador que pueda jugar al ajedrez habrá dado un paso más hacia la creación de una máquina inteligente. Sin embargo, sería muy difícil explicarle a usted, partiendo de cero, cómo escribir un programa completo para jugar al ajedrez. Lo que sí podemos hacer es explicarle algunos de los principios en virtud de los cuales se construyen juegos informatizados "inteligentes", y a un nivel que le permitirá escribir un programa bastante sofisticado en BASIC.

Es importante recordar, no obstante, que los "juegos" a los que estamos aludiendo no son recreativos ni de aventuras ni simulaciones, todos los cuales requieren técnicas de programación diferentes y distinta destreza imaginativa. Comenzaremos nuestro análisis de los juegos de inteligencia con algo que quizá considere un ejemplo trivial, pero que ilustra muchos de los principios que se aplican en la escritura de esta clase de juegos.

La mayoría de los niños conoce el juego tijeras-papel-piedra. Las reglas son sencillas: ambos jugadores piensan en uno de estos tres objetos y después levantan al mismo tiempo una mano en un gesto que representa el objeto escogido. El ganador se decide de acuerdo a estas tres reglas: las tijeras ganan al papel (cortándolo), el papel gana a la piedra (envolviéndola) y la piedra gana a las tijeras (mellándolas).

Para cualquiera que haya seguido nuestro curso de programación BASIC, será muy sencillo escribir un programa que cumpla el papel del ordenador en el juego y lleve el marcador. La función RND se utiliza para seleccionar un elemento de una matriz en serie de tres elementos que contenga 'TIJERAS', 'PAPEL' y 'PIEDRA'. Luego se imprime (PRINT) el elemento elegido al pulsar la barra espaciadora. El

jugador digita su propia elección (el programa confía en su honestidad) y el programa calcula entonces quién ha ganado, visualizando el resultado y acumulando el marcador para él mismo y para su oponente. Si la función RND es verdaderamente al azar, los marcadores deberán ser parejos al cabo de cierta cantidad de vueltas, independientemente de la estrategia que adopte el jugador. Ahora es necesario que determinemos cómo podemos mejorar la estrategia del ordenador para asegurarnos de que él gane al cabo de una serie de vueltas.

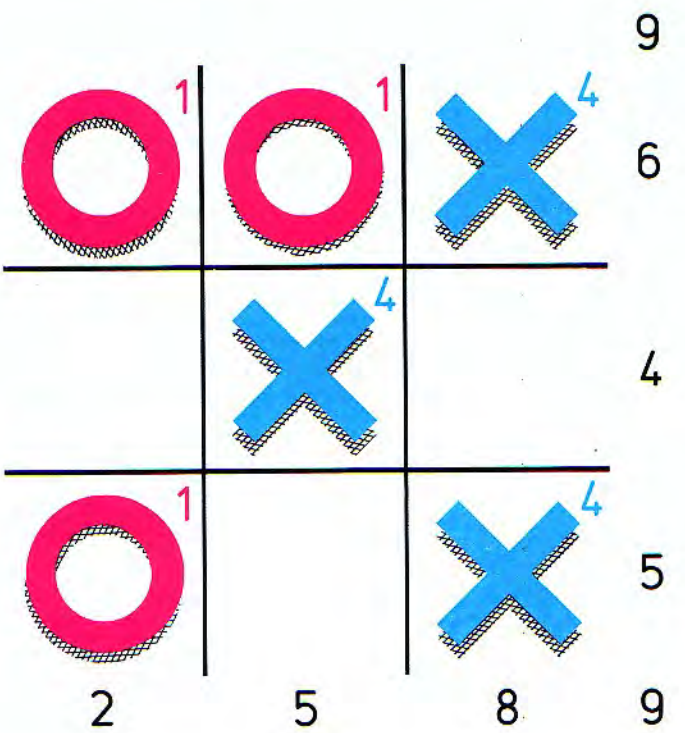
Cuando examinamos las funciones al azar (véase p. 209) aprendimos que generar una secuencia de números verdaderamente aleatorios es una tarea imposible tanto para los humanos como para los ordenadores, aunque estos últimos enfoquen la cuestión mucho mejor. Al cabo de muchas vueltas de nuestro juego, el jugador humano invariablemente favorecerá a uno de los objetos en detrimento de los otros dos. Usted puede escribir una subrutina en su programa que lleve el registro de las elecciones del jugador, utilizando una matriz con tres elementos denominados, pongamos por caso, OPCION(1), OPCION (2) y OPCION(3). Cada vez que el jugador haga una elección, se agrega uno al total del elemento de la matriz correspondiente. El ordenador puede entonces determinar cuál es el objeto que su oponente le presenta más veces y elegir el objeto que venza a la opción favorita del jugador.

Este procedimiento plantea tres problemas. En primer lugar, si el ordenador optara coherentemente por el mismo objeto, el jugador no tardaría mucho en sacar partido de ello. Por consiguiente, por lo general se debe hacer que el ordenador elija entre los tres objetos mediante la función RND, debiéndose agregar una rutina para asegurar que el objeto escogido con mayor frecuencia sea el que derrote a la elección favorita del jugador.

El segundo problema es que el jugador tenderá a cambiar su objeto preferido en el transcurso del juego. De manera que, en vez de llevar el registro de las elecciones del oponente desde el inicio del juego, sería mejor que el programa simplemente registrara, supongamos, las últimas 20 opciones. Ello requeriría una matriz OPCION de 20 por 3 elementos y una subrutina más sofisticada para sumar las tres columnas y predecir cuál sería la mejor elección del ordenador para la siguiente vuelta.

No obstante, el inconveniente más serio de este algoritmo es que el jugador deduzca la estrategia del ordenador. Entonces le resultaría relativamente sencillo jugar de forma que el ordenador perdiera en más de la mitad de las vueltas. El jugador podría, por ejemplo, jugar coherentemente el mismo objeto y después cambiar a otro de forma inesperada, y así una y otra vez. Lo que necesitamos es un algoritmo diferente que evite estos problemas. Sin embargo, valdría la pena desarrollar programas que utilizaran tanto los métodos totalmente al azar como los métodos al azar modificados, y observar los marcadores cuando los emplearan jugadores poco desconfiados.

Dado que los seres humanos son incapaces de tomar una decisión totalmente irracional o al azar, toda elección ha de ser una función de las elecciones anteriores. Esa función puede ser sumamente compleja, sin que el jugador, casi con toda certeza, sea consciente de ello; pero si el ordenador puede calcular una buena aproximación a dicha función, debería ser capaz de ganar la mayor parte de las veces. Dado que cada jugador tendrá su propia fórmula subconsciente, y que probablemente cambiará esa fórmula en el transcurso de un juego largo, se debe hacer que el programa interprete la fórmula mientras se está jugando. Los programas que pueden aprender así se llaman *heurísticos*.



La posición ganadora

La "evaluación de posiciones" es fundamental para el tablero de cualquier programa de juegos, incluso aunque el juego sea tan sencillo como el tres en raya. En este caso, el tablero está representado por una matriz de tres por tres, los círculos del jugador por el valor uno y las cruces del ordenador por un cuatro. Utilizando estos valores se puede evaluar cualquier posición sumando los totales de todas las filas, columnas y diagonales. Un total de 12 en cualquiera de estas líneas indica que es el ordenador el que ha ganado; tres significa que es el jugador quien ha ganado; un total de ocho, que se han jugado dos cruces y que el ordenador puede ganar; y así sucesivamente. Se emplean los valores uno y cuatro porque éstos aseguran que cada combinación de círculos y cruces da un total único

Liz Dixon

Un programa heurístico permite que el ordenador detecte los cambios en la estrategia del oponente y modifique, por consiguiente, su algoritmo. Dicho programa tendría que llevar un registro de, pongamos por caso, las últimas 50 elecciones de ambos contrincantes, en una matriz. El programa explora constantemente este registro y aplica una técnica estadística conocida como "correlación".

Ésta induce al ordenador a establecer cientos de comparaciones entre la elección del jugador y su elección anterior, o la que precedió a ésta, o la elección que hizo cinco vueltas antes. El ordenador efectúa la misma operación con sus propias elecciones. Consideremos, por ejemplo, la correlación entre la jugada del jugador y su jugada anterior. Denominaremos elemento 1 a las tijeras, elemento 2 al papel y elemento 3 a la piedra. Primero debemos establecer una matriz de tres por tres, a la que llamaremos CORR1, porque representa nuestra primera prueba de correlación. Ahora debemos examinar la historia de nuestro juego, analizando las elecciones del jugador durante las últimas 50 jugadas. Cada vez que a tijeras (1) le siga piedra (3), agregaremos uno al elemento CORR1(1,3); cuando a piedra (3) le siga papel (2), agregaremos uno al elemento CORR1(3,2), y así sucesivamente.

Si el jugador estuviera haciendo sus elecciones auténticamente al azar, entonces en cada elemento de CORR1 debería haber valores aproximadamente iguales; pero es muy poco probable que se dé este caso. Por tanto, si la última vez el jugador eligió papel, entonces el elemento en la fila 2 (papel) de CORR1 con el mayor valor nos proporcionará la mejor predicción acerca de cuál será su próxima elección. Cuanto mayor sea la diferencia entre los elementos de cualquier fila, mayor será la correlación y más fiable será la predicción. Sin embargo, es posible que haya muy poca correlación entre la elección del jugador y su elección anterior, en cuyo caso también deberemos efectuar los cálculos de correlación con la antepenúltima elección, o entre la del jugador y la anterior del ordenador.

Cuando todas las rutinas de correlación predicen resultados distintos para la siguiente jugada del jugador, surge un problema. El programa debe decidir cuál es la sugerencia más fiable. En este juego sencillo, todo lo que hay que hacer es comprobar cuál de las pruebas tiene la correlación más pronunciada. Por ejemplo, la matriz CORR1 podría predecir las siguientes probabilidades: tijeras 51 %, papel 29 %, piedra 20 %; mientras que CORR2 (que, supongamos, compara la elección del jugador con la última elección del ordenador) podría dar: tijeras 24 %, papel 60 %, piedra 16 %. Evidentemente, CORR2 tiene la mejor correlación, de manera que se debería escoger su predicción. De hecho, un programa de juegos inteligente constará de una cierta cantidad de subrutinas, cada una de las cuales trabajará con distintas estrategias y aconsejará a la rutina principal la mejor jugada. La rutina de juego puede considerar a estas subrutinas como si se tratara de un "comité", y actuar en base a una decisión tomada por mayoría. Pero a medida que avanza el juego, puede ponderarse cada rutina según si su consejo ha sido bueno o no.

En el caso de que existiera alguna correlación entre las jugadas o elecciones del jugador y las jugadas anteriores del ordenador, se podría programar alguna clase de factor "de farol" que deliberadamente condujera al jugador a conclusiones erróneas. Donde mejor funciona esto es en los juegos con apuestas en los que éstas aumentan a medida que progresa el juego y es mejor perder las primeras jugadas para ganar las últimas.

```
  5 CLS
 10 DIM C1(3,3),C2(3,3)C3(3,3)
 20 CR=0
 30 FOR I=1 TO 3
 40 IF C1(PL,I)>CR THEN BG=I:CR=1C1(PL,I)
 50 IF C2(PP,I)>CR THEN BG=I:CR=C2(PP,I)
 60 IF C3(P3,I)>CR THEN BG=I:CR=C3(P3,I)
 70 NEXT I
 80 CT=BG-1
 90 IF BG=1 THEN CT=3
100 GET PT: IF PT=0 THEN 100
110 REM LA LINEA 100 ESPERA A QUE SE
120 REM PULSE UN DIGITO
130 IF CT=PT-1 THEN CS=CS+1
140 IF CT=PT-2 THEN PS=PS+1
150 IF CT=PT+1 THEN PS=PS+1
160 IF CT=PT+2 THEN CS=CS+1
170 CLS
180 PRINT "SU ELECCION: ";PT
190 PRINT "MI ELECCION: ";CT
200 PRINT "SU MARCADOR ES ";PS
210 PRINT "MI MARCADOR ES ";CS
220 C1(PL,PT)=C1(PL,PT)+1
230 C2(PP,PT)=C2(PP,PT)+1
240 C3(P3,PT)=C3(P3,PT)+1
250 P3=PP
260 PP=PL
270 PL=PT
280 GOTO 20
```

Un alumno lento
Este programa, basado en el juego tijeras, papel, piedra, ilustra cómo un programa puede "aprender" a medida que avanza el juego. El ordenador selecciona entre los números 1, 2 y 3, compara su elección con la que usted ha digitado y ajusta el marcador. Se ha utilizado la sentencia GET para que pueda pulsar sencillamente las tres teclas de números en rápida sucesión. Si intenta efectuar su secuencia al azar, descubrirá que tras unas doscientas pulsaciones el marcador del ordenador tomará la delantera. Existe la posibilidad de burlar este programa y, en consecuencia, ganar, pero se pueden agregar rutinas más sofisticadas para impedírselo

En la Universidad del Estado de Nueva York, en Buffalo (según un informe publicado en la revista *Scientific American* en julio de 1978), se hizo jugar entre sí, durante varios miles de partidas a un conjunto de programas de póker. El ganador absoluto fue un programa denominado AEO (*Adaptive Evaluator of Opponents*: evaluador adaptable de oponentes), que elaboraba un juicio inicial acerca del valor de las manos de sus oponentes e iba modificando esta apreciación a medida que avanzaba la partida. El programa SBI (*Sells and Buys Images:* compraventa de imágenes) funcionó sorprendentemente mal: su técnica consistía en establecer faroles con la intención de "venderle" a su oponente una imagen falsa o de "comprar" el estilo de juego de otros. El BP (*Bayesian Player:* jugar a mantener a raya) intentaba hacer deducciones inductivas y mejorar su juego comparando las consecuencias pronosticadas de sus acciones con las consecuencias reales. Por último, el programa AAL (*Adaptative Aspiration Level*: nivel de aspiración adaptable) intentó imitar una característica del juego humano: adaptar el nivel de aspiración (o sea, el grado de riesgo que se está preparado a asumir) a sus mejores resultados anteriores y a su condición presente.

No existen dos programas de ajedrez u otras rutinas artificialmente inteligentes que trabajen de la misma manera. Pero experimentando con las técnicas que hemos esbozado aquí en juegos cada vez más complicados, quizá llegue usted a ingresar en el club de escritores de programas de ajedrez.

Mapas de memoria

Los lenguajes de alto nivel como el BASIC administran la memoria automáticamente; de lo contrario, necesitaríamos un trazado de la memoria para hallar el camino a través del ordenador

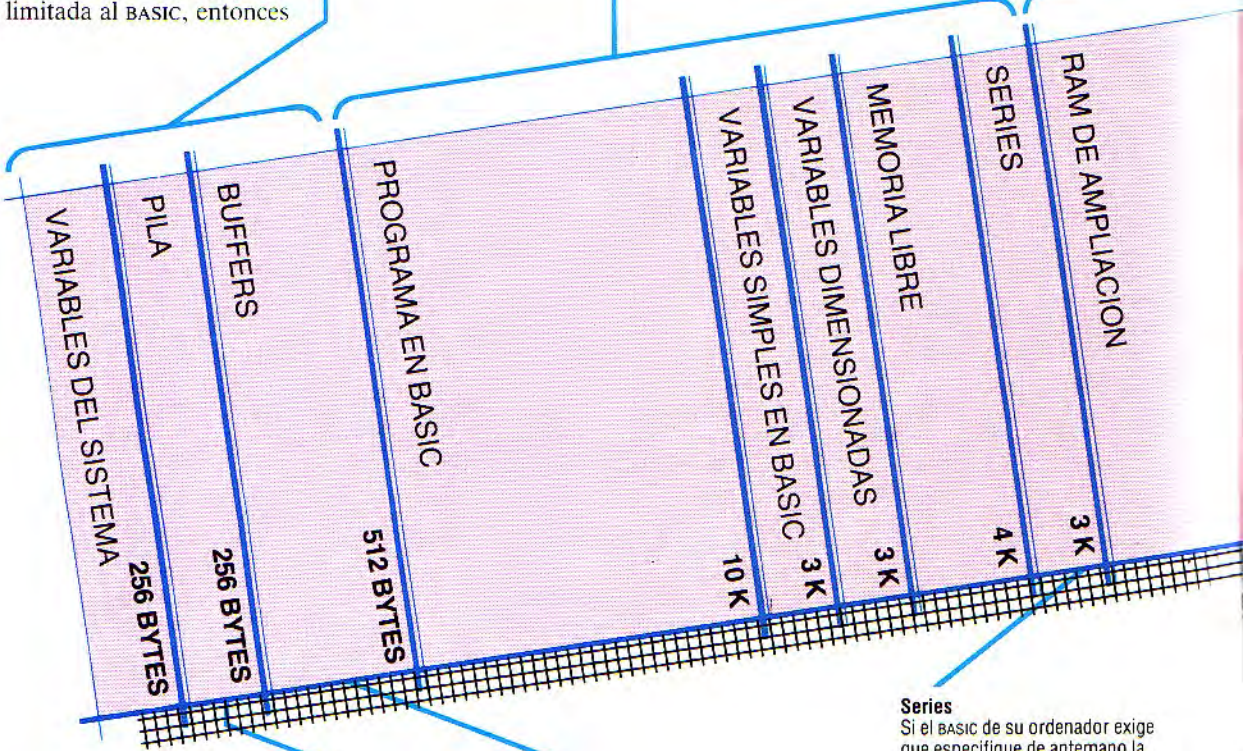
La CPU, en el corazón del ordenador, posee una escala de direccionamiento que determina el número máximo de posiciones de memoria a las que puede acceder, y en la mayoría de los ordenadores personales éste es de 64 Kbytes. Ese espacio de memoria debe contener toda la RAM y ROM que viene con la máquina, toda ampliación de ambas memorias susceptible de realizarse y todos los chips y puertas para interface, que la CPU también considera como posiciones de memoria. Uno de los aspectos más importantes del diseño de un ordenador es el "mapa de memoria": la lista o el diagrama que especifica qué partes del espacio de memoria se destinan a cada una de las funciones de la máquina. Si su programación está limitada al BASIC, entonces no necesita conocer con detalle el mapa de memoria. Pero si se iniciara en el código de lenguaje máquina o si acariciara la idea de construir sus propios accesorios de hardware, entonces conocer el mapa de memoria sería de vital importancia.

En estas páginas le mostramos el contenido de un mapa de memoria típico. Nuestro ejemplo se aproxima más a un sistema basado en un 6502 que a uno basado en un Z80, pero la mayoría de los elementos son comunes a ambos. Algunos fabricantes imprimen un mapa completo en el manual de instrucciones, mientras que otros mantienen reserva absoluta respecto al diseño. Sin embargo, por lo general es posible hallar usuarios que hayan logrado reconstruirlo a partir de la experiencia.

Overhead del sistema
Un ordenador con 4 Kbytes de RAM puede en realidad tener sólo 3 Kbytes disponibles para los programas del usuario. La diferencia estriba en el "overhead" del sistema, una sección de la RAM que queda reservada por el sistema operativo cada vez que se enciende la máquina. Parte de ella se utiliza para las variables del sistema y para los indicadores que señalan en qué parte de la memoria se retienen habitualmente los diferentes datos

RAM para el usuario
Las dimensiones de ésta determinan la sofisticación de los programas que usted puede ejecutar. Quizá sea una de las consideraciones más importantes a tener en cuenta a la hora de adquirir un ordenador personal

Vacío
En el mapa de memoria se debe reservar espacio para la ampliación de RAM. Algunos sistemas permiten agregar más de 64 Kbytes, pero éstos por lo general son "bancos intercambiables" (un circuito especial cambia la sección correspondiente a RAM dentro y fuera del mapa de memoria en función de las necesidades)



Pila
Esta sección reservada de la memoria es para utilización exclusiva de la CPU y está organizada como una estructura de datos LIFO (último en entrar, primero en salir). Un byte se puede "empujar" arriba de la pila o bien se puede "sacar" de arriba para hacerlo volver a la CPU. Cuando se efectúa una rutina GOSUB en BASIC, por ejemplo, la CPU empuja arriba de la pila la posición de la memoria a la cual finalmente habrá de retornar (RETURN). Esta pila se utiliza comúnmente en la evaluación de expresiones aritméticas y en los bucles FOR...NEXT

Buffers
En la memoria se debe reservar un buffer para el teclado, para que los caracteres no se pierdan cuando se les dé entrada a mayor velocidad de la que el programa puede procesar. También se requiere un buffer para la cassette, porque la mayoría de los sistemas operativos escriben los datos en ésta en bloques

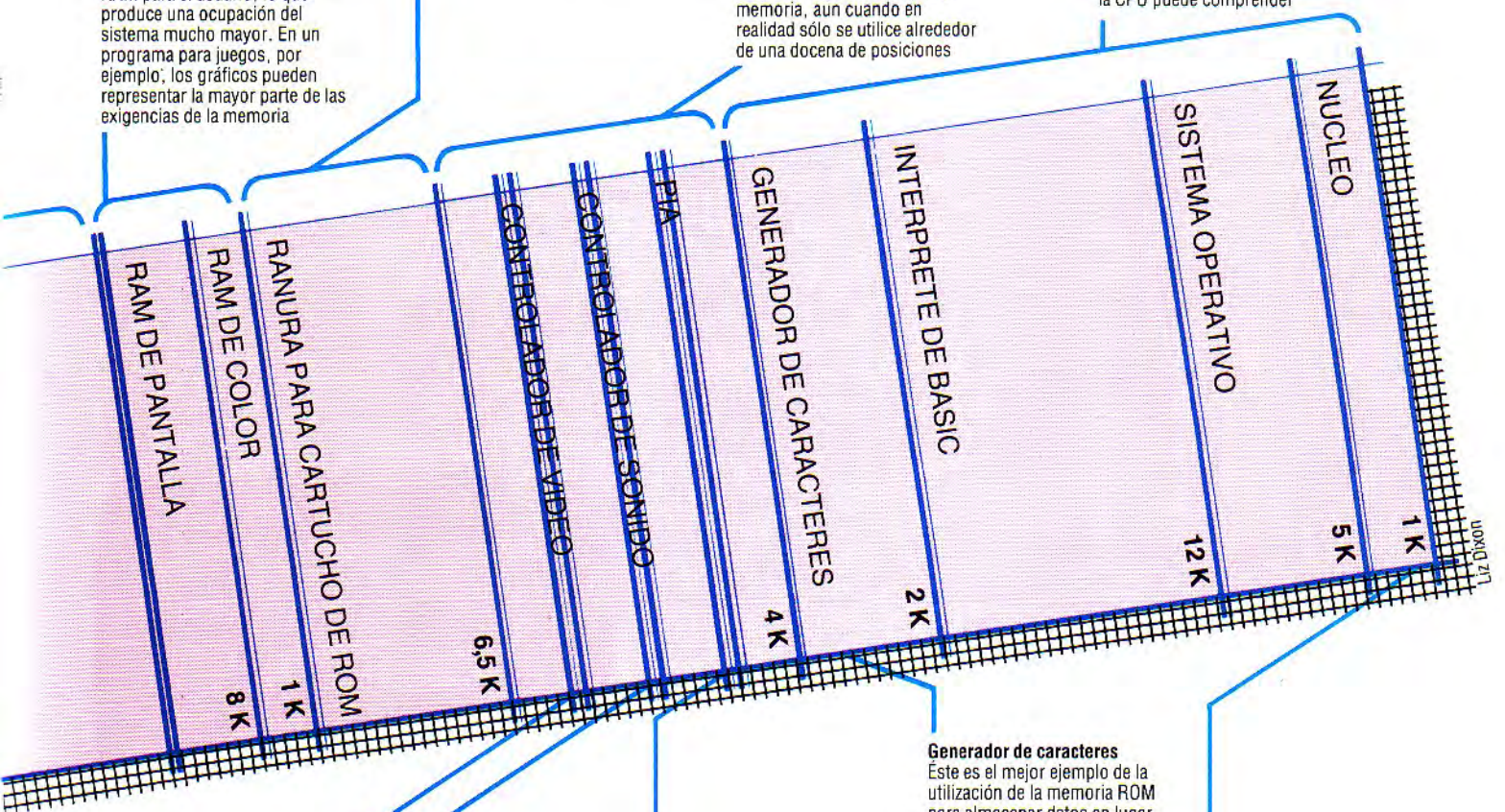
Series
Si el BASIC de su ordenador exige que especifique de antemano la longitud de todas las series, éstas se almacenarán en una tabla de modo similar a las variables dimensionadas. No obstante, de poseer "series dinámicas" cuya longitud pueda variar, entonces los datos verdaderos se almacenarán por separado en una zona de la memoria cuyas dimensiones cambian de modo constante. A intervalos, el sistema operativo promoverá una "recogida de información inservible" que simplemente limpiará la zona de la serie y eliminará los datos que ya estén obsoletos

RAM del sistema
En algunos ordenadores la RAM del sistema no está señalada como una parte de la RAM para el usuario. Por lo general se utiliza para la RAM de pantalla (donde un byte corresponde a cada una de las posiciones de caracteres de la pantalla) y para la RAM de color (donde un byte especifica los colores de fondo y de primer plano para cada posición de caracteres). Los ordenadores que posean una gran variedad de modalidades y resoluciones para gráficos necesitarán usar memoria de la RAM para el usuario, lo que produce una ocupación del sistema mucho mayor. En un programa para juegos, por ejemplo, los gráficos pueden representar la mayor parte de las exigencias de la memoria

Vacío
Cuando se utiliza un programa de un cartucho, aparece en el mapa de memoria como una ROM de ampliación. Algunas máquinas poseen conectores para ROM libres en el tablero de circuitos impresos para lenguajes adicionales. Éstos también estarán reservados en el mapa de memoria

Chips de Input/Output
La CPU se puede comunicar sólo con los dispositivos que aparecen como posiciones en el mapa de memoria, de modo que en el mapa se deben incluir todas las puertas para interfaces y otros chips. Éstos incluyen las interfaces para el teclado, la unidad de cassette, el controlador de video e interfaces externas como la impresora. Por lo general la CPU direcciona la memoria en forma de bloques (casi siempre de 4 Kbytes cada uno). Por consiguiente, los chips de Input/Output podrían ocupar 4 Kbytes del mapa de memoria, aun cuando en realidad sólo se utilice alrededor de una docena de posiciones

ROM del sistema
En un ordenador personal se utiliza ROM para almacenar la información que siempre se necesita y que es invariable. El componente fundamental de la ROM es el sistema operativo, que es el conjunto de programas en código de lenguaje máquina de los que depende el funcionamiento del ordenador. Estos programas realizan funciones tales como explorar el teclado y almacenar información en cassette o recuperarla de la misma. Otro componente es el intérprete de BASIC, que traduce los programas a las instrucciones de bajo nivel que la CPU puede comprender



Controlador de video
Los gráficos más sofisticados, como los sprites y la resolución en modalidades múltiples, se están manipulando cada vez más mediante hardware en lugar de software. El controlador o los controladores de video aparecerán en el mapa de memoria como una docena o cantidad aproximada de registros de un único byte, que determinan todos los componentes visuales, desde el color de fondo de la pantalla hasta la posición exacta de cada sprite

Controlador de sonido
Mediante software se pueden conseguir efectos sonoros rudimentarios, pero los ordenadores con voces múltiples, o con control de sonido ADSR, poseen indefectiblemente un controlador de sonido exclusivo, cuya salida alimenta un pequeño amplificador

PIA
Los adaptadores de interface periférica *(Peripheral Interface Adaptors)* se utilizan para manipular la mayoría de las interfaces simples para teclado, cassette, palanca de mando e impresora. Los chips más sofisticados (como el 6522 Adaptador de Interface Versátil) pueden convertir datos en paralelo en datos en serie, y viceversa, y poseen cronómetros incorporados que se pueden utilizar para programar o para controlar las velocidades de transmisión

Generador de caracteres
Éste es el mejor ejemplo de la utilización de la memoria ROM para almacenar datos en lugar de programas: en este caso los patrones de bits que definen la forma en que los caracteres aparecen en la pantalla. Algunos ordenadores permiten copiar en RAM todo el juego de caracteres o parte del mismo, con lo que se consigue que el usuario pueda definir otros caracteres

Núcleo
El "núcleo" (casi todas las máquinas utilizan su propia palabra para designarlo) es el corazón del sistema operativo. Cuando se enciende la máquina, la CPU salta automáticamente a esta posición y comienza a ejecutar este programa. Busca en el área de la RAM para determinar cuánta memoria hay disponible y comprueba si se encuentra enchufado un cartucho de programas. El "núcleo" también manipula las formas de entrada y salida más elementales

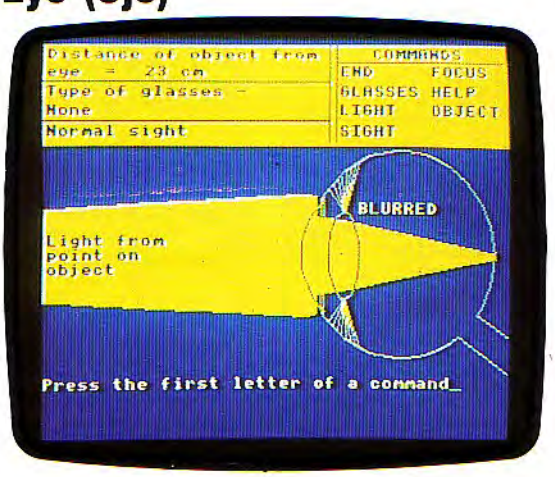
La simulación

Este tipo de software permite realizar experimentos sin aparatos ni material, y se puede usar tanto en el hogar como en la escuela

Los programas de simulación, como los conocidos juegos recreativos que permiten conducir un coche de carreras o pilotar un avión, están diseñados para proporcionar una experiencia lo más parecida posible a la realidad. Existe, no obstante, una amplia gama de software de simulación que intenta no sólo divertir sino también educar. Los programas de este tipo son muy útiles en muchas áreas de los planes de estudio escolares, y especialmente, en aquellas materias (como las ciencias) en las que la experimentación resulta peligrosa, lenta, costosa o complicada.

Los programas de simulación se pueden utilizar como herramientas educativas incluso en el hogar. Por ejemplo, en un programa denominado *Car journey* (Viaje en coche) los niños pueden aplicar sus conocimientos de aritmética y su capacidad de razonamiento para "conducir" un coche. Tal vez los programas de simulación constituyan el más apasionante tipo de software educativo que existe actualmente en el mercado. Lamentablemente, sólo están al alcance de una gama de ordenadores muy reducida: BBC Micro, Spectrum, RML 380Z y Apple, que son las máquinas más usadas en las escuelas.

Eye (Ojo)



Este programa muestra cómo funciona el ojo y cómo las diversas partes que lo componen han de estar correctamente ajustadas para permitir una visión clara. Imita la trayectoria que siguen los rayos de luz desde un objeto hasta la retina (la parte posterior del ojo donde se forman las imágenes). Usted actúa como si fuera el cerebro, controlando elementos como la distancia del objeto, el tamaño del iris y la longitud focal de las lentes, con el objeto de centrar la imagen en la retina. Se le presenta en la pantalla un diagrama del corte del ojo, con el nombre de cada parte. Utilizando la orden LIGHT se puede trazar la trayectoria de un haz de luz desde un objeto hasta el ojo. Si ha escogido correctamente las otras variables, obtendrá el mensaje IN FOCUS; de lo contrario, aparecerá el mensaje BLURRED (borroso).

Una vez se ha conseguido dominar el funcionamiento de un ojo normal, se pueden simular defectos como la miopía. Aunque originalmente se desarrolló para las clases de física y biología, este programa se suele utilizar en cursos generales de alfabetización informática, como introducción de los jóvenes en el tema de los ordenadores. Este programa de simulación lo fabrica Longmans para el BBC Micro.

Ballooning (Viaje en globo)

Ballooning es un programa educativo para el hogar destinado a los niños de entre ocho y doce años. Lo fabrica Heinemann Educational Software para el Spectrum. El usuario está al mando de un globo aerostático y ha de hacerlo volar. En la pantalla se ve un corte transversal del campo, con el globo posado inicialmente en el suelo. También se muestran cuatro instrumentos: indicador de velocidad de ascenso-descenso, medidor de la temperatura del aire, altímetro e indicador de combustible. Sólo existen dos mandos: un mechero de gas para calentar el aire y hacer que el globo se eleve, y una abertura que deja escapar el aire, haciendo que el globo descienda.

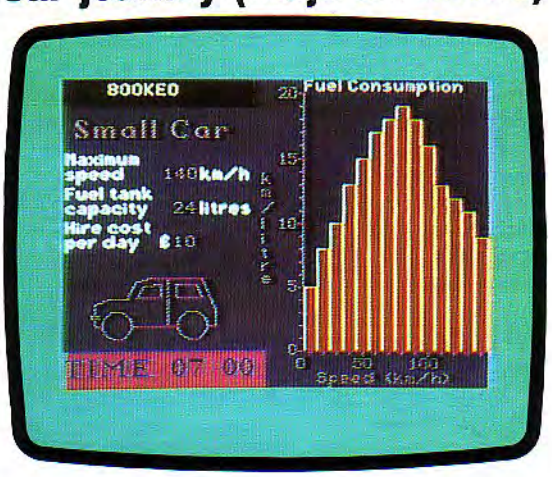
Una sencilla "lección de vuelo" enseña al usuario cómo se utilizan los instrumentos y los mandos para elevarse, volar y hacer aterrizar el globo. Una vez aprendido esto, puede emprender su propia "mi-

sión" de vuelo. Ésta consiste en hacer que el globo aterrice en unos lugares seleccionados (marcados con una X), donde el aeronauta recibe algunas instrucciones. Por ejemplo, una tarea consiste en "ayudar a un granjero a rescatar una oveja": la oveja se encontrará en un campo marcado con una S. Si el globo se quedara sin combustible, debería aterrizar para recoger más bombonas de gas.

Al cabo de algunos intentos infructuosos, en los cuales acabará estrellándose contra los árboles o experimentando otros contratiempos por el estilo, enseguida aprenderá a controlar el globo con gran precisión, valiéndose del mechero. Tampoco le llevará mucho tiempo aprender a leer los instrumentos para poder predecir cuándo debe emplear la abertura o el mechero. Quizá lo más útil sea aprender a controlar un sistema que incorpora un verdadero retardo entre la causa y el efecto.

Este programa es una simulación muy realista, resulta muy entretenido y es de los pocos que agradan por igual a niños y niñas.

Car journey (Viaje en coche)

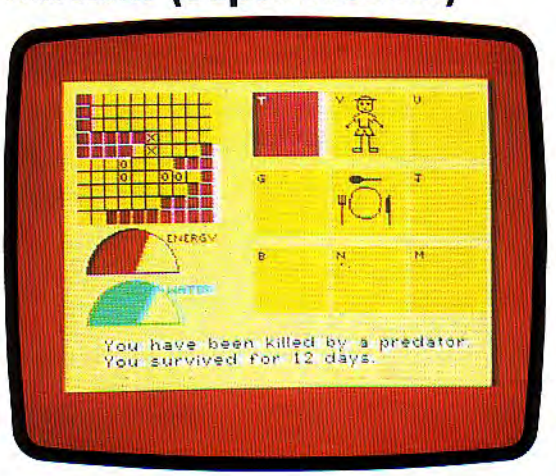


Producido por Heinemann Educational Software para el Spectrum, se trata de un programa educativo para el hogar en el que el usuario asume el papel de propietario de un pequeño servicio de reparto. Se han de tomar diversas decisiones acerca de qué contrato de reparto conviene aceptar, a qué velocidad conducir y qué tipo de vehículo utilizar. Para tomar estas decisiones se deben efectuar cálculos relativos a dinero, distancia, tiempo e incluso consumo de combustible. Se visualiza un mapa de Gran Bretaña que incluye 15 ciudades y las principales carreteras. También se muestran un velocímetro, un cuentakilómetros, un indicador de combustible y un reloj.

La primera tarea consiste en decidir por qué ciudad empezar y después el usuario ha de elegir, entre una lista de doce contratos, el que cree que podrá cumplir. Por ejemplo, un contrato es para recoger unos diamantes en Bristol a las 12 horas y entregarlos antes de las 18 horas del mismo día. Para cumplir este contrato, se debe alquilar un coche, conducirlo hasta Bristol, recoger los diamantes y llevarlos hasta Dover. Si el usuario consigue hacerlo cobra 400 libras, más una bonificación de 10 libras si llega temprano, y entonces podrá escoger otro contrato. Ha de gastar dinero en las paradas donde pernocte, en reparaciones, en gasolina y en multas por exceso de velocidad, y cuando no cumpla un contrato es objeto de una multa de 100 libras. Si acepta una carga pesada, debe descartar el coche y optar por un camión, que es más caro de alquilar, consume más y va más despacio.

Además de desarrollar el conocimiento de los vehículos y las carreteras, *Car journey* también ayuda a desarrollar aptitudes más abstractas, como son la toma de decisiones y el pensamiento lógico. Incluso enseña una teoría económica sencilla, ya que al evaluar los pros y los contras de un determinado contrato, el usuario está efectuando un análisis de coste-beneficio.

Survival (Supervivencia)



Ian McKinnell

Si alguna vez ha tratado de imaginar lo que sentiría y pensaría si fuera un león (o incluso un ratón), entonces *Survival* es el juego para usted. Le permite interpretar el papel de un animal (halcón, petirrojo, león, ratón, mosca o mariposa) y experimentar algunos de los problemas de su existencia diaria y las decisiones que ha de tomar para seguir vivo.

El mundo se representa en la pantalla mediante una cuadrícula, a través de la cual el usuario se puede mover (su posición se ilustra mediante la letra A) pulsando las teclas. Su principal preocupación consiste en hallar comida (los cuadrados marcados con un 0) y evitar los animales de rapiña (marcados con una X). A medida que se aproxima a un cuadrado marcado, una cuadrícula ampliada situada a la derecha ilustra exactamente con qué animal o con qué alimento se ha encontrado. También se muestran dos medidores que indican cuánta energía y cuánta agua le quedan. Si su nivel energético alcanza un nivel muy bajo, debe encontrar rápidamente algún alimento, y si se le acaba el agua debe acercarse a un cuadrado azul (un río); sin embargo, si accidentalmente "cayera" en un cuadrado azul, se ahogaría.

Algunos animales lo pasan peor que otros: la única fuente de alimentación de la mariposa son las flores, y puede ser difícil encontrar alguna. El halcón, en cambio, puede sobrevivir a base de caracoles, moscas y ratones, pero puede caer víctima de un cazador humano. A través de *Survival* se puede aprender qué lugar ocupan diversas especies en la cadena de alimentación y apreciar algunos de los problemas que plantea la supervivencia en un medio salvaje.

La mejor solución

A veces la solución óptima a un problema se halla de manera directa, pero con frecuencia exige la aplicación de matemáticas avanzadas. Los ordenadores asumen esta responsabilidad

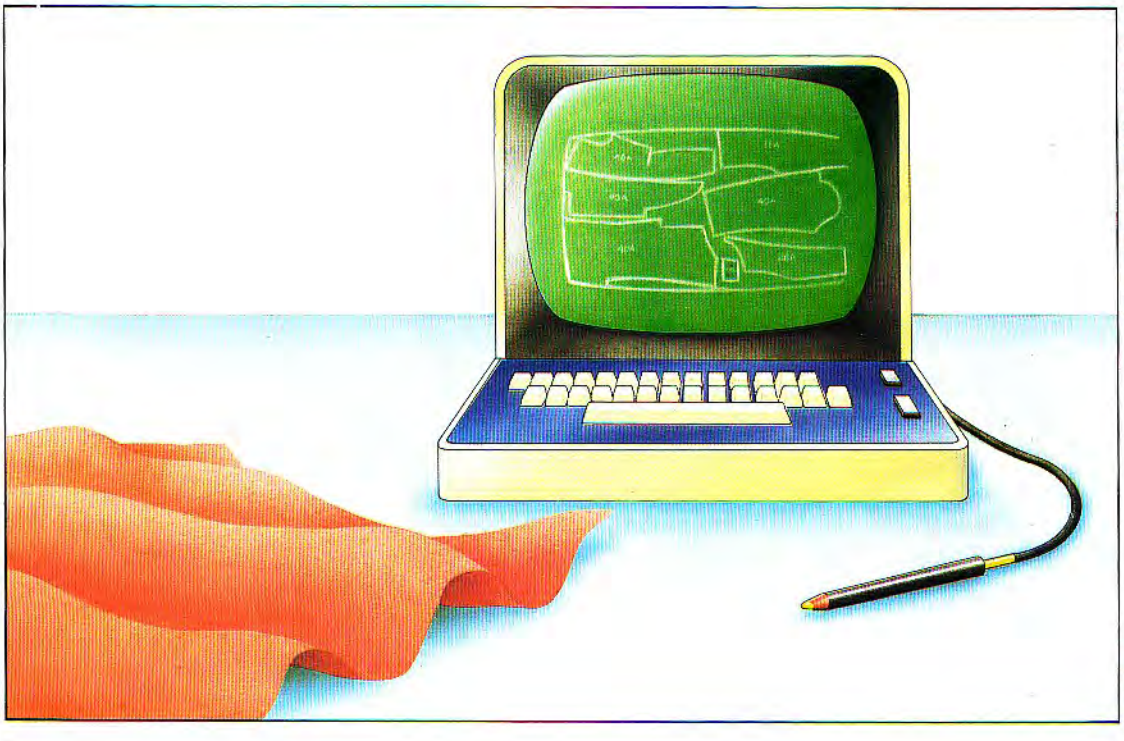
En todas las decisiones que tomamos existe invariablemente una relación de acomodo; por ejemplo, entre costo y eficacia, o entre costo y tiempo. Es improbable que obtengamos un rendimiento máximo absoluto con un costo mínimo absoluto. El resultado "óptimo" estará en algún punto medio entre ambos.

Si tomamos como ejemplo la elección entre dos marcas distintas de detergente, el razonamiento previo a la decisión podría ser el siguiente: «Si compro este detergente, los 150 g me costarán 60 pesetas, y si compro aquella otra marca, pagaré 100 pesetas por 300 g. Pero, ¿y si para obtener el mismo resultado tuviéramos que utilizar un 20 % más del detergente de menor precio? ¿Cuál de las dos marcas sería, entonces, la más barata?" Cuando todo se reduce a una medida común (en este caso, las diferencias porcentuales entre los productos) es fácil predecir la respuesta, incluso antes de realizar cálculo matemático alguno.

El concepto de "sopesar" un cálculo mediante un valor constante es bastante normal y funciona bien cuando las diferencias entre componentes similares (p. ej., el precio o el peso físico) son constantes. Pero cuando estas diferencias cambian en distinta proporción, entonces las operaciones aritméticas se vuelven más complejas y debemos recurrir a otra forma de cálculo (resolviendo una serie de ecuaciones que emplean simultáneamente las mismas variables) con el fin de llegar a la respuesta correcta. Cuando el número de variables sea pequeño, podremos optar por darles entrada en una matriz y manipularla luego. Otra vía consiste en adivinar la respuesta y luego ir modificando sucesivamente la predicción hasta que satisfaga todas las condiciones. Por supuesto, cuanto más acertada sea la predicción, menos tiempo durará el proceso.

Las técnicas de optimización de este tipo son esenciales en el comercio y la industria y se aplican universalmente, en especial en la fabricación y la construcción. La programación lineal, el método del camino crítico y los PERT (técnica de investigación y evaluación de programas) son algunos de los nombres con que se alude a este procedimiento de optimización. En su forma original se adelantó a la era del ordenador en unos 30 años y en la etapa anterior precisaba de un gran aporte de potencial humano para obtener una respuesta correcta en un plazo aceptablemente corto. Las aplicaciones de este tipo son bastante idóneas para los ordenadores personales, pero uno ha de tener presente que la operación por matriz (matriz bidimensional) requiere una gran cantidad de espacio de memoria y que la aritmética de matrices es bastante compleja. Por fortuna, existen varios paquetes de software para sistemas de microordenadores pequeños, de modo que el uso de esta técnica no presenta mayores dificultades.

Una de las áreas comerciales que se ha beneficiado considerablemente de la optimización es la con-



Ajuste perfecto
Acomodar los patrones sobre una plancha de material para minimizar el desecho constituye un buen ejemplo de optimización informatizada. Sus aplicaciones van del corte de metales laminados a la sastrería. En esta ilustración, el ordenador visualiza en la pantalla el trazado sugerido, y un operador experimentado puede hacer ajustes menores con la ayuda de un lápiz óptico

Foto original, cortesía de Burtons Tailoring

Kevin Jones

Optimización de una dieta

En este ejemplo, el objetivo es hallar la combinación óptima de cuatro productos alimenticios que satisfaga unas exigencias dietéticas mínimas especificadas al mínimo coste. Para ello hemos de decirle al ordenador: los componentes nutritivos (rosa) y el precio por unidad (azul) de cada uno de los productos, además de las exigencias mínimas de cada componente nutritivo en una semana (amarillo). El ordenador descubre el elemento más decisivo y manipula el resto de la cuadrícula en función del mismo para encontrar el equilibrio óptimo, que vemos indicado en verde. En este ejemplo, las exigencias se han satisfecho sólo con patatas y leche con un coste mínimo. (N.B.: esta dieta no es aconsejable)

	PRECIO PESETAS	PROTEÍNAS mg	HIDRATOS DE CARBONO mg	GRASA	CALORÍAS	CANTIDAD REQUERIDA
400 g PATATAS	20	10	90	0	400	8,5 kg
200 g SARDINAS	85	80	5	5	500	—
400 g CARNE	450	150	10	80	1300	—
1/2 l. LECHE	30	15	25	20	200	3,75 l
EXIGENCIA MÍNIMA POR SEMANA		250	1000	150	10 000	

Kevin Jones

fección de prendas de vestir. El género normalmente viene en unidades de anchura estandarizada (y, algunas veces, también de longitud estandarizada), y el problema del fabricante consiste en minimizar el desecho de material al cortarlo, prestando al mismo tiempo atención a factores como la dirección del pelillo del tejido.

En uno de los talleres de confección más avanzado de Europa, la disposición de los patrones de las piezas sobre un corte de tela para hacer trajes a medida, se calcula utilizando técnicas de optimización, y el resultado sugerido se visualiza en una unidad de representación visual. En este punto, empleando procedimientos de programación orientada a un objeto (véase p. 262), se solicita al operador del ordenador su opinión, basada en sus conocimientos y su experiencia, para mejorar el cálculo del ordenador. Como promedio, el operador introduce mejoras en una de cada cinco ocasiones.

Debido a que los requerimientos de cada tarea, o de cada prenda, son diferentes, se trata de un excelente ejemplo de la utilización inteligente de una optimización informatizada de bajo nivel combinada con la experiencia del operador. Procedimientos de mayor extensión se emplean en industrias que cortan reiteradamente objetos idénticos en un material en planchas, donde se desarrolla todo el proceso de optimización. Dado que la operación de corte o troquel forma parte de una cadena de producción, se repetirá miles de veces. En este caso, el coste del proceso de optimización dividido por el número de unidades fabricadas se cubre con creces con lo ahorrado en material desechable.

El método del camino crítico (MCC), como sugiere su nombre, es un procedimiento para determinar la sucesión de los trabajos más importantes de un proceso de fabricación o construcción, es decir, la parte del trabajo susceptible de retrasar las demás etapas si no se completa según el plan. Su regulación de tiempo es muy estricta: el período requerido para la ejecución de un segmento debe corresponder a su valor en el diagrama o la tabla del MCC. Se utiliza más comúnmente durante la etapa de planificación de los proyectos de construcción, de modo que los constructores puedan destinar hombres y materiales a las diversas fases del proyecto en un orden correcto: fontanería antes de pavimentos, pintores después de yeseros. También en este caso existen paquetes de software para una amplia gama de microordenadores.

Aunque las operaciones aritméticas del proceso de optimización puedan resultar un tanto intimidatorias para quien no posea experiencia, no se puede negar el éxito y la eficacia de esta técnica. Es una de las pocas tareas de procesamiento de números que normalmente se efectúan en microordenadores pequeños y es un componente importante de los sistemas de inteligencia artificial, que, como sucede con frecuencia, reproducen el proceso de aplicación del sentido común.

Mejores autopistas
Aparte de los factores sociales, el diseño y el trazado de las autopistas, tanto en las ciudades como en el campo, depende en gran medida de las técnicas de optimización. Lo que más le preocupará al ingeniero será la pendiente de las colinas y el ángulo de las curvas, pero el granjero a quien expropien sus tierras se regirá por unos criterios totalmente distintos. Cuando se planifica una nueva carretera se reúne una inmensa cantidad de datos, que sirven para confeccionar un modelo exhaustivo de la situación. Este modelo se utiliza luego para diversos fines, desde representaciones gráficas hasta la optimización de la ruta

Cortesía del Ministerio de Transporte británico

Acorn Electron

En dos años, entre la aparición del BBC Modelo B y el Electron, de Acorn, el desarrollo de la tecnología del microordenador ha sido espectacular

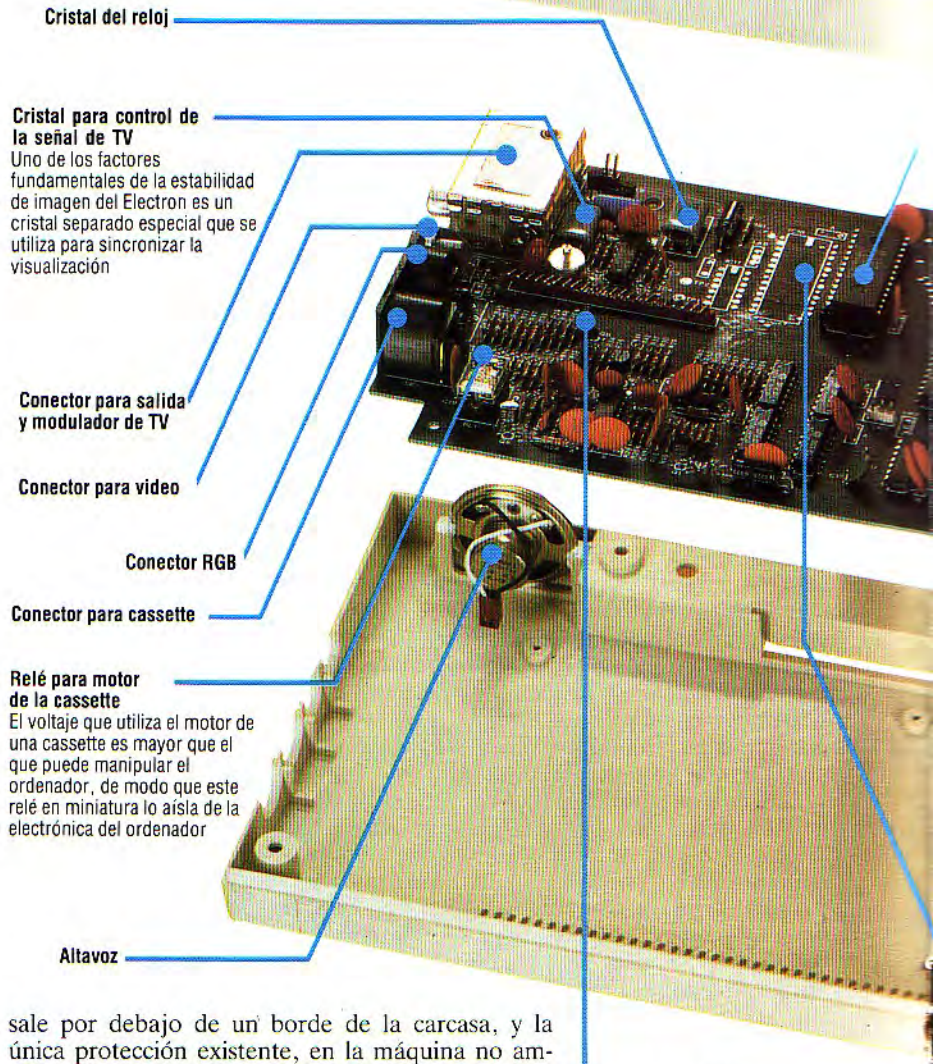
El Acorn Electron es un ordenador elegante que responde a la impresión inicial que transmite: una máquina sólida y bien diseñada. Como versión a escala reducida del BBC Micro, su rendimiento es inferior, pero resulta más cómodo de utilizar. La mayoría de los elementos del BBC Micro se han incorporado al Electron. Por ejemplo, la orden SOUND se utiliza en ambas máquinas en conjunción con la orden ENVELOPE para sintetizar distintos tipos de instrumentos musicales.

El Electron dispone de todas las modalidades de gráficos del BBC Micro, con la excepción del teletexto (MODE 7), que en el BBC se genera mediante un chip especial. El tablero de circuitos del Electron no dispone de este chip, y por ello las visualizaciones al estilo teletexto sólo se pueden producir volviendo a definir la mayoría de los caracteres e imitando el teletexto mediante la utilización de MODE 6 (que, no obstante, está limitada a dos colores). Esto es un lástima, porque en el BBC Micro la modalidad de teletexto es una forma muy económica de producir visualizaciones muy complejas sin necesidad de usar gran cantidad de memoria.

Las facilidades de input y output son, asimismo, menores que en el BBC Micro. La salida visual se realiza a través del canal 36 de TV y por medio de video compuesto y conectores RGB para los monitores monocromáticos o en color. Pero aparte de la conexión para cassette no hay ninguna interface utilizable de inmediato.

La ampliación se puede efectuar a través de un gran conector marginal situado en la parte posterior de la máquina. Lamentablemente, éste sobresale por debajo de un borde de la carcasa, y la única protección existente, en la máquina no ampliada, es una cubierta plástica. En el manual no se proporcionan detalles acerca de las señales que produce, ni ninguna sugerencia relativa a lo que en él se puede conectar. Pero es evidente que aquí es posible enchufar algún tipo de caja de ampliación, porque en las proximidades de la carcasa hay moldeados unos conectores de filamentos de bronce, que se emplean para establecer una conexión mecánica entre el ordenador y el accesorio.

El BASIC incorporado es la ya bien conocida versión del BBC; pero en este caso se ha ampliado considerablemente y posee muchos elementos que

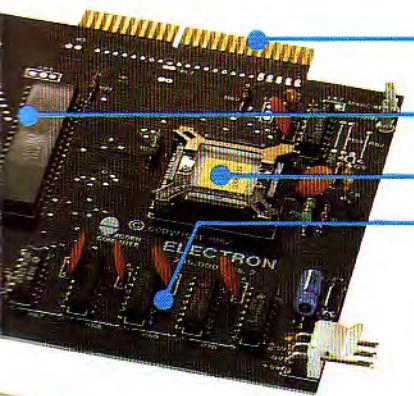


Los creadores del Acorn Electron
Detrás del Electron está la inteligencia de Chris Curry (izquierda) y Herman Hauser (derecha), quienes fueron asimismo responsables en gran medida del diseño del BBC Micro. Curry, ingeniero de desarrollo, trabajaba para Clive Sinclair cuando éste contrató a Hauser. Luego, Curry y Hauser fundarían Acorn

Judy Goldhill

Teclado
Es uno de los mejores teclados de ordenador personal, con verdaderas teclas al estilo de una máquina de escribir y de muy buena calidad. De hecho, es muy similar al del BBC Micro. No hay teclas de función separadas, pero estas facilidades las proporciona la tecla de mayúsculas, que, al ser pulsada junto con una tecla numérica, convierte a ésta en una tecla de función. Lo mismo sucede con las teclas de letras y con tres de las teclas de puntuación, que producen palabras tecla en BASIC si se digitan mientras se mantiene pulsada la tecla de mayúsculas

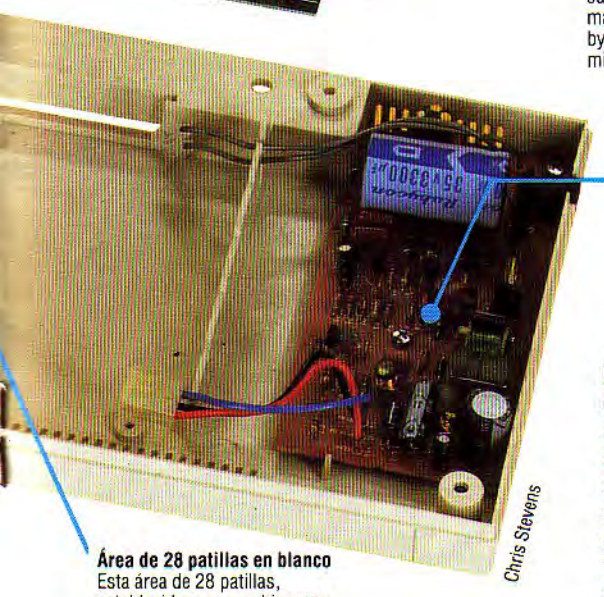
Conector para ampliación
No se proporcionan detalles acerca de los valores de las patillas ni de los tiempos de las señales, pero a través de este conector se dispondrá de la mayor parte del bus del sistema y de las líneas TTL y de potencia. Por consiguiente, la ampliación posible sería considerable



CPU
Controlando la máquina hay un procesador 6502A estándar, sincronizado a 1,79 MHz. Éste toma realmente las decisiones, algo que la ULA no puede hacer por sí sola

ULA
Esta ULA (*Uncommitted Logic Array:* disposición lógica no comprometida) es la más grande que se ha fabricado hasta hoy. Aparte de la ULA, la CPU 6502, la ROM y la RAM, en el tablero sólo hay otros nueve chips, todos ellos de lógica TTL estándar; cada uno de ellos proporciona tan sólo un puñado de puertas lógicas

RAM
Los bytes se cargan de la RAM a la CPU en dos mitades. Primero se accede a los cuatro bits inferiores (un bit proveniente de cada uno de los cuatro chips), y a continuación a los cuatro superiores. En la mayoría de las máquinas, los ocho bits de cada byte se almacenarían en el mismo chip



Sistema de circuitos acondicionador de potencia
El Acorn Electron es una máquina inusual por cuanto requiere una fuente de alimentación eléctrica de 19 v. Ésta ofrece la ventaja de ser más estable, pero necesita un circuito más complejo para ser utilizada por el ordenador

Chris Stevens

Área de 28 patillas en blanco
Esta área de 28 patillas, establecida para un chip y con una conexión cercana vacía, sugiere que se podría agregar otra ROM adicional, o bien que se podría emplear un tipo distinto de chip

hacen grato el uso de la máquina. De especial utilidad es la rutina OSCLI, que permite que un programa en BASIC le envíe órdenes directamente al sistema operativo, y gracias a ella los usuarios experimentados pueden superar algunas de las limitaciones del BASIC. El paquete ensamblador, que es un elemento exclusivo del BASIC del BBC, se ha ampliado también. Posee palabras tecla adicionales para definir el almacenamiento de variables y la impresión de series, dos trabajos rutinarios en lenguaje Assembler.

El rendimiento del Acorn Electron es superior a la media. La imagen es muy uniforme y nítida, con una buena definición y pureza de color. Cuando disponga de algunas facilidades de ampliación, como unidades de disco, el Electron seguramente se convertirá en una máquina popular.

ACORN ELECTRON

DIMENSIONES
340 x 160 x 65 mm

CPU
6502

VELOCIDAD DEL RELOJ
1,79 MHz

MEMORIA
64 Kbytes de ROM
32 Kbytes de RAM (sin ampliación en el tablero)

VISUALIZACION EN VIDEO
Hasta 32 líneas de 80 caracteres. Ocho colores con determinación independiente de fondo y primer plano. 127 caracteres predefinidos y 255 definibles por el usuario

INTERFACES
Canal 36 de TV, video compuesto, TTL RGB, cassette, bus del sistema (no documentado)

LENGUAJE SUMINISTRADO
BASIC del BBC con ensamblador en línea

OTROS LENGUAJES DISPONIBLES
Debería funcionar con otros lenguajes AcornSoft como FORTH y LISP, siempre que se basen en RAM. Los lenguajes basados en ROM, como el BCPL y el PASCAL, son incompatibles con la máquina no ampliada

VIENE CON
Manual de instalación y de BASIC, cable de TV, transformador de potencia, cassette de presentación

TECLADO
Estilo máquina de escribir, con 56 teclas. Entrada de palabras tecla en BASIC mediante una sola tecla. Diez teclas de función definible por el usuario

DOCUMENTACION
Excelente. Hay muchos detalles a disposición del programador interesado y del amante de la experimentación. Todas las palabras tecla en BASIC se explican por separado; hay una sección muy bien realizada acerca del lenguaje Assembler. También están bien descritas las funciones del sistema operativo. Gracias a esta riqueza de información, con esta máquina se puede llevar a cabo la mayoría de las tareas con relativa facilidad

Impresión a chorro

Se puede obtener un impreso a todo color gracias a las modernas impresoras que proyectan sobre el papel gotas de tinta de color

Los distintos tipos de mecanismos de impresión que están a disposición del usuario de ordenadores personales producen una impresión de calidad variable. Los mejores resultados se obtienen con las impresoras de impacto de carácter completo (la rueda margarita constituye un ejemplo excelente de este tipo de impresoras); por el contrario, las impresoras electrostáticas y térmicas ofrecen una reproducción más pobre. No obstante, la impresora matricial (véase p. 74), aunque es ruidosa y produce una tipografía de calidad sólo discreta, es el sistema más popular para el uso con ordenadores personales.

Cuando aparecieron por primera vez dispositivos de impresora-plotter como el Tandy CGP 115, se hicieron más evidentes las limitaciones de las impresoras matriciales. Las máquinas impresoras-plotter utilizan lápices esferográficos en miniatura para crear sobre el papel caracteres completos y gráficos lineales, y con frecuencia éstos suelen ser a cuatro colores. Pero las impresoras que más probabilidades tienen de superar en popularidad a la impresora matricial operan sobre el principio de disparar un flujo de gotas microscópicas de tinta sobre una hoja de papel de acuerdo con patrones controlados. Son las *impresoras de chorro de tinta*.

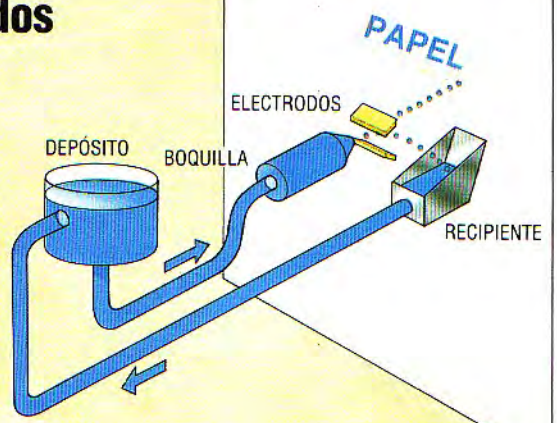
Ya bien establecidas en los sectores industrial y comercial (al igual que la también sofisticada impresora por láser), estas impresoras se están comenzando a introducir en el mercado de ordenadores personales. Funcionan disparando gotas de tinta desde un depósito hasta el extremo de una boquilla muy delgada. Antes de ser expelida, cada pequeña gota de tinta atraviesa un electrodo y recibe una carga eléctrica. El mecanismo de la válvula normalmente es de material piezoeléctrico, lo que permite moldear las gotas mediante vibraciones de muy alta frecuencia.

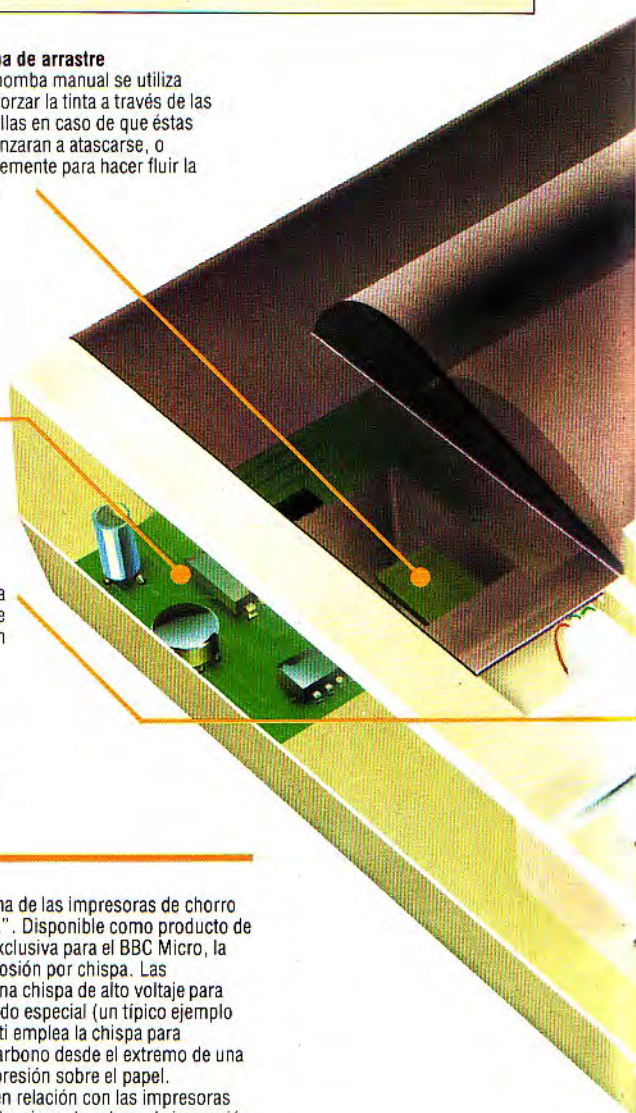
Cuando la gota sale de la boquilla queda suspendida por un campo eléctrico, que también la impulsa hacia el papel. La hoja de papel está extendida sobre una lámina de metal (y no sobre un cilindro de plástico duro o un rodillo de caucho como sucede en una impresora por impacto). La lámina de metal se carga con el potencial opuesto al que retiene la gota y, como las cargas contrarias se atraen, contribuye a llevar la tinta hacia el papel. Esta técnica puede parecer poco fiable pero, sorprendentemente, se producen muy pocas incidencias. Lo peor que puede llegar a ocurrir es que se atasque la boquilla o que las gotas de tinta adquieran un tamaño mayor que el ordinario.

En principio, una impresora de chorro de tinta trabaja, al igual que una impresora matricial, con un solo martillo de impresión. La serie de caracteres ASCII que llega a la impresora se almacena en un buffer hasta que éste está lleno o hasta que reci-

Misiles dirigidos

Las primeras impresoras de chorro de tinta utilizaban un sistema más sofisticado y eran muy caras. En el interior de la boquilla, un dispositivo piezoeléctrico emitía un flujo constante de gotas de tinta cargadas. Estas gotas se podían guiar verticalmente mediante dos electrodos, mientras la cabeza se movía a través del papel. Cuando no se requería ninguna marca, las gotas se podían conducir a un recipiente y después volverlas a reciclar dentro del depósito principal





Bomba de arrastre
Esta bomba manual se utiliza para forzar la tinta a través de las boquillas en caso de que éstas comenzaran a atascarse, o simplemente para hacer fluir la tinta

Tablero de circuitos
Esta impresora contiene su propio microprocesador 6809 y sus propias ROM y RAM. Todos los datos que entran se deben almacenar en buffer, porque el mecanismo imprime tan sólo una línea de puntos en cada ocasión que la cabeza pasa

Bloqueo de cabeza de impresión
El mecanismo de chorro de tinta es muchísimo más delicado que otros mecanismos de impresión y, cuando no está en uso, la cabeza se ha de bloquear en la posición de descanso. El procedimiento operativo inmediatamente posterior a su encendido no es complicado, pero si no se realiza de manera correcta, se puede dañar la máquina

Caracteres chispeantes
Una variación interesante sobre el tema de las impresoras de chorro de tinta es la impresora de "tinta seca". Disponible como producto de Olivetti y como impresora de Acorn exclusiva para el BBC Micro, la unidad se basa en el principio de la erosión por chispa. Las impresoras de este tipo suelen usar una chispa de alto voltaje para quemar un agujero en un papel plateado especial (un típico ejemplo es la impresora ZX). El sistema Olivetti emplea la chispa para transportar diminutas partículas de carbono desde el extremo de una barra renovable para efectuar una impresión sobre el papel.
La impresora ofrece ciertas ventajas en relación con las impresoras matriciales convencionales: es casi silenciosa, la cabeza de impresión es muy ligera (lo que elimina la necesidad de motores muy potentes) y se puede utilizar prácticamente con cualquier tipo de papel. Los únicos inconvenientes son que la velocidad de impresión es muy baja, la cabeza imprime sólo una línea de puntos cada vez que pasa por el papel y la "tinta" tiende a emborronarse

be un "retorno de carro". La impresora examina entonces uno por uno los caracteres y consulta sus correspondientes patrones en la ROM. Por lo general, cada carácter está compuesto por un conjunto de puntos dispuestos en una cuadrícula de ocho por ocho, y la impresora construye estos patrones sobre el papel. Se requieren ocho pasadas de la cabeza de impresión para crear cada una de las líneas de caracteres, pero el ritmo se acelera, permitiendo que la impresora opere en ambos sentidos. Mientras se imprimen de este modo los caracteres del primer buffer, el buffer siguiente se está llenando para imprimir sus caracteres apenas se vacíe el primero. La única diferencia entre la impresora de chorro de tinta y la matricial de un solo martillo de impresión es que la primera dispara electrónicamente gotas de tinta cargadas contra la página, mientras que en la segunda una aguja imprime a través de una cinta cubierta de tinta.

En su forma comercial, las impresoras de chorro de tinta consiguen producir una hoja impresa en unos segundos. No obstante, la calidad de la impresión puede depender de la calidad del papel: cuanto más absorbente es el papel, más tinta chupa y menos nítida resulta la imagen. En las mejores condiciones, las impresoras de chorro de tinta pueden proporcionar un resultado de calidad varias veces superior al de una impresora matricial. Son perfectamente adecuadas para una impresión empresarial a gran escala. Si necesita una impresión de alta calidad y a gran velocidad, la única solución es la impresora por láser (que trabaja según los mismos principios que las fotocopiadoras).

La aplicación más reciente y, quizá, la más interesante, del principio del chorro de tinta, es la que ha utilizado Tandy en su máquina CGP 220. Con ella, el usuario de ordenadores personales ha conseguido por fin una verdadera impresión en color. Además de imprimir en negro, la Tandy CGP 220 contiene depósitos y boquillas separadas para tintas de color magenta (rojo-azul), cyan (azul-verde) y amarillo. Estos colores podrán resultar algo extraños a aquellas personas acostumbradas a trabajar con los gráficos de color en una pantalla de televisión, pero para los pintores son los equivalentes al rojo, verde y azul, y mezclándolos entre sí se puede conseguir todo el espectro.

Electrodo metálico
Inmediatamente detrás del papel hay una lámina metálica, que se carga con el potencial opuesto al que se aplica a las gotas de tinta. Ello produce una aceleración de las gotas hacia el papel

Motores de acción
El movimiento transversal de la cabeza de impresión se obtiene mediante un motor convencional, mientras que el avance del papel lo acciona un motor a impulsos, al igual que en las impresoras matriciales

Conexiones flexibles
La mayoría de las impresoras incorporan unos cables planos flexibles entre el circuito impreso y la cabeza de impresión. Una impresora de chorro de tinta tiene el problema adicional de alimentar con cuatro tintas distintas un dispositivo de movimiento rápido

Cabeza de impresión
Contiene cuatro boquillas (una para cada una de las tintas) y celdas piezoeléctricas, que proporcionan impulsos de presión que crean las gotas de tinta

Paquetes de tinta
Para evitar que los conductos de salida se bloqueen continuamente, se debe emplear una tinta especial. La tinta negra viene en un paquete separado de la roja, la azul y la amarilla, porque es la que se utiliza con mayor frecuencia

Prueba sonora

La síntesis de sonido con el Dragon 32

El Dragon 32 se suministra con un solo oscilador de onda cuadrada para la programación de sonido, pero las órdenes para sonido, extraordinariamente sencillas, de que dispone el BASIC Microsoft Extended Colour permiten la creación de series musicales que con una sola orden ejecutan una melodía aceptable. Lamentablemente, no puede generar ruido. Lo cual resulta sorprendente, ya que es difícil imaginar un juego recreativo que no requiera ruido en algún punto para hacer más interesantes los efectos sonoros.

La orden SOUND sirve sólo para efectos sonoros y el formato es el siguiente:

```
SOUND, P,D
```

donde: P = Tono (1-255) y D = Duración (1-255). El tono es muy impreciso y guarda escasa relación con una escala musical normal, aunque se puede conseguir una aproximación al do central con el valor 89 y el la de diapasón a 440 Hz se halla en 159 más o menos. La duración es igualmente inexacta, pero 16 se aproxima a un segundo, 32 equivale a casi dos segundos y así sucesivamente.

Este programa ilustra cómo se puede utilizar SOUND para un efecto especial; en este caso, con un poco de imaginación, un OVNI despegando:

```
10 FOR P = 10 TO 170 STEP 10
20 FOR D = 16 TO 1 STEP -1
30 SOUND P,D
40 NEXT D
50 NEXT P
```

PLAY puede establecer el tono, la duración y el volumen exactos de una nota. Asimismo, puede especificar una serie de notas a ejecutar (PLAY) con una pausa de tiempo variable entre ellas. Esto hace que resulte muy sencillo crear melodías con diferentes longitudes de notas y silencios, que se ejecutarán (PLAY) con esta única orden:

```
PLAY "T;O;V;L;N;P"
```

donde: T = tiempo (T1-T255); O = octava (O1-O5); V = volumen (V0-V15); L = longitud de nota (L1-L255): N = número de la nota (1-12 o nombre de la nota); y P = pausa antes de la nota siguiente (P0-P255).

No es estrictamente necesario utilizar el punto y coma entre uno y otro parámetro, pero conviene incluirlo para mayor claridad. Este ejemplo es una representación muy arbitraria, porque los parámetros se pueden expresar en cualquier orden. T, O, V y L conservan sus valores hasta que se especifiquen de otra manera. De hecho, T, O, V, L y P corresponden a T2, O2, V15, L4 y P0, respectivamente, si no se especifica en otro sentido, de modo que no hace falta incluirlos siempre en la sentencia PLAY.

Cuando se recurre a una medida del tiempo, como ocurre con L y P, los valores especificados se pueden considerar como si fueran medidas de compás y fracciones: así L1 o P1 es un tiempo entero, L2 o P2 medio tiempo, etc. Los tiempos reales se seleccionan mediante el parámetro T, donde T1 es lento (una nota posee una duración larga) y T255 es muy rápido (una nota posee una duración corta). Además, las longitudes de las notas se pueden definir con más flexibilidad mediante la adición de puntos,

Pasatiempos luminosos

Continuamos con las capacidades gráficas del BBC Modelo B

El BASIC del BBC no proporciona la gama completa de órdenes de alta resolución de que disponen algunos microordenadores. Por ejemplo, no posee órdenes CIRCLE ni PAINT. No obstante, la mayoría de las facilidades se pueden imitar utilizando unas pocas líneas del BASIC del BBC.

La pantalla para gráficos posee las mismas coordenadas, sin considerar el nivel de resolución seleccionado, y los ejes se originan en el extremo inferior izquierdo. Las siguientes órdenes permiten controlar la pantalla para gráficos:

```
MOVEx,y
```

Esta orden desplaza el cursor para gráficos hasta el punto de las coordenadas (x,y), pero no traza una línea. Observe que el cursor para gráficos se puede desplazar independientemente del cursor para texto.

```
DRAWx,y
```

Al digitar la orden DRAW se dibuja una línea desde la posición corriente del cursor para gráficos hasta el punto de la pantalla de las coordenadas (x,y).

```
PLOTk,x,y
```

PLOT es una orden muy versátil; su función la gobierna el valor otorgado a la variable k:

Valor de k	Función
0	movimiento en relación al último punto
1	dibujar línea desde origen en color del primer plano
2	dibujar línea desde origen en color inverso
3	dibujar línea desde origen en color del fondo
4	igual que MOVE
5	igual que DRAW
6	igual que DRAW pero en este caso en color inverso
7	igual que DRAW pero en este caso en color del fondo

como L1... o L5., donde cada punto incrementa la longitud de la nota en la mitad de su valor normal. Por consiguiente, L1... = $1+\frac{1}{2}+\frac{1}{2}+\frac{1}{2}$ = 2 1/2 notas y L5. = $\frac{1}{5}+\frac{1}{10}=\frac{3}{10}$ nota.

No existe ninguna forma categórica de poder representar la relación entre nota y tiempo. Los valores requeridos pueden variar para cada melodía; la mejor forma de seleccionarlos es mediante ensayos y pruebas, que permiten ir eliminando errores. Esto puede ocupar algo de tiempo, pero confiere a la orden una gran flexibilidad.

El parámetro O especifica la octava en la cual se ha de tocar la siguiente nota. O1 comienza con el *do* a 131 Hz y O5 termina con el *si* a 2093 Hz. El do_3 (central) empieza la octava O2, que es la octava que se puede omitir. Dentro de una octava, las notas se pueden especificar de dos maneras. En primer lugar, empleando un número para cada una de las doce notas que componen la octava. Así:

1	2	3	4	5	6
DO	DO#	RE	RE#	MI	FA
7	8	9	10	11	12
FA#	SOL	SOL#	LA	LA#	SI

Esto permite que se pueda especificar una nota como una variable dentro de una octava seleccionada. La segunda manera consiste en emplear directamente el nombre de la nota requerida para que sea más fácil comprender la sentencia en un listado.

Lo que antecede quedará más claro en un ejemplo. La siguiente orden toca fa(6) en la octava omitida O2, durante media unidad temporal (L2) al volumen omitido V15. Luego hace una pausa durante un cuarto de unidad de tiempo (P4), emitiendo después a volumen V20 y durante una unidad temporal L1 la nota *la#* en una octava más alta, la O3. El tiempo se establece en T3:

```
PLAY "T3;L2;6;P4;O3;V20;L1;LA#"
       <  FA  >     <     LA#     >
```

intervalo

Además, los parámetros T, O, V y L se pueden variar en proporciones preestablecidas en la propia orden mediante la adición de un sufijo:

Sufijo	Efecto
+	Suma uno al valor habitual
−	Resta uno al valor habitual
>	Multiplica el valor habitual por dos
<	Divide el valor habitual por dos

El formato es: T+, T−, T> o T< para cada parámetro.

La más útil configuración del Dragon es su capacidad para tocar (PLAY) melodías empleando subseries. Estas primero se definen y luego se tocan (PLAY) en cualquier orden o se repiten:

```
10 A$ = "FA;LA#;SOL"
20 B$ = "DO;RE#;FA;P4;XA$;"
30 PLAY B$
```

Esto define A$ y después lo incluye en B$ como la subserie XA$. La melodía resultante es DO-RE#-FA-P4-FA-LA#-SOL. Esta técnica se puede prolongar en la medida de lo necesario cuando en una pieza musical las secuencias de notas se repiten cierta cantidad de veces. En todos los casos se debe incluir el punto y coma después de la subserie, como en el ejemplo anterior de XA$.

Con números mayores se repiten estas ocho funciones pero con nuevos efectos, como líneas de puntos en lugar de líneas continuas. Los valores de k entre 80 y 87 desempeñan una función particularmente útil. PLOT80,x,y une el punto (x,y) con los dos puntos trazados previamente para formar un triángulo. El triángulo se rellena luego con el color habitual del primer plano. Esto proporciona el único recurso sencillo para pintar (PAINT) formas gráficas.

VDU x es similar a otra orden muy utilizada en BASIC: PRINT CHR$(x). En el artículo anterior sobre los gráficos del BBC Micro vimos que VDU puede ir seguida de una serie de números. VDU v,w,x,y,z equivale a:

```
PRINT CHR$(v);CHR$(w);CHR$(x);CHR$(y);
  CHR$(z)
```

Las órdenes VDU permiten al usuario acceder a aquella parte del sistema operativo del BBC que controla los gráficos y la visualización en pantalla. Si bien estas órdenes se pueden emplear dentro de programas en BASIC, en realidad funcionan independientemente del lenguaje empleado. Por consiguiente, las mismas órdenes VDU se podrían utilizar para una visualización de gráficos en PASCAL o en cualquier otro lenguaje de los que se ofrecen para el BBC. Todas las facilidades para gráficos en BASIC que hemos analizado hasta el momento presente se pueden ejecutar también por medio de la orden VDU adecuada.

Definir caracteres es muy sencillo en el BBC Micro. VDU 23 controla esta función. En capítulos anteriores hemos visto que los códigos ASCII normales se construyen a partir de un bloque de ocho por ocho pixels. Los pixels visibles se pueden representar con un 1 en binario y los no visibles con un 0. Cada fila de ocho bits se puede convertir a su equivalente en decimal, dando un total de ocho números decimales para definir un carácter. VDU 23 permite al usuario volver a definir el carácter con un código ASCII entre 224 y 255. Por ejemplo:

```
10 REM DEFINIR UN CARACTER
20 MODE 2
30 VDU 23,240,16,56,124,146,16,16,16,0
40 PRINT CHR$(240)
50 END
```

Este breve fragmento de programa vuelve a definir el carácter con el código ASCII 240 para crear una forma de flecha. Los últimos ocho números definen esta nueva forma, y la línea 40 imprime (PRINT) el carácter en la pantalla.

VDU 24 y VDU 28 controlan, respectivamente, la creación de "ventanas" de gráficos y de texto en la pantalla. Empleando estas funciones, la salida de gráficos y de texto a la pantalla se puede limitar a zonas definibles. Esto puede resultar de especial utilidad al diseñar programas interactivos, para los cuales es deseable una pantalla dividida. Lo único que se requiere para definir una ventana de gráficos es especificar las coordenadas para las esquinas inferior izquierda y superior derecha.

MODE 1
Este corto listado de programa dibuja en la pantalla una flor de colores en espiral empleando la resolución MODE 1. Observe la utilización de triángulos rellenos para producir los pétalos de la flor

```
10 REM FLOR
20 CLS
30 MODE 1
40 FOR D = 1 TO 3
50 A = 600 : B = 500
60 MOVEA,B
70 FOR C = 1 TO 550 STEP3
80 GCOL0,RND(3)
90 S = (C/(RND(5)+10))
100 X = S*5*SEN(C/16)+A
110 Y = S*5*COS(C/16)+B
120 PLOT85,X,Y
130 NEXT C
140 NEXT D
150 END
```

El patrón en espiral se produce mediante la combinación de seno y coseno en las líneas 100 y 110. Normalmente, esta relación entre las coordenadas *x* e *y* produce un círculo, pero el bucle FOR...NEXT incrementa gradualmente el radio C, produciendo el efecto en espiral. Las coordenadas del centro de la espiral, A y B, se podrían alterar para cambiar la posición de la flor

Ejecución ficticia

Para poder utilizar los archivos de datos, primero es necesario crearlos en forma esquemática y luego proveerles de información

Al terminar el capítulo anterior, nos preguntábamos: ¿cómo podemos hacer que un programa lea un archivo que no existe (en cinta o en disco) cuando se ejecuta por primera vez el programa? Con toda probabilidad desearemos que la primera tarea que ejecute el programa sea leer los archivos de datos y asignarles esta información a las matrices o variables. No obstante, si insistimos en escribir primero el archivo, cada vez que se ejecute el programa tendremos que prestar gran atención a la programación para no perder los datos del archivo.

Por fortuna, existe una solución muy sencilla. Muchos paquetes de software comercial incluyen un programa de "instalación" o "preparación" que se debe ejecutar antes de que se pueda utilizar el propio programa, y ésta es la vía que vamos a seguir. Normalmente, dichos programas le conceden al usuario un pequeño margen de "acomodación" (como seleccionar si la impresora a utilizar será una Epson o una Brother, en paralelo o en serie, etc.), pero además crean archivos de datos que con posterioridad serán empleados por el programa principal. Recuerde que, a diferencia de lo que sucede con los archivos de programas, se puede acceder a los archivos de datos a través de cualquier programa (véase p. 316).

Para solucionar nuestro problema y permitir que se pueda llevar a cabo *LEARCH* (la rutina que lee los archivos y les asigna los datos a las matrices), podemos escribir una programación de preparación muy sencilla, simplemente para abrir un archivo y escribir en él un valor ficticio. Elegiremos un valor que el programa formal pueda reconocer como un registro no válido para la agenda de direcciones. Un valor apropiado podría ser la serie de caracteres @VACIO porque es muy poco probable que algún nombre o dirección comience con esta serie particular. *LEARCH* se habrá de modificar ligeramente de modo que cuando abra el archivo y lo lea verifique su valor antes de seguir adelante. Si su ordenador no posee el símbolo @, tendrá que reemplazarlo por '!' u otro carácter, mientras ésta sea una serie que no se produzca naturalmente en su agenda de direcciones. De todos modos, veamos primero el programa de preparación:

```
10 REM ESTE PROGRAMA CREA UN ARCHIVO DE
   DATOS
20 REM PARA QUE LO UTILICE EL PROGRAMA DE
   LA AGENDA DE DIRECCIONES
30 REM ESCRIBE UN REGISTRO FICTICIO QUE
   PUEDA
40 REM SER UTILIZADO POR *LEARCH*
50 REM
60 REM
70 OPEN "O", #1, "DAT.AGCO"
80 PRINT #1,"@VACIO"
90 CLOSE #1
100 END
```

Como ya hemos mencionado antes en nuestro curso de programación BASIC, los detalles relativos a la lectura y escritura de archivos difieren considerablemente de una versión de BASIC a otra, pero el principio es casi siempre el mismo. Primero se debe declarar abierto (OPEN) el archivo antes de que se lo pueda utilizar tanto para entrada como para salida. Luego, la dirección del flujo de datos se declara IN o OUT. A continuación hay que asignarle al archivo un número de "canal". Esto permite tener abierto y en uso más de un archivo a la vez (por el momento, sin embargo, nosotros emplearemos un solo archivo). Por último, se debe declarar el nombre del archivo que deseamos utilizar.

La línea 70 del programa está en BASIC Microsoft y, en principio, es similar a las sentencias OPEN que emplean la mayoría de las versiones de BASIC (el BASIC del BBC es algo distinto; véase p. 319). Por supuesto, OPEN declara que se va a abrir (OPEN) un archivo y "O" que los datos van a salir. #1 es el número que le estamos asignando al archivo para esta operación; de ser necesario, más adelante se podría utilizar un número de archivo diferente. "DAT.AGCO" es el nombre que damos al archivo.

La línea 80 simplemente escribe un único registro en el archivo. La sintaxis de la escritura de datos en un archivo por lo general (en la mayoría de las versiones de BASIC) es exactamente la misma que la sintaxis utilizada para imprimir (PRINT), excepto que la sentencia PRINT debe ir seguida por el número del archivo (#1, en este caso).

La línea 90 cierra (CLOSE) el archivo. Los archivos se pueden dejar abiertos todo el tiempo que necesite el programa; pero los archivos "abiertos" son vulnerables y se deben cerrar lo antes posible para proteger los datos que contienen.

Existe una cierta confusión en cuanto a las formas de utilizar los términos "registro" y "archivo" al aplicarlos a los ordenadores, y esta confusión aumenta cuando hablamos de bases de datos, por un lado, y de archivos de datos, por otro. En una base de datos, el archivo es un grupo completo de información que guarda una relación entre sí. Utilizando la analogía del mueble archivador de una oficina, el archivo podría ser un cajón con el rótulo PERSONAL. Este archivo podría comprender un registro (una ficha dentro de una carpeta) para cada persona de la empresa. Cada registro (ficha) contendría una cantidad de apartados, idénticos para cada registro, con información como NOMBRE, SEXO, EDAD, SUELDO, AÑOS DE SERVICIO, etc.

Un archivo secuencial en disco o en cinta de cassette no se preocupa de cómo el programa utiliza u organiza la información contenida en él. Los archivos de datos tan sólo contienen una serie de ítems de datos y cada uno de ellos constituye un registro. Un solo registro de un archivo de datos, por lo

tanto, normalmente no correspondería a un registro en el sentido en que este término se aplica en la base de datos.

Corresponde al programa leer los registros del archivo de datos y asignarlos a matrices o a variables. Estas matrices y variables necesitan organizarse para conformar un registro "conceptual" que contenga un conjunto limitado de información relacionada.

No existe una relación de punto por punto entre los registros de un archivo de datos y los registros que componen una base de datos.

Una vez se ha ejecutado el programa de preparación, se supone que no se lo volverá a necesitar. De hecho, si se lo volviera a ejecutar, destruiría todos los datos "legítimos" a los que el usuario pudiera haber dado entrada en la base de datos de la agenda de direcciones. Al analizar el programa *LEARCH* modificado veremos por qué sucedería esto.

Cuando se ejecuta el programa, éste no "sabe" si en el archivo de datos hay información legítima o no. Lo primero que hace *LEARCH* es abrir (OPEN) el archivo "DAT.AGCO" y leer el primer registro (o ítem de datos). A éste no se le atribuye un elemento de una matriz, como usted podría imaginar, sino una variable alfanumérica especial que hemos denominado TEST$. Antes de que se lea ningún otro registro, se verifica TEST$ para comprobar si contiene la serie @VACIO. Si TEST$ contiene, en efecto, @VACIO, el programa sabe que en el archivo no hay ningún dato válido y que, por lo tanto, no tiene sentido intentar leer ningún otro dato y asignárselo a las matrices. Por consiguiente, el archivo se puede cerrar y puede continuar el resto del programa. Puesto que en el archivo no hay ningún dato válido, el usuario no puede hacer nada útil hasta que se haya dado entrada al menos a un registro y, por consiguiente, el valor de TEST$ también se puede utilizar para obligar al programa a ir hacia la subrutina *INCREG* a fin de que se agregue al menos un registro válido antes de que se pueda hacer cualquier otra cosa.

Si, por otra parte, el valor de TEST$ no es @VACIO, el programa puede suponer que en el archivo hay algún dato válido y puede comenzar a asignarles los datos a las matrices adecuadas. La subrutina *LEARCH* modificada continúa:

```
1400 REM SUBRUTINA *LEARCH*
1410 OPEN "I", #1,"DAT.AGCO"
1420 INPUT #1, TEST$
1430 IF TEST$ = "@VACIO" THEN GOTO 1530: REM
     CERRAR Y RETORNAR
1440 LET NOMCAM$(1) = TEST$
1450 INPUT #1,MODCAM$(1), CLLCAM$(1), CIUCAM$(1),
     PROCAM$(1), TELCAM$(1)
1460 INPUT #1,INDCAM$(1)
1470 LET TAMA = 2
1480 FOR L = 2 TO 50
1490 INPUT #1,NOMCAM$(L),MODCAM$(L),
     CLLCAM$(L),CIUCAM$(L),PROCAM$(L)
1500 INPUT #1,TELCAM$(L),INDCAM$(L)
1510 REM ESPACIO PARA LLAMAR A LA SUBRUTINA
     "TAMA"
1520 NEXT L
1530 CLOSE #1
1540 RETURN
```

La línea 1420 asigna un único registro del archivo DAT.AGCO a la variable TEST$. Luego la línea siguiente la verifica para comprobar si su valor es @VACIO. Si lo es, se utiliza una sentencia GOTO para saltar a la línea que cierra el archivo (la 1530) y después la subrutina vuelve (RETURN) al programa que la ha llamado. No se efectúan más intentos por leer los datos. Suponiendo que en el archivo no haya ningún dato válido, el control del programa pasará a *INICIL*, que luego llama a *ESBAND*. Todo cuanto esta rutina hace por el momento es establecer el valor de TAMA en 1 si TEST$ = @VACIO. El código para *ESBAND* se proporciona más abajo. Observe que se han escrito varias REM con el fin de dejar espacio disponible para establecer otras banderas en caso de que con posterioridad deseáramos hacerlo.

```
1600 REM *ESBAND*
1610 REM ESTABLECE BANDERAS DESPUES DE *LEARCH*
1620 REM
1630 REM
1640 IF TEST$ = "@VACIO" THEN LET TAMA = 0
1650 REM
1660 REM
1670 REM
1680 REM
1690 RETURN
```

Después *ESBAND* vuelve (RETURN) a *INICIL*, que a su vez retorna al programa principal. *PROPRI* llama luego a *PRESEN*, que visualiza el mensaje de presentación. En el caso de *PRESEN* no es necesario introducir ninguna modificación en la versión que hemos dado previamente para ella.

La siguiente rutina a la cual llama el programa principal es *ELECCN*. Una modificación muy pequeña de la subrutina *ELECCN* de la página 357 establecerá una forma de obligar al usuario a agregar un registro nuevo si el programa se estuviera ejecutando por primera vez.

```
3500 REM SUBRUTINA *ELECCN*
3510 REM
3520 IF TEST$ = "@VACIO" THEN GOSUB 3860
3530 IF TEST$ = "@VACIO" THEN RETURN
3540 REM 'IMMENU'
3550 PRINT CHR$(12)
3560 PRINT "SELECCIONE UNO DE LOS SIGUIENTES"
3570 PRINT
3580 PRINT
3590 PRINT
3600 PRINT "1. HALLAR REGISTRO (DE NOMBRE)"
3610 PRINT "2. HALLAR NOMERES (DE NOMBRE
     INCOMPLETO)"
3620 PRINT "3. HALLAR REGISTRO (DE CIUDAD)"
3630 PRINT "4. HALLAR REGISTRO (DE INICIAL)"
3640 PRINT "5. LISTAR TODOS LOS REGISTROS"
3650 PRINT "6. AGREGAR REGISTRO NUEVO"
3660 PRINT "7. MODIFICAR REGISTRO"
3670 PRINT "8. BORRAR REGISTRO"
3680 PRINT "9. SALIR Y GUARDAR"
3690 PRINT
3700 PRINT
3710 REM 'ASOPCN'
3720 REM
3730 LET L = 0
3740 LET I = 0
3750 FOR L = 0 TO 1
3760 PRINT "DE ENTRADA A OPCION (1 - 9)"
3770 FOR I = 1 TO 1
3780 LET A$ = INKEY$
3790 IF A$ = "" THEN I = 0
3800 NEXT I
3810 LET OPCN = VAL(A$)
3820 IF OPCN < 1 THEN L = 0 ELSE L = 1
3830 IF OPCN > 9 THEN L = 0
3840 NEXT L
3850 RETURN
```

Se han agregado dos líneas. La primera comprueba el contenido de TEST$. Esta variable aún contiene el valor que se le atribuye en la rutina *LEARCH*. Si éste es @VACIO sabemos que en el archivo no hay ningún dato válido y, por consiguiente, la única opción adecuada es INCREG, que es el número 6. Si se pasa la prueba, el control pasa a *PRIMERA*, una rutina que visualiza un mensaje apropiado y que

establece la variable OPCN en 6. Cuando la subrutina regresa a la línea 3530, se vuelve a comprobar el contenido de TEST$ (está destinada a pasar la prueba) y la subrutina regresa (RETURN) al programa principal saltándose el resto de la subrutina *ELECCN*, dado que es inadecuada.

Tal vez se pregunte por qué se comprueba TEST$ dos veces. Se hace para impedir que la subrutina retorne (RETURN) a un punto equivocado del programa. Sin la línea 3530, el programa seguiría adelante con el resto de *ELECCN*, presentando el menú de opciones incluso aunque éste no fuera necesario. También evita la utilización de sentencias GOTO, si bien IF TEST$ = "@ VACIO" THEN GOTO 3850 funcionaría igualmente. Las sentencias GOTO hacen que el programa resulte confuso y difícil de seguir (se suele decir que los programas que abusan de sentencias GOTO están en "código spaghetti").

Antes de proseguir analizando *PRIMERA*, volveremos a remitir a los lectores a *LEARCH* y a la sentencia GOTO de la línea 1430. Dado que hemos hecho un razonamiento coherente en contra de la utilización de GOTO, ¿por qué hemos empleado una aquí? Hubiera sido igualmente fácil cerrar (CLOSE) el archivo y retornar (RETURN) simplemente comparando el valor de TEST$ en dos líneas separadas. Aquí hemos preferido utilizar una GOTO para ilustrar uno de los pocos casos en los que su utilización resulta disculpable. Esto es, dentro de un segmento de programa muy breve e identificable, y cuando su función es obvia (y recalcada aún más mediante la observación REM). Las sentencias GOTO nunca se deben emplear para saltar fuera de un bucle (lo cual puede dejar el valor de las variables en un estado impredecible), ni para saltar fuera de una subrutina (lo cual confundiría a la instrucción RETURN) ni para saltar a regiones lejanas del programa.

La subrutina *PRIMERA* es simple y directa: se limpia la pantalla y se visualiza un mensaje que informa al usuario de que se ha de dar entrada a un registro. La línea 3870 establece OPCN en 6, de modo que cuando el control vuelve a pasar a *EJECUT*, la rutina *INCREG* se ejecutará automáticamente. Veamos la codificación para *PRIMERA*:

```
3860 REM SUBRUTINA *PRIMERA* (VISUALIZAR
     MENSAJE)
3870 LET OPCN = 6
3880 PRINT CHR$(12); REM LIMPIAR PANTALLA
3890 PRINT
3900 PRINT TAB(10);"NO HAY REGISTROS EN"
3910 PRINT TAB(7);"EL ARCHIVO. DESEA EMPEZAR"
3920 PRINT TAB(8);"POR AGREGAR UN REGISTRO"
3930 PRINT
3940 PRINT TAB(0);"(PULSE BARRA ESPACIADORA
     PARA CONTINUAR)"
3950 FOR B = 1 TO 1
3960 IF INKEY$<>" " THEN B = 0
3970 NEXT B
3980 PRINT CHR$(12); REM LIMPIAR PANTALLA
3990 RETURN
```

La subrutina *INCREG*, que aparece en la página 379, incorpora dos modificaciones pequeñas pero importantes respecto a la versión que habíamos ofrecido anteriormente. Una vez se ha dado entrada a los campos como elementos de las diversas matrices, la variable TAMA se incrementa y TEST$ se establece en una serie nula (véanse líneas 10090 y 10100). TAMA es una variable importante que se emplea en diversas partes del programa, con el objeto de que éste pueda saber sobre qué registros se está operando. TAMA se estableció originalmente en 0 como parte de la subrutina *CREMAT*. Posteriormente, en *ESBAND*, se establece en 1 si TEST$ = "@VACIO". Esto se hace para que cuando se ejecute *INCREG* por primera vez, las sentencias INPUT coloquen los datos en el primer elemento de cada matriz. En otras palabras, INPUT "DE ENTRADA AL NOMBRE";NOMCAM$(TAMA) equivale a INPUT "DE ENTRADA AL NOMBRE";NOMCAM$(1).

La línea 10090 incrementa TAMA, de modo que ahora se convierte en 2. Si se volviera a ejecutar *INCREG*, a los datos se les daría entrada en el segundo elemento de cada matriz. Por último, *INCREG* establece a TEST$ en " " en la línea 10100. Esto se hace porque ahora se ha dado entrada a un registro (aunque aún no ha sido almacenado en el archivo de datos de la cinta o disco). Si se volviera a ejecutar *ELECCN*, como se debe hacer para guardar los datos y hacer salir el programa, no querríamos vernos obligados a volver a agregar un registro nuevo. Si no se restaurara TEST$, el programa se metería en un bucle infinito, y la única forma de salir de él sería restaurar o desconectar el ordenador, con lo cual se perderían todos los datos.

Al establecer TEST$ en una serie nula, las condiciones para *ELECCN* en las líneas 3520 y 3530 fracasarán y permitirán que se visualice el menú de opciones. Lo que suceda entonces con TAMA dependerá de qué rutina se ejecute. Hasta ahora sólo hemos asegurado que TAMA = 1 si en el archivo no hay datos válidos, y que éste se incrementa en 1 cada vez que se agrega un registro. Pero ¿qué sucedería si en el archivo hubiera ya registros válidos? Para saberlo, volvamos a analizar *LEARCH*.

La línea 1420 atribuye el primer ítem de datos a TEST$. Si éste no es @VACIO, se da por sentado que es un ítem válido. Los registros del archivo siempre están en el mismo orden, es decir: NOMCAM, MODCAM, CLLCAM, CIUCAM, PROCAM, TELCAM, INDCAM, NOMCAM, MODCAM, etc. Si el primer registro que se lee es un dato válido, debe pertenecer al primer elemento de la matriz NOMCAM$, de manera que la línea 1440 transfiere este dato de TEST$ a NOMCAM$(1). Las dos líneas siguientes completan los primeros elementos de las otras cinco matrices. Ahora sabemos que tenemos un registro (base de datos) completo, y TAMA se establece en 2.

Después, un bucle de 2 a 50 atribuye los registros a las seis matrices, incrementando el índice L a cada ciclo. Ya hemos tomado la decisión de limitar nuestro programa a tratar con archivos de 50 nombres y direcciones, y las sentencias DIM de la subrutina *CREMAT* destinaban espacio para ello. No obstante, cuando comience a utilizar el programa por primera vez, es poco probable que tenga un archivo completo de 50 entradas, de manera que necesitaremos incluir en el programa una rutina que pueda detectar cuándo se produce este caso, establecer la variable TAMA y abortar el bucle de lectura.

Por consiguiente, hemos incluido la línea 1510 para proporcionar una llamada a la subrutina 'TAMA', que desarrollaremos más avanzado el curso. Hay tres formas posibles de abordar este problema. Primero, al escribir los datos en cinta podríamos hacer que el primer registro a escribir fuera la variable TAMA. La subrutina *LEARCH* se podría entonces modificar para que primero leyera TAMA y luego estableciera un bucle de la forma FOR L = 1 TO TAMA para leer los registros. El segundo método, que es preferible (dado que no se contra-

dice con nuestra condición previa para @VACIO de la línea 1430), consiste en establecer un procedimiento a ejecutar después de que se hayan escrito todos los registros, en el cual se pueda escribir al final una bandera especial (tal vez de la forma @PIN). Luego se podría insertar una condición en *LEARCH* para abortar el bucle cuando se encontrara con @PIN.

El tercer método consiste en hacer uso de la función EOF (*End Of File*: fin del archivo) que ofrecen algunos ordenadores personales, que en realidad es una versión automatizada del segundo método. Estas máquinas disponen de una bandera EOF, que normalmente está establecida en 0, es decir, FALSO, pero que asume otro valor (por lo general, 1 para representar VERDADERO) cuando se ha llegado al final del archivo. Algunas versiones de BASIC permiten probar la bandera EOF como una variable en BASIC; en este caso, una construcción de forma:

```
WHILE NOT EOF(N) (N es el número del archivo)
  DO
    INPUT #N (los datos a leer)
ENDWHILE
```

resolverá el problema. En otras máquinas, la bandera EOF se representa como un único bit al que se debe acceder utilizando la sentencia PEEK. Para averiguar si su ordenador dispone de la función EOF, deberá consultar el manual de instrucciones. Dado que esta función difiere mucho de una máquina a otra, no la utilizaremos en nuestro programa. Pero a modo de ejercicio, quizá a los lectores les interese intentar modificar la subrutina *LEARCH* mediante estos tres métodos posibles de tratar con archivos de menos de 50 entradas.

Por lo general, siempre es más fácil escribir programas que traten con archivos de longitud fija, pero abordar el problema de los archivos de "longitud dinámica" en esta etapa temprana nos permitirá modificar más tarde el programa para hacer frente a archivos de más de 50 entradas.

```
4000 REM SUBRUTINA *EJECUT*
4010 REM
4019 IF OPCN = 6 THEN GOSUB 10000: REM VEASE NOTA
     AL PIE DE PAGINA
4020 REM NORMALMENTE 'ON OPCN GOSUB ETC' --VEASE
     NOTA PIE PAGINA
4030 REM
4040 REM  1 ES *ENCREG*
4050 REM  2 ES *ENCNOM*
4060 REM  3 ES *ENCIUD*
4070 REM  4 ES *ENCINI*
4080 REM  5 ES *LISREG*
4090 REM  6 ES *INCREG*
4100 REM  7 ES *MODREG*
4110 REM  8 ES *BORREG*
4120 REM  9 ES *SAPROG*
4130 REM
4140 RETURN
```

La rutina *EJECUT* normalmente no tendría línea 4019 (y por ello el número de línea impar), y la línea 4020 normalmente sería o bien

```
ON OPCN GOSUB número, número, número etc
```

o una serie de:

```
IF OPCN = 1 THEN GOSUB número
IF OPCN = 2 THEN GOSUB número etc
```

Se incluye la línea 4019 para que el programa funcione incluso aunque no se hayan codificado aún las otras subrutinas *EJECUT*

```
10 REM 'PROPRI'
20 REM *INICIL*
30 GOSUB 1000
40 REM *PRESEN*
50 GOSUB 3000
60 REM *ELECCN*
70 GOSUB 3500
80 REM *EJECUT*
90 GOSUB 4000
100 END

1000 REM SUBRUTINA *INICIL*
1010 GOSUB 1100: REM SUBRUTINA *CREMAT* (CREAR MATRICES)
1020 GOSUB 1400: REM SUBRUTINA *LEARCH* (LEER ARCHIVOS)
1030 GOSUB 1600: REM SUBRUTINA *ESBAND* (ESTABLECER BANDERAS)
1040 REM
1050 REM
1060 REM
1070 REM
1080 REM
1090 RETURN

1100 REM SUBRUTINA *CREMAT* (CREAR MATRICES)
1110 DIM NOMCAM$(50)
1120 DIM MODCAM$(50)
1130 DIM CIUCAM$(50)
1140 DIM PROCAM$(50)
1150 DIM TELCAM$(50)
1160 DIM INDCAM$(50)
1170 REM
1180 REM
1190 REM
1200 REM
1210 LET TAMA = 0
1220 LET RMOD = 0
1230 LET SVED = 0
1240 LET CURS = 0
1250 REM
1260 REM
1270 REM
1280 REM
1290 REM
1300 RETURN

10000 REM SUBRUTINA *INCREG*
10010 PRINT CHR$(12): REM LIMPIAR PANTALLA
10020 INPUT "DE ENTRADA AL NOMBRE";NOMCAM$(TAMA)
10030 INPUT "DE ENTRADA A LA CALLE";CLLCAM$(TAMA)
10040 INPUT "DE ENTRADA A LA CIUDAD";CIUCAM$(TAMA)
10050 INPUT "DE ENTRADA A LA PROVINCIA";PROCAM$(TAMA)
10060 INPUT "DE ENTRADA AL NUMERO DE TELEFONO";TELCAM$(TAMA)
10070 LET RMOD=1: REM ESTABLECIDA BANDERA 'REGISTRO MODIFICADO'
10080 LET INDCAM$(TAMA) = STR$(TAMA)
10090 LET TAMA = TAMA +1
10100 LET TEST$ = ""
10110 REM INSERTAR AQUI LLAMADA A *MODNOM*
10120 REM
10130 REM
10140 REM
10150 RETURN
```

Complementos al BASIC

SPECTRUM

Dado que el Spectrum posee la facilidad de guardar o cargar matrices enteras utilizando la orden SAVE-DATA, como explicamos en la p. 318, la subrutina *LEARCH* será totalmente diferente: se leerán cada una de las matrices (NOMCAM$, CLLCAM$, etc.) sucesivamente. En el próximo capítulo, cuando comencemos a escribir los datos, veremos una versión completa de las subrutinas correspondientes a esta máquina. Mientras tanto, y a modo de ejercicio, los usuarios de ordenadores Spectrum pueden abordar el problema de cómo crear el archivo ficticio que contenga @VACIO, así como determinar cuántas entradas válidas hay en la matriz cuando se lee el archivo

INKEY$

Véase "Complementos al BASIC", p. 175.

OPEN CLOSE

Véase "Complementos al BASIC", p. 319.

Tony Lodge

Por partida doble

Herman Hollerith y James Powers desarrollaron máquinas de tabular. Su rivalidad dominó el mundo de la informática durante seis décadas

James Powers
Las máquinas de Powers eran puramente mecánicas y especializadas en una única aplicación. Así y todo, fue un duro competidor de Hollerith

Cortesía de IBM

Herman Hollerith
Hollerith inventó el lector electromecánico de tarjetas, cuyo desarrollo desembocaría luego en la tabuladora

Las máquinas que Herman Hollerith (véase p. 240) inventara para procesar los resultados del censo de 1890 en Estados Unidos evolucionaron hacia una gama de equipos para el procesamiento de datos con fines generales conocidos como "tabuladoras". Hasta la introducción de los primeros ordenadores comerciales, en la década de los cincuenta, las tabuladoras eran esenciales para el crecimiento de la industria y los negocios. Por ejemplo, durante los años treinta unos importantes almacenes de Pittsburgh experimentaron un sistema de cuentas de clientes en el cual se conectaron 250 terminales distribuidas por el establecimiento a un banco central de tabuladoras mediante las líneas telefónicas. Las mercancías se vendían con unos rótulos perforados y la información se enviaba automáticamente a las tabuladoras, que registraban luego la venta y preparaban una factura para el cliente. Después de comprobarse el margen de crédito del cliente, se enviaba la autorización para la venta a la terminal a través de una máquina de escribir "en línea".

De hecho, la competencia empresarial proporcionó el estímulo inicial para el desarrollo de las tabuladoras. El monopolio que ejercía Hollerith sobre la provisión de equipos para censos se rompió en 1910, cuando el Census Bureau (Departamento de Censos) instó a James Powers a crear máquinas alternativas. Ofreció un sistema de tabuladoras mecánicas que no violaban las patentes de los dispositivos electromecánicos de Hollerith.

La rivalidad entre ellos y las empresas que ambos crearon fomentó el crecimiento de las máquinas para procesamiento de datos.

En 1902 Hollerith diseñó un conmutador de clavijas (parecido al cuadro conmutador telefónico) que podía seleccionar las columnas de la ficha perforada que se habían de agregar y a las que luego había que dar salida. De esta forma, la máquina de Hollerith poseía una capacidad de programación de la que carecían las máquinas de su competidor; Powers siempre fabricó máquinas para aplicaciones específicas; en 1924 patentó una forma de representar datos alfanuméricos en fichas perforadas mediante la utilización de un único agujero en cada columna para un número, y una combinación de agujeros para representar una letra. Hollerith respondió rápidamente con su propio sistema: la nueva tarjeta estándar de 80 columnas. Cada columna de esta tarjeta contenía 12 filas de agujeros que se "leían" mediante escobillas de alambre que completaban un circuito eléctrico con un contacto metálico debajo de la tarjeta.

Las primeras tabuladoras sólo podían contar o acumular totales, pero posteriormente se proporcionaron funciones matemáticas más avanzadas para la manipulación de datos. A diferencia de los ordenadores, que fueron inventados por los científicos con fines matemáticos, la tabuladora se creó para que fuera un procesador de información. Ello estimuló enseguida a elaborar aplicaciones para las nuevas máquinas. Se adaptaron tabuladoras especiales para utilizarlas en tablas de cálculo, en análisis de ondas y en astronomía: en 1930 las tabuladoras identificaron el planeta Plutón. Finalmente, las tabuladoras alcanzaron el nivel de sofisticación suficiente para tratar de forma interactiva grandes cantidades de datos; IBM patentó una que podía llevar los registros de las transacciones de 10 000 cuentas bancarias. Pero su mayor éxito fue cotejar datos a una escala hasta entonces desconocida.

Las máquinas tabuladoras

En su período de apogeo, a comienzo de la década de los cincuenta, la tabuladora estaba compuesta por ocho unidades separadas. Los datos se introducían en cada tarjeta mediante una "perforadora de tarjetas", que podía procesar 200 tarjetas por hora. Un "verificador" separado comprobaba la exactitud del operador de la perforadora y, cuando las tarjetas se gastaban, una "perforadora de reproducción" creaba copias nuevas. Para dar una referencia sencilla, un "intérprete" imprimía una explicación de los datos en la parte superior de cada columna. La "tabuladora" propiamente dicha acumulaba en las columnas los totales de los datos y transfería la información a una velocidad de 9 000 tarjetas por hora. Esta tabuladora a menudo estaba conectada a una "perforadora de multiplicación", que proporcionaba funciones matemáticas más sofisticadas. El "cotejador" podía comparar los datos de dos grupos de tarjetas o intercalar dos grupos entre sí. Por último, la "clasificadora" podía clasificar un grupo de tarjetas en 13 grupos: uno para cada uno de los 12 agujeros y otro para una columna en blanco. La operación de la tabuladora se podía modificar mediante códigos de control (en las posiciones 11 y 12) y las tarjetas de control eran de colores brillantes para que pudieran ser reconocidas. Al encontrarse con una tarjeta de control, la tabuladora empezaba una operación nueva, tal como contar un apartado diferente. En el caso de un censo, un ejemplo de apartado sería los datos relativos a una casa, a una calle o a una ciudad. A cada cambio de apartado la tabuladora imprimía un subtotal (en nuestro ejemplo, éste proporcionaría el número de habitantes de una casa, de una calle o de una ciudad)

BBC Hulton Picture Library

Informática y ficción

Superman III
El fraude mediante ordenador es el tema central del tercer filme de Superman. Richard Pryor interpreta el papel de un malvado que se enriquece hurtando medio centavo en cada transacción a través del ordenador de un banco. Esta parte del guión se basa en varios casos reales de fraude. La película termina con la destrucción del mayor ordenador del mundo, que había sido construido únicamente con fines criminales

Los ordenadores se han usado a menudo en temas de ciencia-ficción, y algún escritor anticipó tecnologías hoy usuales

Muchos adelantos científicos y técnicos han sido imaginados por escritores de ficción y realizadores cinematográficos mucho antes de que fueran realmente factibles. Arthur C. Clarke, autor de *2001: una odisea del espacio (2001: a space odyssey)*, fue el primero en exponer la idea de la utilización de satélites geoestacionarios, en un artículo publicado en la revista *Wireless World*, a principios de la década de los cincuenta, es decir, con una antelación de casi veinte años sobre su puesta en funcionamiento. De una forma similar, en el relato de Robert Heinlein *Waldo*, se describía la manipulación de objetos por control remoto bastante antes de que se empezaran a emplear manos artificiales de robot. De hecho, numerosos inventores y científicos se han inspirado en las ideas surgidas de las mentes creadoras de escritores de obras de ficción y realizadores cinematográficos.

Sin embargo, frecuentemente, los ordenadores descritos en las obras de ciencia-ficción tienen poco que ver con la realidad. En la película futurista *Rollerball*, por ejemplo, se podía ver un ordenador que constaba de entrada con reconocimiento de voz y salida hablada, y cuya forma era semejante a un depósito para líquidos de estructura cúbica. En la realidad, por supuesto, el papel que desempeñan estas máquinas no suele ser muy espectacular ni interesante, aunque también en algunos filmes su única misión sea puramente decorativa, constituyendo un elemento más del mobiliario. No cabe la menor duda de que en los años sesenta y setenta, muchas películas en las que aparecían ordenadores con un diseño similar al que tienen los actuales han ayudado a educar al público, al mostrarle cómo eran en realidad estos nuevos, misteriosos, casi míticos «ordenadores».

La idea del ordenador empezó a tomar cuerpo en las novelas de ciencia-ficción poco después de que Charles Babbage (véase p. 220) comenzara su trabajo precursor sobre un ingenio analítico, a mediados del siglo XIX. En 1879, Edward Page Mitchell escribió un relato llamado *The ablest man in the world*, en el que se describe la implantación de una máquina calculadora en el cerebro de un deficiente mental, que lo convierte en un genio. Las ideas de Mitchell iban muy por delante de los avances científicos reales. En primer lugar, se centró en la idea de la miniaturización: la máquina computerizada es al mismo tiempo lo suficientemente pequeña como para ser colocada en el interior del cráneo del protagonista y lo suficientemente potente como para dotar a éste de una inteligencia superior. En segundo lugar, Mitchell prefiguró la idea de interconectar un ordenador con el cuerpo humano. Hoy en día, más de un siglo después de que fuera escrito este relato, se empiezan a perfeccionar las técnicas para conectar dispositivos electromecánicos de fácil control en el sistema nervioso central.

Por lo general, sólo unos cuantos escritores poseen conocimientos amplios de la arquitectura de los ordenadores, aunque algunos de ellos sean expertos ingenieros y muchos los utilicen (como procesadores de textos) en su trabajo. Del mismo modo, un escritor puede describir de manera convincente un viaje intergaláctico, sin necesidad de ser un astrofísico altamente cualificado o un experto en la tecnología de los cohetes. Asimismo, no existe ninguna razón por la cual no sea posible especular sobre las posibles características de las futuras generaciones de ordenadores aunque no se posea un conocimiento profundo de las máquinas que se emplean en la actualidad.

B.F.I. Stills

Ordenador HAL
El ordenador Heurístico y ALgorítmico de la obra de Arthur Clarke *2001: una odisea del espacio* es un buen ejemplo de la omnipotente máquina ordenadora que aparece con frecuencia en los filmes de ciencia-ficción. En este caso, HAL es el responsable de los detalles de los objetivos de la misión mientras que los miembros humanos de la tripulación desarrollan una misión secundaria, lo cual conduce a la máquina a creer que puede prescindir de éstos. Se supone que Clark eligió las iniciales HAL por su proximidad en el orden alfabético con IBM

Sin embargo, estas teorías han conducido al concepto de un ordenador de ciencia-ficción estándar que posee, en comparación con las posibilidades reales de los ordenadores actuales, una gama de propiedades que son imposibles de lograr. Para empezar, este ordenador-tipo almacena en su memoria todos los datos e información existentes en el mundo y cualquier idea que haya imaginado la mente humana, y es capaz de recuperar el dato necesario instantáneamente, mediante un procedimiento similar al funcionamiento del cerebro. El ordenador HAL de *2001: una odisea del espacio* es una de estas máquinas superinteligentes.

Otra de las características con que dotan los escritores de ciencia-ficción a sus ordenadores es la omnipresencia, si bien, al mismo tiempo, se crea la sensación de que muy pocas personas tienen acceso a ellos. Dos de los requisitos básicos del superordenador son la capacidad de hablar (pero sin hacer ninguna mención a que se puede tratar simplemente de un producto sintético de ensamblamiento de fonemas), y el reconocimiento de la voz (que indefectiblemente tiene en cuenta las peculiaridades del habla de cada persona). Pero también el reconocimiento visual de los objetos y la capacidad de sintetizar alimentos (quizá a partir de sus componentes elementales básicos) constituyen a su vez otras propiedades de gran utilidad.

El superordenador que hemos descrito en términos generales, en la mayoría de los casos está dotado asimismo de atributos humanos, y esta caracterización le hace aparecer como una especie de ser superior. Sin embargo, algunas veces el papel que desempeña puede ser decididamente malévolo. En el filme *Dark star*, por ejemplo, una bomba controlada por ordenador adquiere las características perturbadas de un asesino psicópata. Cuando es representado de esta forma, el superordenador, ciertamente, forma parte del reino de la fantasía, pero, al mismo tiempo, en los componentes de la informática actual es posible entrever las características básicas de esas propiedades que hoy son sólo atributo de la ficción.

En estos momentos ya existen máquinas con una gran capacidad de memoria y que requieren un tiempo de acceso muy pequeño. A principios de los ochenta se crearon memorias que alcanzaban el valor del gigabyte (mil millones de bytes), y las máquinas comerciales más rápidas poseían una velocidad de procesamiento superior a los diez millones de instrucciones por segundo. Volviendo otra vez al campo de los ordenadores que hablan, se está muy cerca de alcanzar el nivel de perfección mostrado en las películas. La calidad que pueda poseer esta característica depende sencillamente del espacio de memoria disponible, de la velocidad de procesamiento y del tiempo de programación. En cambio, el reconocimiento de voz es más difícil de lograr, debido a la amplia gama de formas del habla humana. Dos personas pueden hablar el mismo idioma, pero para el ordenador es como si se expresaran en lenguas completamente diferentes.

El reconocimiento visual de objetos se encuentra también en una fase muy elemental de desarrollo, pero la tecnología ligada a esta esfera está avanzando rápidamente. Cuando analizamos los robots industriales (véase p. 281), nos dimos cuenta de que se ha producido un gran adelanto en el reconocimiento de objetos mediante cámaras de televisión con un dispositivo de carga en paralelo, y vimos que el robot era capaz de elegir un objeto determinado de entre varios dispuestos aleatoriamente. Un reconocimiento visual que sea significativo depende del tamaño del vocabulario óptico, el cual, a su vez, está en función de la capacidad de memoria y de la potencia de procesamiento. En lo referente a la síntesis de alimentos, quizá no sea posible hacer que un plato tenga la apariencia de filete con patatas o pescado frito, pero, sin lugar a dudas, existe la posibilidad de hacer que tenga el aroma y el sabor de éstos, aun cuando los ordenadores no puedan, todavía, crearlos a partir de sus elementos.

No todos los autores llegan al extremo de atribuir poderes extraordinarios a sus máquinas de ficción. John Brunner, por ejemplo, en su novela de ciencia-ficción *Situación en Zanzíbar (Stand on Zanzibar)*, publicada en 1969, describe el mundo tal como podría ser en el año 2010, con unos problemas de superpoblación y de falta de alimentos que habrían alcanzado niveles insostenibles. El ordenador que describe, al que llama «Shalmaneser»,

tiene, por descontado, una capacidad de memoria y una velocidad de procesamiento considerables (puesto que está conectado a todos los aparatos de televisión de la Tierra), pero su lenguaje es muy similar a los que se utilizan en la actualidad:

PROGRAMA RECHAZADO
Q razón del rechazo
ANOMALIAS EN DATOS BASICOS
Q definir Q especificar
INFORMACION NO ACEPTABLE EN LAS SIGUIENTES CATEGORIAS HISTORIA COMERCIO INTERACCION SOCIAL CULTURA
Q aceptar datos como están
PREGUNTA SIN SENTIDO E IMPRACTICABLE

Evidentemente, Brunner llegó hasta tal punto en la utilización de un lenguaje cercano al normal que un analista lo podría tomar como una respuesta de un sistema realmente existente. Sus otras predicciones son asimismo convincentes, y no es sorprendente que la novela ganara los premios más importantes del género de la ciencia-ficción.

En filmes y novelas más recientes, los ordenadores se han convertido en algo más que una parte del mobiliario o incluso que personajes secundarios, y en la actualidad frecuentemente constituyen la trama central del argumento. Un buen ejemplo de ello lo constituye *TRON*, producción de la firma Walt Disney. Anteriormente ya hemos hablado de este extraordinario largometraje (su nombre deriva de un sistema de realización *TRace ON*), presentándolo como representativo del sistema de realización de dibujos animados mediante ayuda de ordenadores (véase p. 181). La acción se desarrolla en el mundo real y en el *interior* de un ordenador. El mundo exterior está formado por personajes tales como ingenieros en software, programadores de sistemas y otras personas relacionadas con el mundo de la informática. Pero en el interior de la máquina, las secciones independientes del programa y del sistema son los verdaderos protagonistas, y la arquitectura de la máquina es el escenario en el cual tiene lugar la acción.

También existen obras de ficción en las que, en realidad, no se hace mención de los ordenadores, pero en las cuales el lector no tiene la menor duda de que sin esta maquinaria informática extremadamente poderosa, la situación que se describe nunca podría existir. Ejemplos notables entre estas últimas son *1984*, de George Orwell, y *Un mundo feliz (Brave new world)*, de Aldoux Huxley. Ambas novelas se desarrollan en una época futura con respecto al tiempo en que fueron escritas, y en un mundo totalmente dominado por una pequeña camarilla que reprime al resto de la población. Quizá deberíamos fijarnos en estos dos libros para formarnos una idea de las consecuencias que podría acarrear una mala utilización del poder de la informática.

Es totalmente imposible agotar el tema en este análisis del ordenador como argumento de ciencia-ficción, pero es necesario mencionar algunas novelas que describen ideas especialmente ingeniosas. Por ejemplo, la obra de John Barth *Giles Goat-Boy*. Esta voluminosa novela (812 páginas en su edición inglesa en rústica) se centra en la premisa de que el libro es obra de un superordenador llamado WESCAC, y relata un incidente que le sucedió a su "autor".

Por último, deberíamos recordar también que no es únicamente en la esfera de la ficción donde se encuentran creaciones acerca de los ordenadores. De los miles de libros que se han escrito centrados en el tema específico de los ordenadores y de la informática, hay uno que sobresale del resto simplemente por su calidad narrativa. La obra de Tracy Kidder *Soul of a new machine* es la historia del desarrollo del Data General's Eagle, un miniordenador de 32 bits. Si bien el tema trata, aparentemente, de los ingenieros y directores que participaron en el proyecto, no cabe duda de que el protagonista es el propio ordenador.

Juegos de guerra
Inconscientemente, un adolescente que intenta poner en comunicación su ordenador con el de un amigo, a través de la red telefónica, interfiere el ordenador principal de defensa de la OTAN. Creyendo que lo que ve en su pantalla es un juego, empieza a jugar, hasta descubrir que ha desencadenado la tercera guerra mundial...

Autor original

Es posible escribir programas que generen por sí mismos otros programas o que corrijan errores humanos en la codificación

"Si los ordenadores son tan inteligentes, ¿por qué necesitan ser programados por los seres humanos?" Las personas con experiencia en informática tenderán a restar importancia a una pregunta de este tipo, hecha por un principiante escéptico, pero no es tan banal como pudiera parecer a simple vista. En la actualidad se realizan numerosas investigaciones con la finalidad de escribir programas que puedan generar otros programas, y sistemas operativos capaces de corregir errores en el código escrito por el operador.

Téngase en cuenta, por ejemplo, el mensaje ¿ERROR SINTAXIS?, con que se encuentran frecuentemente los que utilizan los ordenadores personales. Esto puede resultar bastante desconcertante porque el mensaje suministra muy poca información. Un compilador de un ordenador de mayor capacidad daría normalmente muchos más datos, como podría ser la naturaleza del error que se ha hallado. Por ejemplo, el mensaje indicando el error podría tener la siguiente configuración:

```
1090 LET A = (C*2+F$)*((FG-C)*TH+1))
                ↑                ↑
ERRORES:1) MAL EMPAREJADO — VARIABLE
           ALFANUMERICA F$ NO PERMITIDA
        2) ULTIMO CIERRE PARENTESIS NO
           ESPERADO
```

No existe ninguna razón fundamental por la cual estas técnicas no puedan utilizarse en un intérprete de un ordenador personal, ya que el coste del ROM suplementario que sería necesario para poder almacenar las rutinas correspondientes sería mínimo. No obstante, muy pocos lo emplean, así como tampoco utilizan procedimientos elementales de control de error: la mayoría de ellos no verifican la sintaxis del código cuando éste es introducido. Sin embargo, con frecuencia existe la posibilidad de comprar chips de ROM suplementarios o cartuchos de software que amplíen la gama de órdenes disponibles en BASIC, en particular aquellas que tienen relación con el desarrollo y la eliminación de errores en los programas. Estas órdenes BASIC comprenden:

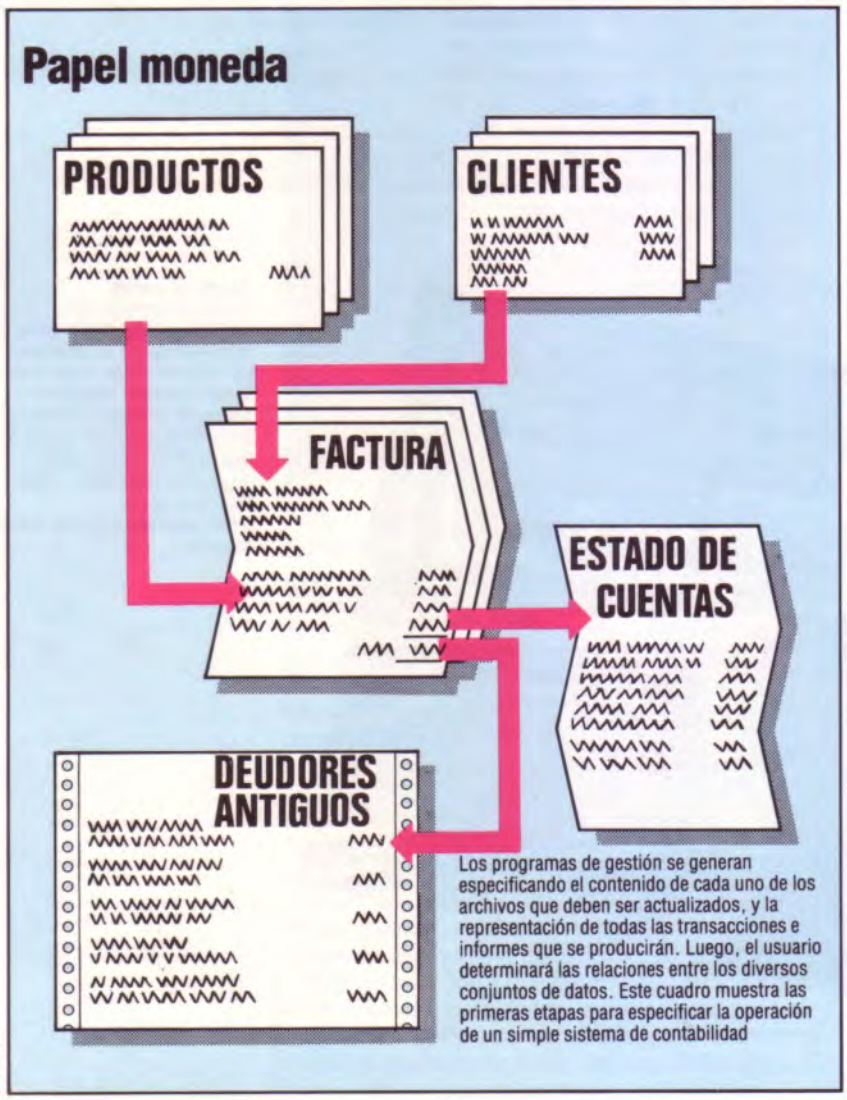
HELP: imprime la línea del programa y subraya la posición exacta del carácter en donde termina la realización del programa. Esto indicará normalmente, aunque no siempre, el origen del error de sintaxis.

DUMP: imprime una lista de todos los nombres de variables y de sus contenidos utilizados en ese momento por el programa. Esto es útil para deducir hasta qué punto se ha producido el procesamiento del programa antes de ocurrir el error.

TRACE: visualiza (en una ventana en la esquina de la pantalla) el número (o números) de línea que se está procesando mientras el programa está en funcionamiento. Esto ayuda al usuario a trazar el organigrama del programa, y asegura, entre otras cosas, que las subrutinas están siendo ejecutadas en el orden deseado.

La escritura de programas que permitan a un ordenador corregir errores humanos de codificación no es, en general, una tarea sencilla. Sin embargo, en el caso de algunos errores específicos sí resulta bastante simple. Por ejemplo, sabemos que todas las líneas del programa tienen que empezar con una contraseña en BASIC (si bien algunas máquinas permiten omitir la palabra LET). Por tanto, si una línea empieza con PRUNT o PRONT, será fácil deducir que debería decir PRINT.

Pero para otro tipo de errores más complejos que estos procedimientos básicos descritos, la corrección automática presenta un grado de dificultad muy superior. En el ejemplo que hemos dado ante-



Los programas de gestión se generan especificando el contenido de cada uno de los archivos que deben ser actualizados, y la representación de todas las transacciones e informes que se producirán. Luego, el usuario determinará las relaciones entre los diversos conjuntos de datos. Este cuadro muestra las primeras etapas para especificar la operación de un simple sistema de contabilidad

Liz Dixon

riormente, ¿es F$ un error de impresión y debería decir F o FS o F4? ¿O algo totalmente distinto? Si tuviera que enseñar el listado a otro programador experimentado, éste debería ser capaz de identificar las faltas y efectuar las correcciones. Para tomar sus decisiones, tendría que basarse en dos criterios: el contexto en el que ha aparecido la línea del programa, y su propia experiencia.

Por extraño que pueda parecer, esta técnica ha sido aplicada con mayor frecuencia para corregir el texto que para la verificación del código del programa. Un paquete para comprobar la escritura correcta de las palabras, por ejemplo, repasará todo el texto e indicará los términos que no se corresponden con las entradas de su diccionario, el cual puede constar de unas 50 000 voces, registradas en un disco. La mayoría de estos paquetes es capaz de aprender nuevas palabras (tales como nombres propios o de compañías) y de añadirlas a sus diccionarios. Algunos de ellos, de características más complejas, pueden incluso sugerir la forma correcta de escribir una palabra en el caso de que se detecte un pequeño error de escritura. También se han desarrollado procesadores de textos que tienen la posibilidad de aplicar estos mismos procesos a la corrección ortográfica y de estilo, suministrando indicaciones tales como puntuación incorrecta, repetición de una misma palabra dentro de un párrafo, metáforas confusas y adjetivos inadecuados.

Sin embargo, la mayor parte de las investigaciones se han centrado en el desarrollo de sistemas capaces de *crear* programas, en vez de *corregir* los existentes. En 1981 se anunció la salida al mercado de un producto de software que hizo estallar una de las batallas más enconadas jamás libradas en la industria del microordenador. Con un gran sentido de la oportunidad, recibió el nombre de "El último", y fue presentado con la pretensión de ser un programa que podía escribir cualquier otro que se deseara; por lo tanto, se constituía en el último programa que se necesitaría comprar. Esto resultó ser un reclamo publicitario que no se ajustaba a la realidad, aunque "El último" era una ayuda muy valiosa para el desarrollo de cierto tipo de programas, principalmente aplicaciones comerciales. En la actualidad existen en el mercado algunos de estos productos, de los cuales sólo una mínima parte está destinada a la gama de los ordenadores personales. A estos productos se les designa colectivamente con el nombre de *generadores de programas*.

Veamos ahora en qué consiste el concepto básico de un programa que sea capaz de escribir otros. Consideremos este ejemplo trivial:

```
10 PRINT "¿QUE DEBE VISUALIZAR EL PROGRAMA
   EN LA PANTALLA?"
20 INPUT A$
30 PRINT "EL PROGRAMA ES:"
40 PRINT "10 PRINT";CHR$(34);A$;CHR$(34)
```

Si a esta pregunta se responde con HOLA, el programa (que debe funcionar en la mayoría de los ordenadores personales) debería imprimir la línea:

```
EL PROGRAMA ES
10 PRINT "HOLA"
```

Si se aplica la misma técnica a las fases de entrada, cálculo y salida de la aplicación que se tiene en mente, se podría escribir un generador de programas muy simple. Si todas las preguntas que efectúa el ordenador son expresadas claramente, una persona que no tuviera experiencia previa podría desarrollar, con el generador, un programa sencillo.

Herramientas del oficio

"El juego de herramientas del programador" para numerosos ordenadores personales, puede encontrarse en el mercado en forma de chips de ROM o cartuchos. Permite ampliar la gama de órdenes en lenguaje BASIC, especialmente para escritura de programas y eliminación de errores

Kevin Jones

Los generadores de programas producidos comercialmente emplean una técnica semejante. La mayoría de las aplicaciones de gestión consiste en una combinación de cinco procesos distintos: entrada de datos, salida a pantalla o impresora, almacenamiento en un archivo de datos, recuperación de información y cálculo. El generador tendrá unas subrutinas normales y muy flexibles para cada uno de estos procesos. Al pedir que se especifique la estructura exacta de los datos, y cómo deben aparecer en la pantalla y en la impresora, se hará que el generador cambie los valores de algunas variables de las subrutinas y las relacione entre sí para crear el programa.

Aunque los generadores de programas cada vez se vuelvan más sofisticados, no serán capaces de sustituir, en un futuro inmediato, a los programadores humanos, debido a que se encuentran limitados por los siguientes factores. Primero, la técnica descrita es muy adecuada para aplicaciones comerciales tales como contabilidad o control de stocks, pero generalmente estos generadores de programas no pueden aplicarse en el procesamiento de textos o en los programas para juegos. En segundo lugar, debido a que el generador de programas debe hacer uso de las subrutinas flexibles normales, el listado resultante no será tan eficiente (desde el punto de vista de la velocidad y de la memoria utilizada) como si hubiera sido escrito con esa finalidad por un programador. En tercer lugar, los programas producidos por los generadores no suelen resultar tan cómodos de utilizar como los sistemas que realizan hoy los programadores humanos.

Por último, los generadores de programas existentes hoy en día, en la práctica sólo son capaces de sustituir la última etapa de la programación: la escritura del código. El usuario todavía tiene que esforzarse en pensar la disposición exacta de los datos, entrada y salida que serán necesarios para que funcione el programa. Generalmente, las primeras etapas de la programación son las más difíciles de desarrollar, y requieren que, además de tener un conocimiento profundo de programación, se posean también otras cualidades. Las compañías más importantes disponen de especialistas, llamados *analistas de sistema*, para especificar los programas que necesitan, y estas especificaciones son codificadas por los programadores. Para poder crear un programa de ordenador, los generadores de programas aún deben adquirir estas cualidades.

Sobre el arco iris

Algunos ordenadores pueden adquirir mayor complejidad gracias a una serie de complementos. Ampliemos al máximo un Spectrum

Acoplador acústico
El Micro-Myte 60, representado aquí, permite poner en comunicación dos ordenadores

Microdisco
El microdisco usa cartuchos que contienen un bucle cerrado de cinta magnética; cualquier punto dado de éste pasa por el cabezal de lectura-escritura cada siete segundos. La transferencia de información se produce a 6 Kbytes por segundo (cuatro veces la velocidad de un reproductor normal de cassette). Tal como vemos, se pueden conectar conjuntamente hasta ocho microdiscos, que proporcionan una capacidad total de 700 Kbytes o más

Teclado
El teclado FDS de Fuller posee teclas para funciones y un espaciador que abarca toda su anchura

Palancas de mando
Mediante la interface 2 de Sinclair puede acoplarse cualquier palanca de mando que emplee la interface Atari, sin que haya que tener en cuenta su sistema de funcionamiento. La interface 2 puede conectar dos palancas a un tiempo

Paquete RAM
El Spectrum de 16 Kbytes puede ampliarse mediante la adición de un cartucho suplementario de 32 Kbytes

Cuando fue presentado a principios de 1982, el Sinclair Spectrum fue recibido como un adelanto decisivo, tanto desde el punto de vista de su rendimiento como de su precio. En 1983, su primer año completo de producción, las ventas del Spectrum (600 000 unidades) representaban más de la mitad del número total de ordenadores personales comercializados en Gran Bretaña, éxito sorprendente incluso para los mismos fabricantes. El Spectrum constituía, ciertamente, una mejora considerable con respecto al anterior modelo de Sinclair, el ZX81, con 16 o 48 Kbytes de RAM como capacidad normal de memoria; ocho colores para márgenes, fondo y texto, y una capacidad limitada de resolución gráfica; un teclado mejorado, pero todavía poco eficiente; y la posibilidad de generar sonidos simples. Pero todas estas características diferentes no fueron impedimento para que los fabricantes independientes produjeran una amplia gama de accesorios. La misma Sinclair no ha permanecido inactiva en este aspecto, añadiendo almacenamiento secuencial, en forma de microdisco, e interfaces para la conexión de cartuchos de ROM y palancas de mando para juegos.

Monitor TV
Como no se puede adaptar un monitor a la salida del Spectrum, es difícil obtener gráficos de alta calidad. En la figura se muestra el Profeel de Sony, sistema de televisión modular que tiene el receptor separado del monitor

Impresora en serie
Además de la impresora ZX, cabe la posibilidad de montar una matriz de puntos convencional o una impresora margarita, utilizando una interface RS232 o Centronics. La interface 1 de Sinclair admite dispositivos en serie

Sintetizador de música
El Trichord de Petron constituye un buen ejemplo de un tipo de sintetizador de música adecuado para el Spectrum. El Trichord es una unidad de tres voces que ofrece hasta 6134 acordes de tres notas si se utiliza con un Spectrum de 48 Kbytes

Reproductor cassette
Incluso con el microdisco montado, el Spectrum puede escribir o leer conectado a una grabadora de cassette doméstica. Por lo tanto, se puede utilizar el software comercial en cassette

Sintetizador de voz
El Cheetah Sweet Talker es un sintetizador que emplea el sistema alofónico simplificado para formar palabras a partir de las sílabas que las componen. Sesenta y tres alófonos y cuatro intervalos de espacio de longitudes variables constituyen esta unidad

Interface 2
Algunas compañías independientes han producido interfaces para palancas de mando programables, aunque el mismo modelo Sinclair incorpora una conexión ROM, de forma que los juegos y lenguajes alternativos sean más fáciles de montar y usar

Impresora ZX
Con el fin de producir una impresora de bajo coste, suficiente para cubrir las necesidades básicas de un usuario de ordenador personal (primordialmente listados de programas), Sinclair ha creado una unidad que emplea una forma de erosión por chispa para imprimir sobre un papel de base alumínica

Teclado
Si bien el teclado estándar del Spectrum está muy mejorado con respecto a los que incorporaban los ZX80 y ZX81, éste continúa teniendo una configuración inadecuada

Interface 1
Diseñada específicamente para servir al micromando, la interface 1 también contiene un sistema de circuitos que permite usar varios dispositivos, como impresoras, plotters o modems. Además, hace posible conectar diversos Spectrum entre sí en una red local

Diseñada a medida

La ULA (disposición lógica no comprometida) controla todas las funciones de un ordenador personal, salvo la CPU, ROM y RAM

De los numerosos avances producidos en el diseño electrónico, a consecuencia del gran desarrollo que ha tenido lugar en la industria del microordenador, uno de los más significativos ha sido la realización de un tipo de chip que recibe el nombre de *disposición lógica no comprometida* (*Uncommitted Logic Array*: ULA), con el cual es posible construir ordenadores muy complejos, así como otros dispositivos, con sólo cuatro componentes: CPU, RAM, ROM y —para unir a los tres— una ULA.

¿Y qué es una ULA? Tal como sugiere su nombre, consta de un gran número (una disposición) de puertas lógicas, las cuales, inicialmente, no están comprometidas pero pueden ser modificadas para realizar casi todas las operaciones que necesita el diseñador. La ULA puede considerarse como un desarrollo de la ROM, puesto que el contenido de ambas sólo puede ser determinado por el fabricante del chip, y no por el usuario.

Antes de que una ROM o una ULA sea "programada", ésta únicamente consiste en un gran número de circuitos (o celdas) electrónicos sencillos, que no están conectados y, por tanto, no pueden realizar ninguna acción. Todos los chips se construyen por la superposición de capas de materiales semiconductores (véase p. 122). La última capa, normalmente, está integrada por un material conductor, y forma las conexiones entre las diversas celdas. Lo que proporciona a la ULA su gran flexibilidad es la gran variedad de interconexiones posibles; y si bien la constitución de cada celda es bastante simple, consistiendo nada más que en un par de transistores o en una sola resistencia, pueden conectarse entre sí mediante la capa final y formar circuitos bastante complicados: un flip-flop, por ejemplo (véase p. 305).

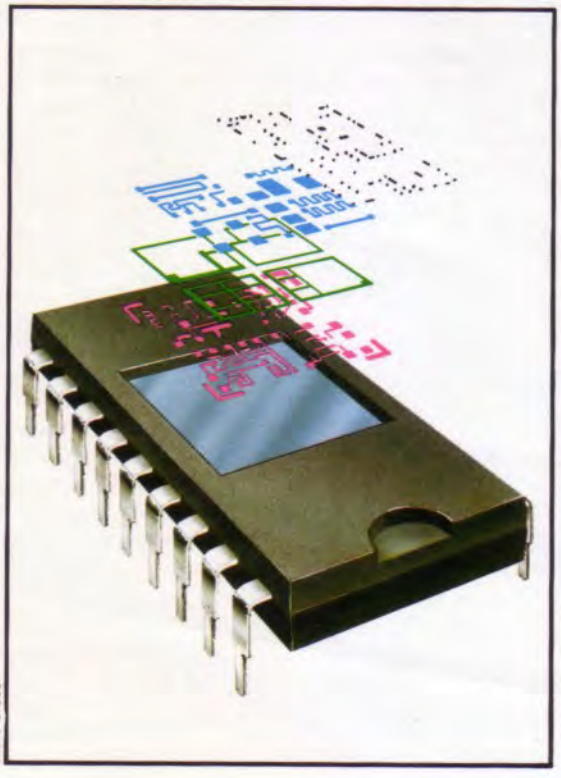
Estos circuitos, llamados *módulos*, se pueden construir con menos de media docena de celdas, y puesto que una ULA de gran tamaño puede tener varios miles de celdas, los mismos módulos pueden conectarse entre sí para constituir circuitos complejos, tales como registradores, contadores o circuitos de medición.

Una ULA puede programarse para que realice una gama extremadamente amplia de actividades. Puede utilizarse como sintetizador de sonido, o para controlar la exposición, el enfoque y el motor de una cámara, o para efectuar la mayoría de las funciones de un termómetro digital. Y, aparte de la ULA, apenas es necesario otro circuito externo.

Tal como se podía esperar, en el proceso de diseño de la capa que conecta las células de una ULA se emplean ampliamente los ordenadores. Un miniordenador como el DEC PCP11/23 puede hacer funcionar un sistema de diseño auxiliado por ordenador y crea en primer lugar un diagrama codificado de la lógica deseada. Luego, el sistema traza, y asimismo codifica, un mapa del esquema planeado. Esto es efectuado por un terminal de gráficos, y en un plotter se puede producir una copia.

Una vez se ha completado el diseño, éste es transmitido a un ordenador de mayor capacidad, el cual verifica que el plan es aceptable, lo compara con el diseño lógico original y comprueba que no tenga ningún error de importancia. Luego, es sometido a otro programa que simula el circuito que resultaría, utilizando un programa piloto suministrado por el cliente. Una vez finalizado el diseño, el ordenador puede producir la plantilla óptica que se emplea para construir la capa final.

¿Hasta dónde pueden llegar las ULA? La idea de colocar un gran número de circuitos sencillos de silicio y permitir que el usuario decida cómo deben actuar es tan atractiva que se puede convertir en el método aceptado de realizar la mayor parte de los sistemas de circuitos. Sin embargo, en el nivel presente de la tecnología, las ULA únicamente son rentables cuando se necesitan, como mínimo, unos dos mil circuitos idénticos. Las PROM (*Programmable Read Only Memory*: ROM programable), EPROM (*Erasable PROM*: PROM que puede borrarse); EEPROM (*Electrically Erasable PROM*: PROM que puede borrarse eléctricamente) y EAROM (*Electrically Alterable ROM*: ROM alterable eléctricamente) son todas alternativas de la ROM, que pueden ser programadas por el usuario mediante un equipo adecuado.



Steve Cross

ULA
Todos los chips semiconductores se construyen a partir de capas de depósitos de semiconductores, que son tratados químicamente de forma individual para crear los elementos del circuito. La última capa determina la conexión entre los elementos. Una ULA consiste en una disposición de elementos lógicos que pueden combinarse para formar un circuito lógico complejo

Frente a la realidad

La simulación es una de las mejores aplicaciones educativas de los ordenadores. Veamos con detenimiento algunos programas

Vientos

Winds (Vientos), programa de simulación realizado por la Longmans para el BBC Micro, parte del supuesto de que la persona que lo está haciendo funcionar es el patrón de una vieja goleta. En la pantalla del monitor aparece un mapa del globo terráqueo, en el cual el navío está representado por un pequeño punto y se gobierna mediante una brújula. Se eligen los puertos de salida y llegada y la fecha en que empieza la travesía. La velocidad del barco depende de la dirección y la intensidad de los vientos dominantes, que aparecen representados en la parte inferior de la pantalla. Otros parámetros representados en el monitor son: la posición de la goleta, indicada por la longitud y la latitud; la zona de vientos (alisios, p. ej.); la fecha; la distancia ya recorrida, y la duración total de la travesía.

Tomemos la ruta más directa desde Londres hasta Río de Janeiro, y supongamos que se zarpa el primero de enero. El barco se dirige hacia el sur y avanza rápidamente gracias a los vientos dominantes en la zona, hasta llegar a la línea del ecuador. Aquí, debido a la zona de las calmas ecuatoriales, no progresa durante tres días. Finalmente, empiezan a soplar los vientos alisios del sudoeste, pero esto crea un problema: ¿cómo navegar en dirección sudoeste con vientos procedentes de este mismo cuadrante? La solución es navegar en zigzag (o en "virada"), primero hacia el sur y luego hacia el oeste, hasta alcanzar Río de Janeiro, después de recorrer 15 480 km durante 207 días.

Otros viajes pueden estar erizados de peligros: huracanes, hielos polares, naufragios, pueden ser algunos de los contratiempos que haya que afrontar. Este programa de simulación se puede utilizar de varias formas distintas. La más sencilla de ellas podría consistir en emplearlo a modo de juego educativo para enseñar a niños los puntos cardinales de la brújula.

Simulación de vuelo

Flight simulation (Simulación de vuelo) es una versión para ordenadores personales de un juego recreativo, en el cual uno mismo es el piloto de un pequeño aeroplano. La pantalla muestra la visión que tiene el piloto desde la cabina del avión, con el cuadro de instrumentos en la parte inferior de aquélla. Éste está compuesto por diversos aparatos de medición, diales y testigos ópticos, que deben ser observados atentamente con el fin de pilotar el avión de manera correcta. La columna de mando del aeroplano está representada por las cuatro teclas del ordenador marcadas con una flecha; los otros controles (potencia, equipo para ayuda en el aterrizaje, etc.) también son manejados a través del teclado.

Sin embargo, de forma diferente que en los juegos recreativos, aquí se tiene la oportunidad de familiarizarse con los mandos, empezando a pilotar con el avión ya en el aire. Para obtener un conocimiento de las características del vuelo, se puede visualizar en la pantalla un plano que indica la posición del aparato, de los aerofaros de navegación y de las pistas de aterrizaje. La mejor forma de navegar es no perder de vista alguno de los aerofaros de navegación, y luego ladear el avión hasta que quede alineado con éste. Esto es indicado por el punto luminoso intermitente del "Reloj RDF", que se desplaza hasta alcanzar la parte superior del dial. Entonces se debe volar en línea recta hasta estar a la altura del aerofaro. Mediante este método, resulta fácil llegar hasta el aeropuerto.

La maniobra más difícil de efectuar es el aterrizaje. Es necesario estar perfectamente en línea recta con la pista y aproximarse a la velocidad, altura y ángulo exactos. Aun en el caso de que todos estos factores sean correctos, se puede cometer un fatal error: ¡olvidarse de bajar el tren de aterrizaje!

Simulación fisiológica

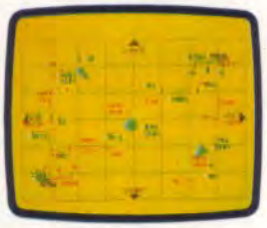
En *Physiological simulation* (Simulación fisiológica) se representa el cerebro humano y la misión consiste en ¡mantener con vida al cuerpo durante 50 minutos! El programa simula los diversos cambios fisiológicos (temperatura del cuerpo, pérdida de agua, etc.) que se producen cuando el cuerpo humano realiza diversas actividades. Lo primero que hay que hacer es introducir en el ordenador la edad y el sexo de la persona y la actividad que se desea realizar. La más fácil de simular es dormir, puesto que el cuerpo gasta poca energía. Otras pueden consistir en andar, subir una montaña y, la más agotadora de todas, correr.

Los parámetros que se deben controlar son: la capacidad torácica, el ritmo respiratorio y la transpiración. Hay que elegir unos valores iniciales; por ejemplo, una capacidad torácica de 2,5 litros, un ritmo respiratorio de 15 veces por minuto, y una cantidad de sudor de tres gramos por minuto, y a partir de aquí se puede empezar la simulación.

En la pantalla se ven cinco gráficos de las diversas funciones corporales, y un reloj. A medida que va transcurriendo el tiempo, los gráficos suministran datos de cómo se está desarrollando la actividad elegida, y debe evitarse que cualquiera de ellos alcance un nivel peligroso. Si, por ejemplo, la temperatura del cuerpo aumenta excesivamente, se puede suspender la actividad durante un corto período de tiempo e incrementar el régimen de transpiración para intentar que vuelva a bajar. Si esta tentativa resulta infructuosa y un gráfico sobrepasa el nivel de peligro, se puede recibir un diagnóstico conciso como "LA PERSONA ESTA MUERTA".

Vientos

Simulación de vuelo



Simulación fisiológica

Ian McKinnell

Memotech MTX 512

Este ordenador destaca por su alto nivel de construcción y su interesante software para el manejo del código máquina

El Memotech MTX 512 posee una serie de características que lo acercan en gran medida a lo que se denomina un ordenador "MSX standard" (véase p. 252); y si no fuera por la utilización del chip de sonido 76489 de Texas Instruments (MSX designa un AY-3-8910 de General Instruments), el MTX 512 podría considerarse como una de estas máquinas "estándar". Por otra parte, sin embargo, se ajusta a las especificaciones MSX al contar con una unidad central de proceso Z80, con un videoprocesador TMS9918/9928 de Texas Instruments, y una versión de BASIC que se halla aceptablemente cercana a la versión Microsoft.

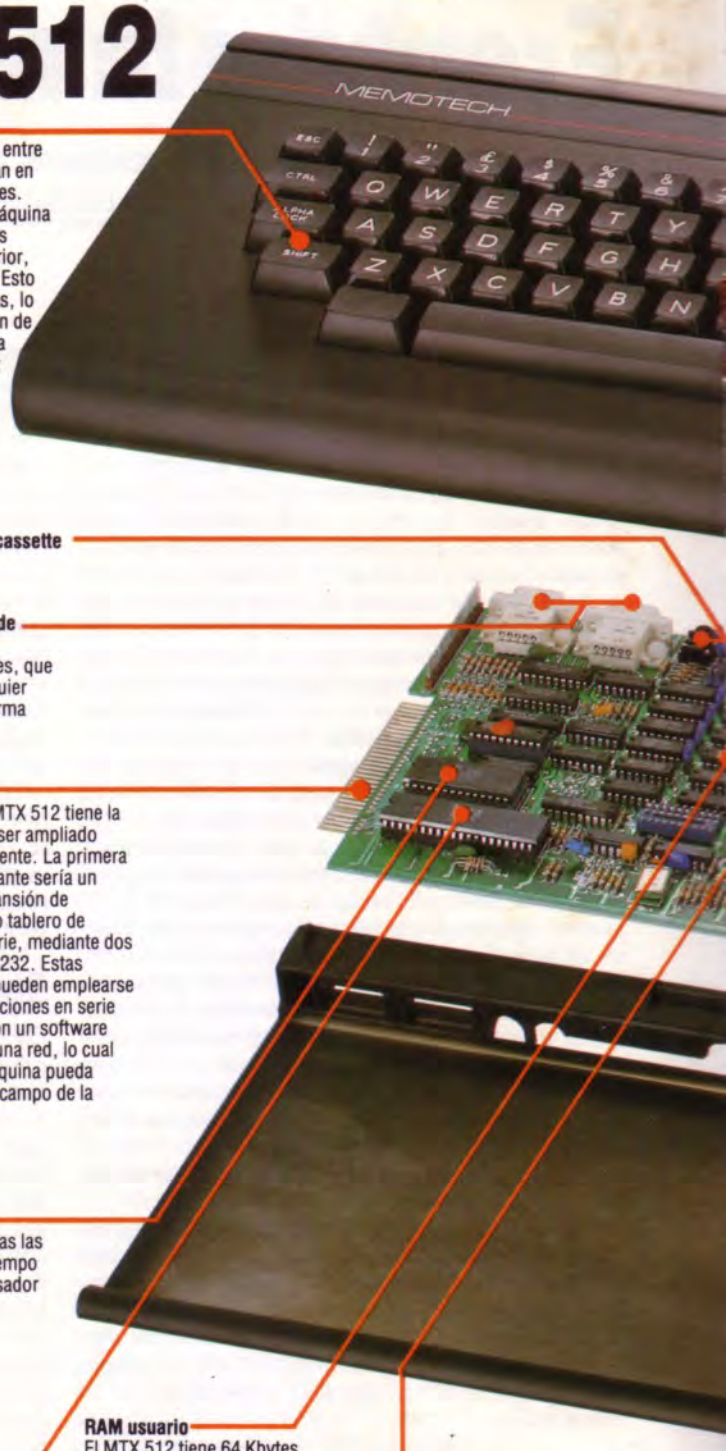
El Memotech MTX 512 es una máquina tan completa y con un diseño tan elegante que no cabe la menor duda de que conseguirá numerosos adeptos. Su apariencia externa representa un avance considerable con respecto a otros muchos ordenadores, en los cuales, con bastante frecuencia, se introducen apretadamente componentes electrónicos de una complejidad muy elevada en una envoltura barata y muy poco dimensionada, de color negro y con un diseño atractivo, construida de aluminio.

La máquina está diseñada de forma que permita un fácil acceso al interior (simplemente quitando dos tornillos Allen y girando la cubierta inferior) para dejar a la vista el tablero de circuitos. Comparado con otros ordenadores, el MTX 512 tiene un número relativamente grande de chips. Es evidente que los diseñadores de la máquina prefirieron, o encontraron que era más económico, no emplear algunas ULA de gran capacidad. Mediante la utilización de una distribución más tradicional, consistente en numerosos chips montados de modo hermético, la máquina facilita un diagnóstico de fallos más rápido y fácil. En cambio, con la utilización de ULA, las averías son más difíciles de localizar e imposibles de reparar.

El manual de instrucciones no es tan bueno como el de otras marcas y, aparte de las cubiertas, no tiene ningún esquema en color. Otra desventaja la constituye la falta de un índice, lo cual dificulta su manejo. Sin embargo, es un manual bastante amplio y completo. Memotech ha decidido hacer una máquina "abierta", en el sentido de que no oculta ningún secreto al comprador. La información sobre la máquina es presentada con todo tipo de detalles: todos los esquemas de la memoria, tablas de posiciones útiles, formas de input/output, el diagrama de los circuitos y una buena introducción al lenguaje BASIC. También se incluyen unos capítulos específicos sobre NODDY (véase recuadro), el ensamblador-desensamblador y gráficos.

Una característica que distingue al Memotech MTX 512 de otros ordenadores es poseer un ensamblador-desensamblador que puede proporcionar, junto con el paquete de software Front Panel que se suministra, una programación de código máquina. El paquete ensamblador, sin embargo, no puede manejar direcciones simbólicas ni etiquetas; pero dado que, durante la programación, se mantienen notas pormenorizadas, resulta bastante adecuado para programas de tamaño medio. En próximos capítulos del curso veremos con mayor detalle tanto los paquetes ensambladores como el código máquina.

El Front Panel es una adición original en una máquina de este nivel, y es capaz de corregir cualquier tipo de error de código máquina. Desafortunada-



Teclado
Este teclado se encuentra entre los mejores que se montan en los ordenadores personales. Tiene 79 teclas del tipo máquina de escribir profesional, las cuales, en su parte posterior, tienen una hoja de acero. Esto hace que sean muy rígidas, lo que, junto con la fundición de aluminio, proporciona a la máquina una gran solidez

Interface cassette

Conectores de palancas de mando
Dispone de dos conexiones, que pueden adaptarse a cualquier palanca que emplee la norma Atari

Ampliación
El Memotech MTX 512 tiene la posibilidad de ser ampliado considerablemente. La primera adición importante sería un tablero de expansión de memoria y otro tablero de interface en serie, mediante dos conexiones RS232. Estas ampliaciones pueden emplearse para comunicaciones en serie normales o, con un software adecuado, en una red, lo cual hará que la máquina pueda competir en el campo de la educación

Chip reloj
El Z80 CTC suministra todas las funciones de control de tiempo usadas por el microprocesador

RAM usuario
El MTX 512 tiene 64 Kbytes estándar. El MTX 500 tiene 32 Kbytes

CPU
El microprocesador Z80 se utiliza con una velocidad de reloj de 4 MHz

Interface en paralelo
Esta conexión se ajusta a las normas Centronics para interfaces en paralelo y, junto con las interfaces RS232, permite que el MTX 512 dirija prácticamente cualquier impresora

MEMOTECH MTX 512

DIMENSIONES

488×202×56 mm

CPU

Z80

VELOCIDAD DEL RELOJ

4 MHz

MEMORIA

ROM: 24 K
RAM: 64 K de RAM usuario, más 16 K RAM video
Ampliable a 512 K

VISUALIZACION EN VIDEO

24 líneas de 40 caracteres, 16 colores con fondo y primer plano ajustables independientemente. 127 caracteres predefinidos y otros 127 definibles por el usuario

INTERFACES

Cassette, TV, monitor video compuesto

LENGUAJE SUMINISTRADO

BASIC, NODDY, ensamblador

OTROS LENGUAJES DISPONIBLES

No determinados

VIENE CON

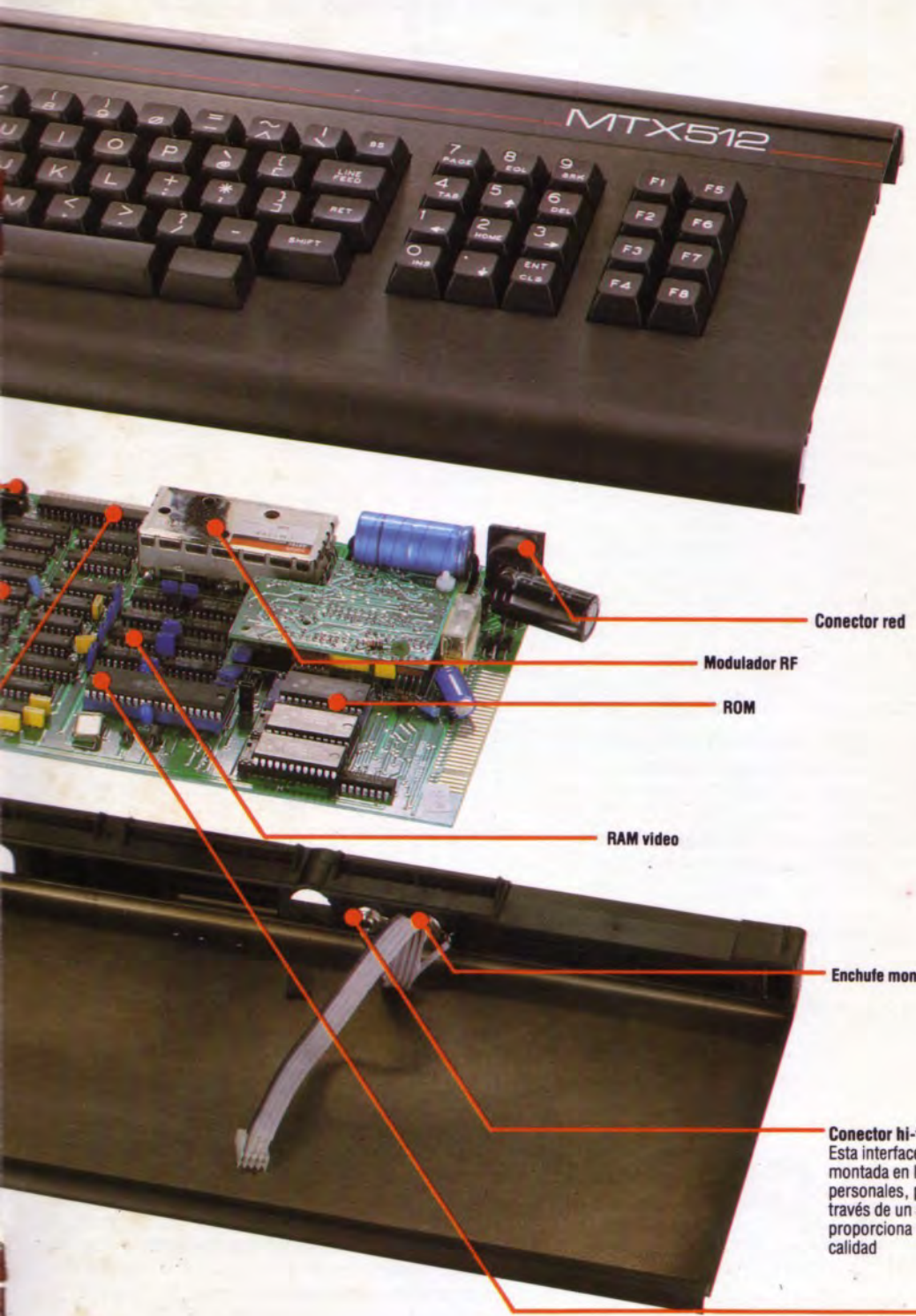
Instalación y manuales BASIC, cable TV

TECLADO

79 teclas de alta calidad

DOCUMENTACION

Razonablemente completa, pero de presentación poco atractiva. Contiene información suficiente sobre el funcionamiento interno de la máquina, lo que permite a la mayoría de los programadores experimentados conseguir un control total



Conector hi-fi
Esta interface, raramente montada en los ordenadores personales, permite la salida a través de un altavoz de hi-fi, que proporciona un sonido de mayor calidad

Chip de gráficos
El TMS 9928 de Texas Instruments controla todos los aspectos de la generación de video y hace que el MTX produzca unos gráficos de características similares al TI99/4A y al Sord M5. Asimismo, el sistema de funcionamiento de este ordenador presenta otras propiedades útiles, como, por ejemplo, la capacidad de dividir la pantalla en varias ventanas

mente, éste es un aspecto de la máquina que posee una documentación muy somera, y aunque se proporciona una lista de las diversas órdenes, se facilita muy poca información sobre sus funciones, y muy pocos ejemplos de su empleo.

El Memotech MTX 512 puede ampliarse considerablemente, y con las diversas extensiones que están previstas, puede convertirse en una máquina muy capaz. Sin lugar a dudas, satisfará a numerosos usuarios y estimulará el desarrollo de un amplio software complementario.

NODDY

El sistema de software incluye un subconjunto de lenguaje NODDY y proporciona a la máquina una nueva dimensión. El NODDY, creado para ser utilizado por usuarios inexpertos, da la impresión de ser un lenguaje sencillo. Sin embargo, posee algunas órdenes de una elevada sofisticación. Está limitado porque tiene sólo 11 órdenes; además, no hay posibilidad de efectuar operaciones aritméticas. Ello es debido a que el lenguaje está ideado principalmente para manejar información textual. Quienes se inician en el manejo de ordenadores, a menudo hallan más fácil usar como datos básicos un texto que números

Sonido espectacular

El Oric-1 permite un sofisticado control del sonido

MUSIC y PLAY
El siguiente programa emplea las órdenes MUSIC y PLAY para tocar siete veces el acorde de *do mayor* invertido (*do*, *sol* y *mi*), usando cada vez una envoltura:

```
10 REM********
20 REM *ACORDE*
30 REM********
40 MUSIC 1,4,1,0:REM
   *DO*
50 MUSIC 2,3,8,0:REM
   *SOL*
60 MUSIC 3,3,5,0:REM
   *MI*
70 FOR E=1TO7:REM
   *SELEC.ENV.*
80 PLAY 7,0,E,750:REM
   *TOCA ACORDE*
90 PLAY 0,0,0,0:REM
   *STOP ACORDE*
100 WAIT 50:REM
    *PAUSA*
110 NEXT E:REM *OTRA
    ENV*
```

La gama de posibilidades del ordenador Oric-1 es muy amplia, y entre éstas quizá una de las más sobresalientes sea su capacidad de creación de sonido. Tiene una escala de siete octavas y en su versión estándar está provisto de tres osciladores, un generador de ruidos y siete envolturas preseleccionadas (véase p. 276) que pueden elegirse para dar forma a los sonidos producidos. La salida del sonido es a través del altavoz del televisor.

El BASIC del Oric-1 define un conjunto de órdenes para la elaboración del sonido: ZAP, PING, SHOOT y EXPLODE. El programa siguiente muestra cómo se pueden emplear estas órdenes, y da información sobre la orden WAIT, que detiene el ordenador durante un tiempo establecido, en centésimas de segundo (en este caso, dos segundos):

```
10 ZAP:WAIT 200
20 EXPLODE
30 GOTO 10
```

La orden SOUND se emplea sobre todo para producir efectos especiales, y se construye como sigue:

```
SOUND C,P,V
```

donde C=número del canal u oscilador (1-6); P=tono (10-5000), y V=volumen (0-15). Si el canal se coloca en 1, 2 o 3, se selecciona uno de los tres osciladores (4, 5 o 6 son equivalentes, pero también seleccionan el ruido). El tono no es del todo preciso; 10 es la nota más alta (a aproximadamente 10 KHz), y 5000 la más baja (a 100 Hz). El volumen máximo se obtiene colocando el mando a 15, pero el más agradable es a 6. Si V se sitúa en 0, el control es efectuado por una envoltura de volumen, seleccionada por la orden PLAY.

El inconveniente más importante de la orden SOUND es que no se puede determinar la duración de una nota. Además, SOUND no puede pararse por sí misma. El sonido de una nota sólo se detiene con el empleo de la orden PLAY y a continuación colocar todo ceros para que cese PLAY.

La orden MUSIC es ideal para especificar con precisión las notas. Su construcción sencilla hace que pueda entenderse con facilidad un programa de música relativamente complejo. El formato es:

```
MUSIC C, O, N, V
```

donde C=canal (1, 2 o 3); O=octava (0-6); N=nota (1-12), y V=volumen (0-15). Esta orden funciona de forma semejante a SOUND. El canal selecciona los osciladores 1, 2 o 3 (aunque en MUSIC no podemos obtener ruido). La gama de volumen abarca desde 0 (en cuyo caso el control lo efectúa la orden PLAY) hasta 15. La octava permite la selección de una octava determinada, en la que la nota (N) será parte de las órdenes. Una octava situada en 0 produce las notas más bajas, empezando en 32,7 Hz. Colocada en 6 se alcanzará la frecuencia de 3951,07 KHz. Para la nota (N) componente de la orden, los números 1 a 12 corresponden a las notas musicales normales, con la siguiente distribución:

1	2	3	4	5	6
DO	DO#	RE	RE#	MI	FA
7	**8**	**9**	**10**	**11**	**12**
FA#	SOL	SOL#	LA	LA#	SI

Luz y color

Gráficos del Spectrum: limitados pero fáciles de utilizar

El ordenador Spectrum representa un buen punto de partida para aquellos usuarios que estén interesados en gráficos en color de alta resolución. Su simplicidad de uso hace que el diseño de gráficos sea fácilmente accesible, incluso para quienes tengan una experiencia limitada en programación.

En este ordenador se dispone de los juegos normales de caracteres en mayúsculas y minúsculas, junto con varios caracteres exclusivos de Sinclair. Éstos pueden imprimirse en cualquiera de los ocho colores. Los colores para los caracteres, pantalla y recuadros se obtienen a través de las órdenes INK, PAPER y BORDER, respectivamente. Además del juego de caracteres estándar, el usuario puede definir hasta un total de 21 caracteres gráficos.

La visualización en pantalla consiste en 24 filas de 32 espacios cada una de ellas. Las dos últimas filas de la parte inferior, sin embargo, están reservadas para mensajes desde el ordenador o para entradas a través del teclado. Por tanto, la pantalla que en realidad se puede emplear es de 22×32 caracteres. En alta resolución, esto se convierte en 176×256 pixels. Una característica extremadamente útil del Spectrum es que tiene la posibilidad de mezclar en la pantalla gráficos de alta resolución junto con texto, permitiendo la creación de diagramas con epígrafes, gráficos, etc. Una vez que se ha diseñado una visualización en pantalla, es posible guardarla, mediante la orden SAVE, en una cinta magnética y volverla a utilizar cuando sea necesario. La orden SCREEN$ se encarga de efectuar este cometido, y puede usarse también para transferir el contenido de la pantalla a una impresora.

Para tocar la nota *la* a 440 Hz en el canal 1 a un volumen de 6, la orden debería ser:

```
MUSIC 1, 3, 10, 6
```

Sin embargo, para aprovechar todas las posibilidades del Oric-1, es mejor utilizar la orden MUSIC conjuntamente con PLAY. Esta última se forma de la manera siguiente:

```
PLAY C, N, E P
```

donde C=canal (0-7); N=ruido (0-7); E=envoltura (1-7); y P=período de envoltura (0-32767). Canal y ruido seleccionan unas opciones más complejas que las órdenes anteriores, tal como se indica a continuación:

Número	Canal	Ruido
0	Ningún osc.	Ninguno
1	Osc. 1	+Osc. 1
2	Osc. 2	+Osc. 2
3	Oscs. 1 y 2	+Oscs. 1 o 2
4	Osc. 3	+Osc. 3
5	Oscs. 1 y 3	+Oscs. 1 o 3
6	Oscs. 2 y 3	+Oscs. 2 o 3
7	Oscs. 1, 2 y 3	+Oscs. 1, 2 o 3

Las órdenes de MUSIC (o SOUND) definidas previamente, con el volumen situado en 0, pueden darse (PLAY) conjuntamente, de acuerdo con el número de canal seleccionado, para producir acordes de hasta tres notas. La parte correspondiente al ruido de esa orden selecciona cuál de los osciladores, en el caso de que haya alguno, es el que debe efectuar la mezcla. La E selecciona una de las siete envolturas de volumen para la nota o notas especificadas. Estas opciones están indicadas en el manual de instrucciones del Oric.

El único control variable sobre la envoltura es P, que permite al programador especificar su duración total (desde 0 hasta 32767). El período varía con cada envoltura, pero como dato orientativo se puede decir que con un valor de 5000 dura, aproximadamente, 2 segundos.

Las órdenes de sonido del Oric-1 son fáciles de usar, y constituyen una prueba evidente del esfuerzo realizado para incorporar a este ordenador unas configuraciones de alto nivel. Sólo hay otro ordenador personal que posee también un BASIC práctico y un control sobre la envoltura: es el BBC Micro, cuyas posibilidades de creación de sonido son mucho más notables. A pesar de ello, el bajo coste del Oric-1 hace que éste sea un ordenador valioso para quienes quieran componer música electrónica con un presupuesto bajo.

SOUND
Este pequeño programa utiliza la orden SOUND para producir un ruido como de una nave espacial que aterriza:

```
10 REM**********
20 REM *ATERRIZAJE*
30 REM**********
40 FOR P=10 TO 3000
   STEP 10
50 SOUND 2,P,6
60 PLAY 2,0,1,1
70 NEXT P
80 WAIT 75
90 PLAY 0,0,0,0
100 END
```

Asimismo puede lograrse que en la pantalla aparezca un gráfico de baja resolución. Para ello debe usarse la orden PRINT AT, la cual permite situar el carácter tanto horizontal como verticalmente. Este ordenador dispone también de una serie de efectos especiales. Además de la visualización, los caracteres pueden ser intermitentes o fijos empleando las órdenes FLASH o BRIGHT respectivamente. La orden OVER es también útil para la baja resolución. Permite que un segundo carácter se combine con el original cualquiera que sea su posición. Esto resulta particularmente efectivo para mezclar texto y gráficos de alta resolución, puesto que resulta posible escribir sobre los diagramas sin borrarlos. Sin embargo, este efecto debe emplearse con cierta precaución, pues si se recompone el color mediante la orden INK en un cuadrado particular, la visualización original también cambia al color nuevo.

La visualización en pantalla está gobernada por dos áreas de la memoria: una de ellas representa en la pantalla los caracteres, y la otra mantiene la información correspondiente a los atributos de una posición específica del carácter. Esta información se refiere a los siguientes puntos: los colores de los caracteres (INK) y de la pantalla (PAPER); si el carácter es intermitente (FLASH), etc. Estos atributos están representados por un solo byte, y mediante la orden ATTR de un programa en BASIC puede determinarse el estado de cualquier posición de pantalla.

Mediante las órdenes en BASIC, el ordenador Spectrum logra fácilmente visualizaciones de alta resolución. Esto se debe, en gran parte, a que no existe una pantalla independiente para alta resolución, haciendo que sea sencillo mezclar gráficos y texto en una sola representación.

Las órdenes en BASIC incluyen:

```
PLOTx,y
```

Esto hace corresponder el pixel de coordenadas (x,y) con el color que tenga el carácter (INK).

```
DRAWx,y,p
```

DRAW crea una línea entre la posición real del cursor y las coordenadas especificadas en la orden. Si se añade un tercer número, la línea dibujada será un arco de circunferencia. Este tercer número normalmente será una fracción de π (3,14159...). Si este tercer dígito fuera π, se obtendría la representación de una semicircunferencia, mientras que con π/2 trazaría un cuadrante. La orientación de la concavidad o convexidad del arco se determina haciendo que el tercer número sea positivo o negativo.

```
CIRCLEx,y,r
```

La orden CIRCLE dibuja un círculo, de centro (x,y) y radio r. Con numerosas órdenes CIRCLE en BASIC es posible formar elipses mediante la deformación de la circunferencia primitiva. Sin embargo, el Spectrum no posee esta propiedad.

Existe una desventaja importante en el uso del color en gráficos de alta resolución. Como consecuencia de la posibilidad de mezclar texto y diagramas, sólo puede especificarse un color de carácter (INK) en cada cuadrado de ocho por ocho pixels. Así, si se cruzan dos líneas de color diferente, en el interior del cuadrado donde se produce la intersección, todos los pixels tomarán el último color (INK).

Sonría, por favor
Este programa muestra el uso de las órdenes de alta resolución PLOT, CIRCLE y DRAW, para crear una cara "sonriente":

```
10 REM *CARA
   SONRIENTE*
20 CLS
30 BORDER 6
40 PAPER 6
50 INK 2
60 CIRCLE 122,88,50
70 CIRCLE 97,108,5
80 CIRCLE 147,108,5
90 PLOT 92,68
100 DRAW 152,68,PI/3
110 END
```

Control centralizado

En un ordenador personal pueden conectarse sensores para luz, temperatura y otros efectos. La información puede usarse para el control de un sistema de calefacción y una alarma antirrobo

Los microprocesadores son utilizados cada día en mayor número en una amplia gama de aplicaciones domésticas, como, por ejemplo, lavadoras automáticas, lavavajillas, grabadoras de video y unidades de calefacción central. No resulta sorprendente, por consiguiente, que se puedan interconectar estos chips de control de forma que los aparatos de toda la casa puedan compartir entre ellos la información, o transmitir los datos a un sistema central de control. Es perfectamente posible diseñar y construir sistemas de control centralizado que regulen las aplicaciones domésticas. Estos sistemas pueden dividirse en tres categorías: sistemas permanentes y temporales y controladores interconectados.

Los pertenecientes a la primera categoría están comercializados, pero también cabe la posibilidad de diseñarlos uno mismo. Éstos se acoplan a un ordenador personal convencional y emplean interfaces específicas para conectarse directamente, y regulan unidades eléctricas o electromecánicas como luces o termostatos. Sin embargo, para montar un sistema de este tipo, es necesario poseer un conocimiento amplio del hardware del ordenador y ser capaz de escribir los programas de control necesarios. Los sistemas permanentes tienen también la limitación fundamental de que el programa debe funcionar continuamente. Cualquier interrupción en el suministro del fluido eléctrico hará que el dispositivo quede, en el mejor de los casos, bloqueado en una posición determinada y no pueda llevar a cabo los ajustes y órdenes indicados por el programa de control.

La segunda categoría de sistemas de control temporal emplea señales electrónicas generadas en circunstancias determinadas por los dispositivos de regulación conectados a la calefacción central, alarma antirrobo, o detectores de fuego y humo. Cuando alguno de estos aparatos envía alguna información fuera de lo normal al ordenador, éste emite una señal prioritaria que hace que quede interrumpido el programa en uso. Si bien un sistema de este tipo debe funcionar continuamente, es mucho más adecuado para hacer frente a una avería en la fuente de alimentación, puesto que los dispositivos que controla son parcialmente autorregulables.

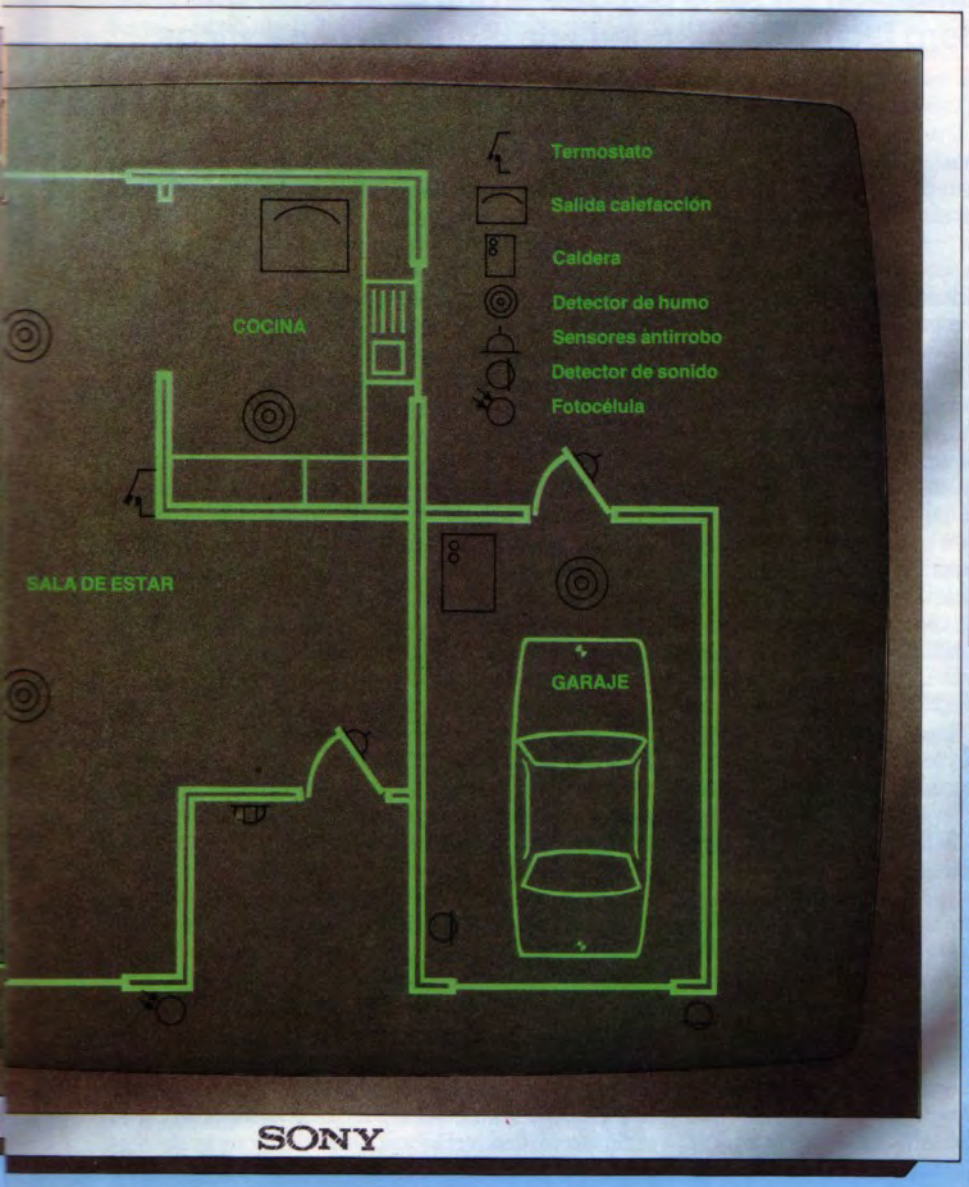
La memoria del ordenador contiene varios programas: uno para cada dispositivo conectado a él, además del programa que se está utilizando. Supongamos que en un momento determinado el ordenador está ocupado con un juego de aventuras y que entonces entra en funcionamiento el detector de humo. En respuesta a la señal, el ordenador interrumpe el programa del juego (manteniendo toda la información necesaria sobre la situación del juego en aquel preciso momento), y empieza a funcionar el programa detector de humo. En la pantalla del ordenador aparecerá un mensaje comunicando que se ha detectado un fuego potencial; o, si no se estaba empleando el ordenador, puede empezar a sonar una alarma. Una vez que se ha detectado el origen del humo y se ha solucionado el problema, se puede volver al punto exacto del juego en el momento en que se había interrumpido. Sin embargo, si la señal hubiera procedido del regulador del sistema de calefacción central, el ordenador verificaría el tiempo y los sensores de temperaturas externa e interna y, si fuera necesario, pondría en funcionamiento la caldera, realizando todos estos pasos con una rapidez tal que no se podría detectar que el juego hubiera sufrido una interrupción.

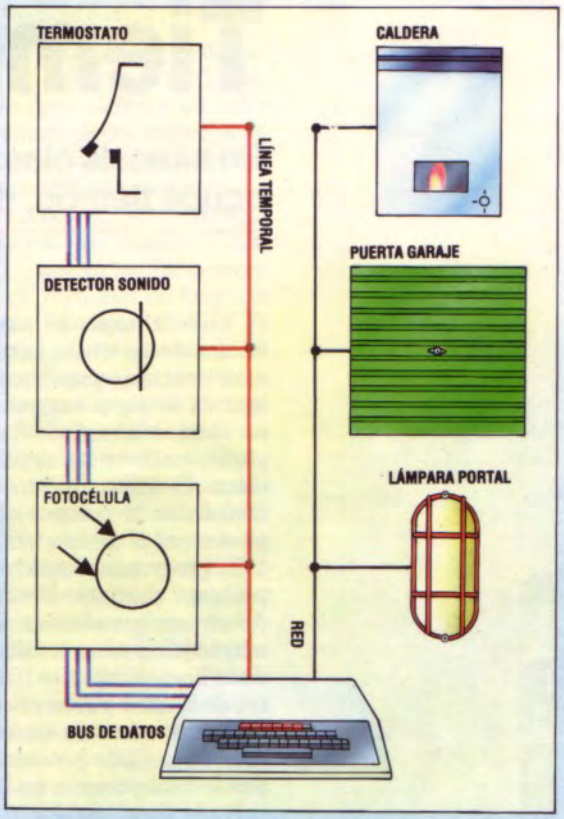
Un ejemplo de la tercera categoría (controladores interconectados) es un sistema llamado BSR Home Controller. Este ingenioso dispositivo utiliza el tendido eléctrico normal de una casa para controlar las unidades enchufadas a cualquier conexión de un circuito determinado. Cada controlador es identificado por un número de código (en realidad, una dirección), que le permite ser conectado o desconectado mediante una señal de alta frecuencia a través del tendido eléctrico normal. Sin embargo, el montaje de este tipo de interfaces es muy peligroso. Únicamente un técnico cualificado debe efectuar las conexiones desde las salidas del ordenador a la red de corriente.

Una vez que haya sido instalado el sistema de control, el siguiente paso consistirá en que éste sea capaz de funcionar mediante algún tipo de control remoto; de esta forma será posible que, incluso hallándose el usuario fuera de casa, pueda desconectar la calefacción o poner en funcionamiento la alarma antirrobo. En cualquiera de estos sistemas puede montarse un dispositivo de comunicaciones estándar, por ejemplo un modem, que permite realizar el control mediante una terminal a distancia. Cuando el sistema disponga de esta característica, para tener acceso a él será necesario utilizar algún tipo de contraseña.

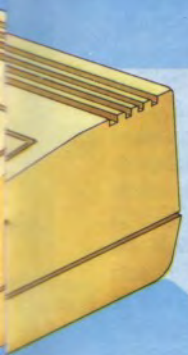
El coste principal de montaje de un sistema de control computerizado en una casa lo comporta la compra del hardware para conectar el ordenador a la red de suministro eléctrico, ya que es necesario montar numerosos aislantes, relés e interruptores de estado sólido para poder hacer el montaje de una forma segura y eficiente. Pero, probablemente, la tarea más absorbente para el usuario de un ordenador personal, para poner en funcionamiento un sistema de este tipo, sea la escritura del software. Debido a que estos dispositivos se fundamentan en su rapidez de respuesta (de nada serviría que la alarma contra fuego empezara a funcionar después de que la casa hubiera ardido), los programas de control deben escribirse en código máquina. Este tipo de programas aún no están comercializados, pero en un futuro próximo será posible adquirirlos.

Kevin Jones





Dirección doméstica
Este diagrama muestra dos de las técnicas mediante las cuales un ordenador personal puede dirigir cierto número de aplicaciones domésticas. Cuando uno de los tres sensores de la izquierda tiene algo que transmitir, envía un pulso electrónico a través del circuito común. Éste conduce al microprocesador, el cual interrumpe provisionalmente cualquier programa que esté en funcionamiento y salta a una rutina especial, que lee toda la información que el sensor suministra al bus de datos. Los componentes de la derecha están conectados a una red y, por lo tanto, el ordenador puede activar cualquiera de ellos simplemente enviando un paquete de datos consistente en, por ejemplo, el número del dispositivo de la puerta del garaje y la instrucción para abrirla



Centro de control

Es posible obtener una representación esquematizada de su casa en la pantalla del ordenador. Los sistemas de seguridad y control computerizados de plantas industriales emplean dichos métodos. Desde luego, si el ordenador funciona con el método "temporal", no es necesario obtener una visualización en la pantalla, puesto que el software efectúa todas las operaciones de control internamente, por lo general sin un retraso perceptible en el programa que funciona en ese momento en el ordenador. Es muy probable que no pasen muchos años antes de que las casas sean diseñadas incorporando dispositivos de este tipo

Tiempo y movimiento

En BASIC, la clasificación en serie puede ser una operación que ocupe tiempo, pero facilita la búsqueda de registros específicos

A estas alturas del curso de programación, ya hemos desarrollado la mayor parte de la codificación necesaria para crear las entradas en nuestra agenda de direcciones en "base de datos", pero aún no hemos abordado la programación necesaria para conservar las entradas en cinta magnética o disco. El único aspecto importante que no hemos tratado es la formación de una rutina adecuada para crear el campo MODCAM$.

El programa completo para ello se incluye en el presente capítulo. Primero, todos los caracteres deben ser convertidos al tipo de caja alta (letras mayúsculas) entre las líneas 10250 y 10330. Luego, de la línea 10350 a la 10370 se cuentan los caracteres de la serie y se verifican uno por uno para comprobar si existe un espacio. El último espacio encontrado ajusta la variable S al valor correspondiente a su posición en la serie.

De la línea 10400 a la 10420 se traspasan los caracteres, uno a uno, desde la serie en que figuran en mayúsculas hasta CNOM$. Los caracteres son trasferidos, hasta que se llega al último espacio, si tienen un valor ASCII mayor que 64. Cualquier carácter que no supere esta prueba es ignorado. Así, el proceso elimina puntos (ASCII 46), apóstrofos (ASCII 39), espacios (ASCII 32) y los demás signos de puntuación. Desde la línea 10450 a la 10470 se procede igual para los caracteres situados tras el espacio final, traspasándolos a SNOM$.

Si N$ contiene únicamente una palabra, SUPERMERCADO, por ejemplo, la variable S será 0 y todos los caracteres serán transferidos a SNOM$. La variable utilizada para designar los nombres de pila ha sido llamada CNOM$, mientras que la variable SNOM$ se utiliza para designar el apellido.

Las líneas 10490 y 10500 son necesarias para convertir en cero las variables alfanuméricas empleadas en esta rutina, antes de que sean usadas otra vez. Éste es un punto que no hay que perder de vista siempre que se empleen estructuras del tipo LET X$=X$+Y$. Si se produce un fallo en la "limpieza" de las variables, se originará una acumulación cada vez mayor de caracteres erróneos en las variables siempre que sean utilizadas. Téngase en cuenta que OPCN es situado en 0 en la rutina INCREG, puesto que únicamente se quiere comprobar que el usuario añade un registro si no hay ninguno en el archivo (es decir, la primera vez que se utiliza el programa).

Ahora que ya se dispone de un medio de añadir tantos registros como se desee en el archivo, es necesario imaginar cómo se puede conservar el archivo en cinta magnética o disco. La forma más sencilla consistiría en escribir todos los registros en el archivo de datos (DAT.AGCO, en esta versión del programa) en el orden en que han sido introducidos. La desventaja más importante de este método de aproximación se pone de manifiesto cuando es necesario buscar en el archivo un registro determinado. Si no se tiene la entera seguridad de que todos los registros del archivo están ordenados en alguna forma, la única manera de hallar un registro sería examinando cada registro desde el principio con el fin de comprobar si existe la correspondencia adecuada. Si el registro que se está buscando da la casualidad que era el último que se introdujo, será necesario examinar cada uno de los registros de la base de datos antes de poder localizar el que se buscaba. Si el último registro introducido hubiera sido el correspondiente a Lorenzo Díaz (es decir, MODCAM$ (TAMA−1)="DIAZ LORENZO"), una rutina de búsqueda haría que el registro estuviera en algún punto cercano al principio del archivo (si los registros estuvieran ordenados). Desafortunadamente, tanto la tarea de búsqueda como la de ordenación son actividades que consumen mucho tiempo; por consiguiente, hay que determinar las prioridades y considerar qué resulta más interesante. Aquí se ha adoptado el principio de que una agenda de direcciones es con mayor frecuencia consultada que modificada (añadiendo nuevas direcciones o en alguna otra forma). En tal caso, es mejor suponer que habrá más búsquedas que ordenaciones, y habrá que asegurarse de que los registros están ordenados, antes de ser almacenados en el archivo de datos, una vez utilizado el programa.

Sin perder de vista esta observación, se crea una variable, llamada RMOD, para ser usada como bandera. Puede tener dos valores: 0 o 1. Inicialmente toma el valor 0, para indicar que ningún registro ha sido modificado durante la actual ejecución del programa. Cualquier operación que modifique el archivo en alguna forma (por ejemplo, añadiendo un nuevo registro) sitúa RMOD en 1. Las operaciones que "necesiten saber" si ha sido modificado el archivo, verificarán el valor de RMOD antes de empezar a procesar. Por ejemplo, SAPROG, la rutina que guarda el archivo y las salidas del programa, comprueba RMOD, en la línea 11050. Si RMOD=0, no es necesario ningún tipo de clasificación o conservación, porque se da por supuesto que el archivo de datos, en cinta magnética o en disco, está completamente ordenado y sin modificar. Otras rutinas, tales como las que buscan un registro determinado a través del archivo, también necesitan verificar el valor de RMOD. Si éste corresponde a 0, la búsqueda (u otra operación) puede continuar. Si RMOD es 1, lo primero que deberá hacer la rutina será llamar a la rutina de clasificación. Efectuada la clasificación de todo el archivo, la rutina de clasificación volverá a poner 0 en RMOD.

En nuestro caso, la rutina de clasificación, llamada *CLSREG* en el listado del programa, vuelve a colocar la variable RMOD en 0 en la línea 11320, una vez que todos los registros han sido clasificados. Antes de continuar y centrar la atención en *SAPROG* (la rutina que guarda el archivo en cinta magnética o disco y luego da por terminado el pro-

grama), digamos algunas palabras sobre qué se entiende por *CLSREG*. Consiste en una técnica de clasificación sencilla, denominada "clasificación burbuja" (véase p. 286). Existen muchas formas de clasificar datos, y ésta es una de las más simples y lentas. Hay otras rutinas más eficaces, pero los tipos de clasificaciones más sofisticadas resultan mucho más difíciles de entender que la mencionada anteriormente. El tipo de rutina a emplear depende del número de datos que haya que clasificar. La "complejidad de tiempo" de una clasificación burbuja como la nuestra es n^2. En otras palabras, el tiempo necesario para clasificar los datos se incrementa con el cuadrado del número de elementos a ordenar. Si dos elementos requirieran cuatro milisegundos para ser clasificados, cuatro necesitarían 16 milisegundos, 50 elementos dos segundos y medio y si fueran 1 000 se tardaría más de 15 minutos. Una espera de dos o tres segundos es aceptable en un programa semejante al nuestro, pero de ninguna manera una demora de un cuarto de hora.

La forma en que ha sido escrito este programa permite un máximo de 50 registros únicamente, por tanto no se presentarán problemas de espera.

Los datos que se están clasificando son las series de caracteres de MODCAM$(L) y MODCAM$(L+1). Los registros únicamente son intercambiados si MODCAM$(L) es mayor que MODCAM$(L+1), y el campo de índice (que por el momento no se está utilizando) es actualizado en las líneas 11490 y 11570. Cada vez que se intercambian dos registros, la variable S (para indicar que se ha producido un intercambio) toma el valor 1. Cuando la rutina de clasificación alcanza la línea 11290, verifica el valor de S y vuelve atrás para comparar otra vez todos los registros. Cuando todos ellos estén en orden, el valor de S se situará en 0 y la rutina habrá finalizado, una vez que el valor de RMOD sea también 0.

La rutina SAPROG (denominada *SAPROG* en el listado del programa) empieza en la línea 11000. El primer paso que efectúa es verificar si, durante la ejecución actual del programa, se ha modificado algún registro (línea 11050: IF RMOD=0 THEN RETURN). Si no se ha producido ninguna modificación en el archivo, no será necesario proceder otra vez a la grabación en cinta magnética o disco y la rutina volverá al programa principal. Esto hace que volvamos de nuevo a la línea 100, la cual verifica el valor de OPCN. Si tiene un valor 9 (tal como debería ser si *SAPROG* está siendo procesado), el programa principal sigue hasta la orden END, en la línea 110.

Si el programa encuentra que RMOD tiene el valor 1 en la línea 11050, significa que uno o más registros han sido modificados en alguna forma y que existe alguna posibilidad de que ya no estén en orden. Si se presenta esta situación, la rutina *SAPROG* hace entrar en funcionamiento la rutina de clasificación (línea 11070) y luego, tras ser clasificados todos los registros, los conserva en cinta magnética o disco.

La rutina de guardar (*GRDREG*) es llamada en la línea 11090 y comienza a actuar en la línea 12000. *GRDREG*, en el listado principal, está escrito en BASIC Microsoft, por eso es importante recordar que los detalles del manejo del archivo varían de una versión a otra de BASIC (véase "Complementos al BASIC"). La línea 12030 abre el archivo de datos DAT.AGCO y asigna el número de canal #1 a la operación. La línea 12050 establece los límites del bucle que cuenta todos los registros a través del archivo. El límite superior es TAMA-1, y no TAMA, porque esta última variable tiene siempre un valor mayor en una unidad al número de los registros vigentes en el archivo (así, si se añade un registro nuevo, no será escrito sobre uno ya existente).

El formato de las líneas 12060 y 12070 es particularmente digno de ser tenido en cuenta. Cada campo está separado por ",", que es también enviado al archivo. La mayoría de versiones de lenguaje BASIC necesitan esta coma, porque INPUT# y PRINT# trabajan de la misma forma que las órdenes INPUT y PRINT normales. Supóngase la sentencia INPUT X,Y,Z. Ésta espera que se produzca una entrada a través del teclado semejante a 10,12,15<CR>, la cual atribuiría los valores 10, 12 y 15 a X,Y, y Z respectivamente. Sin comas, la orden INPUT no sería capaz de decir dónde termina cada dato y asignaría toda la información a la primera variable. De una forma semejante, la orden INPUT# (en la mayoría de las versiones de BASIC) no podría indicar dónde termina cada registro de archivo de datos e intentaría llenar cada variable alfanumérica con tantos datos como pudiera. Puesto que la mayor parte de las variables alfanuméricas en BASIC pueden contener hasta 255 caracteres, los datos del archivo estarían todos pronto asignados, mucho antes de que hubiera terminado el bucle FOR L=1 TO TAMA-1. Esto ocasionaría un mensaje erróneo INPUT PAST END (indicando que se había emitido una orden INPUT después de agotar todos los datos), y las variables alfanuméricas (como, por ejemplo, NOMCAM$(x)) contendrían muchos más datos de lo debido.

Una vez que todos los registros han sido almacenados en el archivo de datos, desde L=1 TO TAMA-1, *GRDREG* RETURN vuelve a la línea 90 del programa principal. La línea 100 verifica el valor de OPCN para comprobar si la última operación era *SAPROG* o no. Si era 9 (guardar y salir), el programa continúa hasta la sentencia END, en la línea 110. Si OPCN es otro valor, el programa vuelve atrás hasta *ELECCN* y permite que el usuario elija de nuevo otra opción.

Por último, debemos mencionar la rutina *FINARCH*, que empieza en la línea 12500. Constituye una alternativa posible a la sentencia de la línea 1510. Tal como está configurado el programa, éste depende de la presencia de una función de final de archivo: IF EOF(1)=−1 THEN LET L=50. Todos los lenguajes BASIC tienen alguna forma de indicar que se ha alcanzado el final de un archivo, bien sea mediante una función especial como EOF(x) o un PEEK a una posición especial de la memoria. Se sugiere la rutina *FINARCH* de la línea 12500 si no se dispone de una función EOF, en cuyo caso la línea 1510 debería ser sustituida por GOSUB 12500.

Complementos al BASIC

SPECTRUM

Antes de procesar el programa de la agenda de direcciones, se debe crear, en una cinta magnética, el archivo de campo de nombres. El siguiente programa tiene esta finalidad.

```
10 REM PROGRAMA PARA CREAR
   ARCHIVO NCAM EN CINTA
20 DIM Z$(1,30)
30 LET Z$(1)="@ VACIO"
40 SAVE "NCAM" DATA Z$( )
50 STOP
```

SPECTRUM

Cuando se pare el programa, rebobinar la cinta y digitar VERIFY "NCAM" DATA Z$ ( ) para verificar que ha funcionado SAVE. Esta verificación de los datos y programas almacenados en cinta magnética duplica el tiempo de espera hasta que se para la grabadora de cassette, pero es una norma general que merece la pena tener en cuenta, antes de desconectar la máquina.
A continuación se indican las versiones para el ordenador Spectrum de las líneas y subrutinas del listado principal:

```
1100 REM *CREMAT*
1110 DIM N$(50,30)
1120 DIM M$(50,30)
1130 DIM C$(50,30)
1140 DIM D$(50,15)
1150 DIM P$(50,15)
1160 DIM T$(50,15)
1170 DIM X$(50,30)
1180 DIM B$(30)
1190 DIM Z$(30)
1250 LET Z$="@VACIO"
.
1400 REM *LEARCH* SR
1410 LOAD "NCAM" DATA N$( )
1420 IF N$(1)=Z$ THEN LET
     Q$=Z$:RETURN
1430 LOAD "MCAM" DATA M$( )
1440 LOAD "CCAM"DATA C$( )
1450 LOAD "DCAM" DATA D$( )
1460 LOAD "PCAM" DATA P$( )
1470 LOAD "TELCAM" DATA T$( )
1480 LOAD "INDCAM" DATA X$( )
1490 REM *FINARCH*
1500 GOSUB 12500
.
1540 RETURN
.
1640 IF Q$=Z$ THEN LET TAMA=1
.
3520 IF Q$=Z$ THEN GOSUB 3860:RETURN
.
3810 LET OPCN=CODE A$-48
.
10090 LET Q$="  "
.
10200 REM*MODNOM* SR
.
10250 LET R$=N$(TAMA):LET S$=""
10260 FOR L=1 TO LEN (R$)
10270 LET A$=R$(L)
10280 LET T=CODE A$
10290 IF T>=97 THEN LET T=T-32
10300 LET A$=CHR$ T
10310 LET S$=S$+A$
10320 NEXT L
10330 LET R$=S$:LET S$="":LET
      A$="":LET T=LEN(R$)
10340 REM LOCALIZAR ULTIMO ESPACIO
10350 FOR L=1 TO T
10360 IF R$(L)="  " THEN LET S=L:LET
      L=T
10370 NEXT L
10380 REM QUITAR SOBRANTE
10390 REM ALMACENAR NOMBRES DE
      PILA EN S$
10400 FOR=1 TO S-1
10410 IF CODE (R$(L))>64 THEN LET
      S$=S$+R$(L)
10420 NEXT L
10430 REM QUITAR SOBRANTES
10440 REM ALMACENAR APELLIDOS EN A$
10450 FOR L=S+1 TO LEN (R$)
10460 IF CODE(R$(L))>64 THEN LET
      A$=A$+R$(L)
10470 NEXT L
10480 LET M$(SIZE)=A$+"  "+S$
10490 LET S$="":LET A$=""
10510 RETURN
```

N.B. Debido a la forma en que el Spectrum trata las series, la rutina anterior divide el nombre en el primer espacio, no en el último.

```
12000 REM *GRDREG* SR
12030 SAVE "NCAM" DATA N$( )
12040 SAVE "MCAM" DATA M$( )
12050 SAVE "CCAM" DATA C$( )
12060 SAVE "DCAM" DATA D$( )
12070 SAVE "PCAM" DATA P$( )
12080 SAVE "TELCAM" DATA T$( )
12090 SAVE "INDCAM" DATA X$( )
.
12150 RETURN
.
12500 REM *FINARCH* SR
12510 LET TAMA=50
12520 FOR L=1 TO 50
12530 IF N$(L)=B$ THEN LET
      TAMA=L:LET L=50
12540 NEXT L
12560 RETURN
```

LYNX

El ordenador Lynx no admite la orden STR$. Véase "Complementos al BASIC", p. 357. Para dimensionar matrices mediante la orden DIM, véase "Complementos al BASIC", p. 275.

EOF

En los ordenadores Commodore 64 y Vic-20, reemplazar la línea 1520 por:

```
1520 IF ST AND 64 THEN LET L=50
```

En el Dragon 32, suprimir la línea 1520 y sustituirla por:

```
1485 IF EOF(-1) THEN GOTO 1510
```

En el BBC Micro, reemplazarla por:

```
1520 IF EOF#X THEN LET L=50
```

donde X es la variable numérica empleada en la orden OPENOUT (véase p. 319).

OPEN CLOSE

Véase p. 319.

```
10 REM    'PROPRI'
20 REM    #INICIL#
30 GOSUB  1000
40 REM    #PRESEN#
50 GOSUB  3000
60 REM    #ELECCN#
70 GOSUB  3500
80 REM    #EJECUT#
90 GOSUB  4000
100 IF OPCN <> 9 THEN 60
110 END
1000 REM  SUBRUTINA #INICIL#
1010 GOSUB 1100: REM SUBRUTINA #CREMAT# (CREAR
     MATRICES)
1020 GOSUB 1400: REM SUBRUTINA #LEARCH# (LEER ARCHIVOS)
1030 GOSUB 1600: REM SUBRUTINA #ESBAND# (ESTABLECER
     BANDERAS)
1040 REM
1050 REM
1060 REM
1070 REM
1080 REM
1090 RETURN
1100 REM SUBRUTINA #CREMAT# (CREAR MATRICES)
1110 DIM NOMCAM$ (50)
1120 DIM MODCAM$ (50)
1130 DIM CLLCAM$ (50)
1140 DIM CIUCAM$ (50)
1150 DIM PROCAM$ (50)
1160 DIM TELCAM$ (50)
1170 DIM INDCAM$ (50)
1180 REM
1190 REM
1200 REM
1210 LET TAMA = 0
1220 LET RMOD = 0
1230 LET SVED = 0
1240 LET CURS = 0
1250 REM
1260 REM
1270 REM
1280 REM
1290 REM
1300 RETURN
1400 REM SUBRUTINA #LEARCH#
1410 OPEN "I",#1, "DAT.AGCO"
1420 INPUT #1, TEST$
1430 IF TEST$ = "@VACIO" THEN GOTO 1530: REM CERRAR Y
     RETORNAR
1440 LET NOMCAM$ (1) = TEST$
1450 INPUT #1,MODCAM$ (1), CLLCAM$ (1), CIUCAM$ (1),
     PROCAM$ (1), TELCAM$ (1)
1460 INPUT #1, INDCAM$ (1)
1470 LET TAMA = 2
```

```
1480 FOR L = 2 TO 50
1490 INPUT #1, NOMCAM$ (L), MODCAM$ (L), CLLCAM$ (L),
     CIUCAM$ (L), PROCAM$ (L)
1500 INPUT #1, TELCAM$ (L), INDCAM$ (L)
1510 LET TAMA = TAMA + 1
1520 IF EOF(1) = -1 THEN LET L = 50
1530 NEXT L
1540 CLOSE #1
1550 RETURN
1600 REM SUBRUTINA #ESBAND#
1610 REM ESTABLECE BANDERAS DESPUES DE #LEARCH#
1620 REM
1630 REM
1640 IF TEST$ = "@VACIO" THEN LET TAMA = 1
1650 REM
1660 REM
1670 REM
1680 REM
1690 RETURN
3000 REM SUBRUTINA #PRESEN#
3010 PRINT CHR$ (12):REM LIMPIAR PANTALLA
3020 PRINT
3030 PRINT
3040 PRINT
3050 PRINT
3060 PRINT TAB(11); "#BIEN VENIDO A LA#"
3070 PRINT TAB(9); "#AGENDA COMPUTERIZADA#"
3080 PRINT TAB(12); "#DE MI COMPUTER#"
3090 PRINT
3100 PRINT TAB(0); "(PULSE BARRA ESPACIADORA PARA CONTINUAR)
3110 FOR L = 1 TO 1
3120 IF INKEY$ <> " " THEN L = 0
3130 NEXT L
3140 PRINT CHR$(12)
3150 RETURN
3500 REM SUBRUTINA #ELECCN#
3510 REM
3520 IF TEST$ = "@VACIO" THEN GOSUB 3860: REM SUBRUTINA
     #PRIMERA#
3530 IF TEST$ = "@VACIO" THEN RETURN
3540 REM 'IMMENU'
3550 PRINT CHR$(12)
3560 PRINT "SELECCIONE UNO DE LOS SIGUIENTES"
3570 PRINT
3580 PRINT
3590 PRINT
3600 PRINT "1. HALLAR REGISTRO (DE NOMBRE)"
3610 PRINT "2. HALLAR NOMBRES (DE NOMBRE INCOMPLETO)"
3620 PRINT "3. HALLAR REGISTRO (DE CIUDAD)"
3630 PRINT "4. HALLAR REGISTRO (DE INICIAL)"
3640 PRINT "5. LISTAR TODOS LOS REGISTROS"
3650 PRINT "6. AGREGAR REGISTRO NUEVO"
3660 PRINT "7. MODIFICAR REGISTRO"
3670 PRINT "8. BORRAR REGISTRO"
3680 PRINT "9. SALIR Y GUARDAR"
3690 PRINT
3700 PRINT
3710 REM 'ASOPCN'
3720 REM
3730 LET L = 0
3740 LET I = 0
3750 FOR L = 1 TO 1
3760 PRINT "DE ENTRADA A OPCION (1-9)"
3770 FOR I = 1 TO 1
3780 LET A$ = INKEY$
3790 IF A$ = "" THEN I = 0
3800 NEXT I
3810 LET OPCN = VAL(A$)
3820 IF OPCN <1 THEN L = 0
3830 IF OPCN >9 THEN L = 0
3840 NEXT L
3850 RETURN
3860 REM SUBRUTINA #PRIMERA# (VISUALIZAR MENSAJE)
3870 LET OPCN = 6
3880 PRINT CHR$(12)= REM LIMPIAR PANTALLA
3890 PRINT
3900 PRINT TAB(10); "NO HAY REGISTROS EN"
3910 PRINT TAB(7); "EL ARCHIVO. DEBERA EMPEZAR"
3920 PRINT TAB(8); "POR AGREGAR UN REGISTRO"
3930 PRINT
3940 PRINT TAB(0); "PULSE BARRA ESPACIADORA PARA CONTINUAR"
3950 FOR B = 1 TO 1
3960 IF INKEY$ <> " " THEN B = 0
3970 NEXT B
3980 PRINT CHR$(12): REM LIMPIAR PANTALLA
3990 RETURN
4000 REM SUBRUTINA #EJECUT#
4010 REM
4020 IF OPCN = 6 THEN GOSUB 10000
4030 REM
4040 REM  1 ES #ENCREG#
4050 REM  2 ES #ENCNOM#
4060 REM  3 ES #ENCIUD#
4070 REM  4 ES #ENCINI#
4080 REM  5 ES #LISREG#
4090 IF OPCN = 6 THEN GOSUB 10000
4100 REM  7 ES #MODREG#
4110 REM  8 ES #BORREG#
4120 IF OPCN = 9 THEN GOSUB 11000
4130 REM
4140 RETURN
10000 REM SUBRUTINA #INCREG#
10010 PRINT CHR$(12): REM LIMPIAR PANTALLA
10020 INPUT "DE ENTRADA AL NOMBRE"; NOMCAM$ (TAMA)
10030 INPUT "DE ENTRADA A LA CALLE"; CLLCAM$ (TAMA)
10040 INPUT "DE ENTRADA A LA CIUDAD"; CIUCAM$ (TAMA)
10050 INPUT "DE ENTRADA A LA PROVINCIA"; PROCAM$ (TAMA)
10060 INPUT "DE ENTRADA AL NUMERO DE TELEFONO"; TELCAM$
      (TAMA)
10070 LET RMOD = 1: REM ESTABLECIDA BANDERA 'REGISTRO
      MODIFICADO
10080 LET INDCAM$ (TAMA) = STR$ (TAMA)
10090 LET TEST$ = ""
10100 GOSUB 10200: REM #MODNOM#
10110 LET OPCN = 0
10120 LET TAMA = TAMA +1
10130 REM
10140 REM
10150 RETURN
10200 REM RUTINA #MODNOM#
10210 REM CONVIERTE CONTENIDO DE NOMCAM$ A MAYUSCULAS,
10220 REM QUITA SOBRANTE Y ALMACENA EN EL ORDEN:
10230 REM APELLIDO+ESPACIO+NOMBRE DE PILA EN MODCAM$
10240 REM
10250 LET N$ = NOMCAM$ (TAMA)
10260 FOR L = 1 TO LEN(N$)
10270 LET TEMP$ = MID$(N$,L,1)
10280 LET T = ASC(TEMP$)
10290 IF T >= 97 THEN T = T - 32
10300 LET TEMP$ = CHR$(T)
10310 LET P$ = P$ + TEMP$
10320 NEXT L
10330 LET N$ = P$
10340 REM LOCALIZAR ULTIMO ESPACIO
10350 FOR L = 1 TO LEN(N$)
10360 IF MID$(N$,L,1) = " " THEN S = L
10370 NEXT L
10380 REM QUITAR SOBRANTE Y ALMACENAR NOMBRE DE PILA
10390 REM EN CNOM$
10400 FOR L = 1 TO S - 1
10410 IF ASC(MID$(N$,L,1)) > 64 THEN CNOM$ = CNOM$ +
      MID$(N$,L,1)
10420 NEXT L
10430 REM QUITAR SOBRANTE Y ALMACENAR APELLIDO
10440 REM EN SNOM$
10450 FOR L = S + 1 TO LEN(N$)
10460 IF ASC(MID$(N$,L,1)) > 64 THEN SNOM$ = SNOM$ +
      MID$(N$,L,1)
10470 NEXT L
10480 LET MODCAM$ (TAMA) = SNOM$ + " " + CNOM$
10490 LET P$ = "": LET N$ = "": LET SNOM$ = "": LET
      CNOM$ = ""
10500 LET P$ = "": LET N$ = "": LET SNOM$ = "": LET
      CNOM$ = ""
10510 RETURN
11000 REM SUBRUTINA #SAPROG#
11010 REM CLASIFICA Y GUARDA ARCHIVO
11020 REM SI ALGUN REGISTRO HA SIDO
11030 REM MODIFICADO (RMOD = 1)
11040 REM
11050 IF RMOD = 0 THEN RETURN
11060 REM
11070 GOSUB 11200: REM #CLSREG#
11080 REM
11090 GOSUB 12000: REM #GRDREG#
11100 RETURN
11200 REM SUBRUTINA #CLSREG#
11210 REM CLASIFICA TODOS LOS REGISTROS EN ORDEN
      ALFABETICO
11220 REM MEDIANTE MODCAM$ Y ACTUALIZA INDCAM$
11230 REM
11240 REM
11250 LET S = 0
11260 FOR L = 1 TO TAMA - 2
11270 IF MODCAM$(L) > MODCAM$(L + 1) THEN GOSUB 11350
11280 NEXT L
11290 IF S = 1 THEN 11250
11300 REM
11310 REM
11320 LET RMOD = 0: REM LIMPIA BANDERA 'REGISTRO
      MODIFICADO'
11330 REM
11340 RETURN
11350 REM SUBRUTINA "INTERCALAR"
11360 LET TNOMC$ = NOMCAM$(L)
11370 LET TMODC$ = MODCAM$(L)
11380 LET TCLLC$ = CLLCAM$(L)
11390 LET TCIUC$ = CIUCAM$(L)
11400 LET TPROC$ = PROCAM$(L)
11410 LET TTELC$ = TELCAM$(L)
11420 REM
11430 LET NOMCAM$(L) = NOMCAM$(L + 1)
11440 LET MODCAM$(L) = MODCAM$(L + 1)
11450 LET CLLCAM$(L) = CLLCAM$(L + 1)
11460 LET CIUCAM$(L) = CIUCAM$(L + 1)
11470 LET PROCAM$(L) = PROCAM$(L + 1)
11480 LET TELCAM$(L) = TELCAM$(L + 1)
11490 LET INDCAM$(L) = INDCAM$(L + 1)
11500 REM
11510 LET NOMCAM$(L + 1) = TNOMC$
11520 LET MODCAM$(L + 1) = TMODC$
11530 LET CLLCAM$(L + 1) = TCLLC$
11540 LET CIUCAM$(L + 1) = TCIUC$
11550 LET PROCAM$(L + 1) = TPROC$
11560 LET TELCAM$(L + 1) = TTELC$
11570 LET INDCAM$(L + 1) = STR$(L + 1)
11580 LET S = 1
11590 REM
11600 RETURN
12000 REM SUBRUTINA #GRDREG#
12010 REM
12020 REM
12030 OPEN "O",#1, "DAT.AGCO"
12040 REM
12050 FOR L = 1 TO TAMA - 1
12060 PRINT #1, NOMCAM$(L);","; MODCAM$(L);","; CLLCAM$(L);
      ","; CIUCAM$(L)
12070 PRINT #1, PROCAM$(L);","; TELCAM$(L);",";
      INDCAM$(L);
12080 NEXT L
12090 REM
12100 REM
12110 REM
12120 REM
12130 CLOSE #1
12140 REM
12150 RETURN
12500 REM SUBRUTINA #FINARCH#
12510 IF NOMCAM$(L) = "" THEN LET L = 50
12520 IF NOMCAM$(L) = "" THEN RETURN
12530 LET TAMA = TAMA + 1
12540 REM
12550 REM
12560 RETURN
```

Vannevar Bush

El analizador diferencial
Esta máquina fue diseñada para resolver un tipo importante de funciones matemáticas, que se presentan en numerosas áreas de la ciencia y de la ingeniería, conocidas como ecuaciones diferenciales de segundo grado. El método había sido sugerido por primera vez por lord Kelvin y hacía referencia a la alimentación de la salida de un "integrador" (un dispositivo que calcula con precisión el área de una superficie irregular) conectada a la entrada de otro. Sin embargo, la potencia de salida era demasiado débil y no se podía utilizar como entrada; este método no pudo ser aplicado hasta que fueron inventados los amplificadores.
La máquina que Bush construyó en 1931 era totalmente mecánica y constaba de una compleja estructura de engranajes, ejes y motores eléctricos. La entrada y la salida se producían mediante giro de ejes y el problema de feedback fue solventado mediante un amplificador de par.
En la década de los cuarenta, se construyó un analizador diferencial más avanzado utilizando componentes eléctricos, pero la máquina pesaba más de cien toneladas. La salida se producía mediante cinco registros de tipo digital y tenía una precisión de uno a 10 000. Las condiciones iniciales y los parámetros de control se suministraban en cinta de papel perforado

El analizador diferencial de Bush era una calculadora electromecánica que resolvía ecuaciones diferenciales

Muchas personas sostienen la teoría de que el norteamericano Vannevar Bush fue el padre del ordenador. Su contribución más importante al desarrollo de la ciencia informática tuvo lugar en 1931, cuando creó un analizador diferencial mecánico que constituyó el punto de partida de una serie de investigaciones que finalmente conducirían al desarrollo del ordenador digital.

Bush nació cerca de Boston (Massachusetts) el 11 de marzo de 1890 y, siguiendo los pasos de su padre, estudió la carrera de ingeniería. Tras graduarse en 1913, trabajó durante un corto período para la General Electric, antes de aceptar el cargo de profesor adjunto en su antigua escuela. Luego realizó estudios de posgrado en la Universidad de Harvard y en el Massachusetts Institute of Technology (MIT). Durante la primera guerra mundial colaboró con las fuerzas armadas norteamericanas en la creación y el posterior desarrollo de un detector de submarinos.

Cuando Bush ideó y realizó su primer invento, un dispositivo topográfico, aún era un estudiante. El mecanismo, que estaba suspendido entre dos ruedas de bicicleta, calculaba la altura del suelo sobre el cual se desplazaba y ofrecía una representación en forma gráfica del perfil del terreno. Incorporaba también un dispositivo que hacía las funciones de un integrador, puesto que la determinación de la altura de una posición cualquiera requería el conocimiento de todos los valores previos.

En el MIT, Bush fue profesor de transmisión de energía eléctrica, y empezó las investigaciones sobre uno de los principales problemas concernientes al suministro de electricidad: cómo evitar los cortocircuitos que se producen como resultado de un incremento de la demanda de energía en un momento determinado. Las ecuaciones matemáticas que rigen una situación de este tipo habían sido descubiertas a finales del siglo pasado por el científico escocés James Clerk Maxwell (1831-1879). Pero el problema comportaba tal número de ecuaciones simultáneas que era imposible de resolver manualmente, y por eso Bush empezó sus trabajos sobre el diseño de una máquina que llevara a efecto estos complicados cálculos. Se inspiró también en los trabajos de lord Kelvin (1824-1907), autor de una máquina para resolver las ecuaciones matemáticas relativas a la predicción de las mareas.

A principios de la década de los veinte, Bush construyó su primera máquina, a la que dio el nombre de "producto telegráfico". Esta máquina permitía que un operador humano trazara las trayectorias dibujadas en un gráfico (utilizando un potenciómetro: dispositivo que transforma la medida de una posición en un voltaje). Luego, con estas señales eléctricas, se alimentaba un vatímetro especialmente diseñado: el disco giratorio que figura en todos los contadores de consumo de energía eléctrica, el cual registra la cantidad de vatios consumidos mediante la integración de los valores fluctuantes de intensidad y voltaje para dar el "producto".

El éxito de esta máquina en la resolución de un conjunto de ecuaciones simultáneas sugirió la posibilidad de construir un dispositivo que fuera capaz de resolver ecuaciones diferenciales de segundo grado e incluso más difíciles. Las posteriores investigaciones de Bush en este campo condujeron a concluir el primer analizador diferencial en 1931. La máquina constituyó un éxito completo, construyéndose diversas unidades en Gran Bretaña y el resto en Europa. En Norteamérica, la Escuela Moore de Ingeniería Eléctrica de la Universidad de Pennsylvania —que posteriormente construiría el ordenador ENIAC (véase p. 88)— encargó una de ellas. Cuando el "producto telegráfico" de Bush tenía un margen de error del 2 %, el analizador diferencial proporcionaba resultados con una precisión del 99,95 %. Sin embargo, el coste en la mejora de la fiabilidad de este tipo de dispositivos mecánicos se multiplicaba por diez por cada decimal adicional. Con el desarrollo del ordenador digital, en cambio, el coste de una máquina, para un incremento similar en la fiabilidad, sólo se duplicaba.

Bush llegó a ser decano de la escuela de ingeniería y vicepresidente del Instituto Carnegie en 1939; su habilidad en la administración del fondo para la investigación científica que legara el millonario Carnegie dio como resultado que fuera nombrado, al año siguiente, presidente del Comité de Investigaciones para la Defensa Nacional. En este cargo, Bush fue responsable de las investigaciones del Ejército norteamericano durante la segunda guerra mundial y, en particular, influyó para que se autorizara el proyecto Manhattan, que conduciría a la creación de la bomba atómica. Se jubiló en 1955 y se dedicó a sus hobbies. Murió en 1974.

Tony Stead

Mentes jóvenes
Según parece, los niños son mucho más receptivos a la nueva tecnología que la mayoría de los adultos, quienes experimentan una reacción negativa ante la idea de tener que aprender ideas nuevas. La versatilidad del microprocesador significa que virtualmente no existe ningún límite mínimo de edad para los juguetes electrónicos y los dispositivos educativos

Para niños

Los más recientes juguetes educativos poseen un potencial de procesamiento equiparable al de un ordenador personal y utilizan técnicas de programación similares

Además de constituir el corazón de todo microordenador, el microprocesador se ha convertido en una configuración estándar de muchos aparatos domésticos, como máquinas de coser, lavadoras e incluso cerraduras para puertas. Los fabricantes de juguetes, asimismo, han hecho experimentos con microprocesadores, especialmente para el control de prototipos de coches y de trenes. Sin embargo, al menos una firma de ordenadores (Texas Instruments) ha descubierto que la producción de juguetes didácticos basados en microprocesadores es muy rentable. La primera incursión de TI en este mercado fue una unidad parecida a una calculadora que planteaba problemas de aritmética elemental. *Little professor* (Pequeño profesor) consiguió una gran popularidad y, aunque haya sido sustituido por *Speak and maths* (Lenguaje y matemáticas), todavía se vende en cantidades significativas.

Speak and maths fue el segundo juguete didáctico de Texas Instruments que empleó el chip TI para síntesis de voz. Éste también se utilizó en el ordenador personal TI99/4A, que se retiró del mercado a fines de 1983, cuando Texas Instruments decidió abandonar la informática doméstica de bajo coste. *Speak and spell* (Hable y deletree), que se lanzó en 1978, posee un vocabulario de algunos cientos de palabras. La unidad posee un teclado alfabético completo (con algunas teclas adicionales) compuesto a partir de una membrana de capas múltiples similar a la del ZX81 de Sinclair. Al pulsar la tecla ENQUIRY (pregunta) al usuario se le solicita de forma audible que deletree una palabra. Cada tecla que se digita se visualiza mediante diodos emisores de luz hasta que se completa la palabra. *Speak and spell* le dice entonces de forma audible al usuario si la ha deletreado de la manera correcta o no.

Speak and maths funciona de modo parecido, pero plantea problemas de aritmética.

Un tercer juguete parlante de TI, *Touch and tell* (Toque y diga), tal vez sea más recreativo que educativo. Utiliza una serie de capas plásticas, cada una de las cuales lleva impreso un patrón o imagen y se identifica por una codificación magnética. Al tocar el niño una zona de la imagen, *Touch and tell* identifica de forma audible el objeto seleccionado.

Mientras que la síntesis de voz es con mucho la técnica de informática más sofisticada que utilizan los fabricantes de juegos y de juguetes, la aplicación más popular son las versiones reducidas de algunos de los juegos recreativos más populares. Deben de existir tantas variedades de este tipo de juegos como juegos recreativos propiamente dichos. Otra área en la que el microordenador ha causado impacto en el mercado del juguete es la de los coches y camiones autodirigidos. Quizá el más conocido de éstos sea el *Big Trak* (véase p. 36), que se programa dando entrada a las instrucciones en un teclado montado en su superficie superior. El juguete se parece a una "tortuga" (véase p. 34), y se puede controlar desde un microordenador a través de su puerta en paralelo.

Otros juguetes basados en microprocesador son: el *Simon*, que invita al niño a repetir una frase musical al azar con luces intermitentes; *Playskool's maximus*, un "maestro" de aritmética similar al *Little professor*, y diversos robots. Entre los diseñados para niños mayores (y para adultos) podemos incluir: *Electroni-kit, Mykit systems* y *Radionics*, que, como sus nombres sugieren, son juguetes electrónicos desmontables que utilizan componentes encerrados en cápsulas, que, una vez armados, se pueden enchufar en un tablero base para formar diversos dispositivos sencillos.

Electroni-kit
Tal como su nombre indica, *Electroni-kit* es un juego para armar que se utiliza para crear dispositivos electrónicos. Sus componentes vienen encerrados en cápsulas de plástico transparente y se enchufan en un tablero base (siguiendo unos diagramas esquematizados que vienen con el juego) para construir el dispositivo deseado. El modelo más sofisticado de la gama incluye los componentes de un microordenador elemental, diseñado para enseñar operaciones muy sencillas en código de lenguaje máquina. Pero con sólo 96 bytes de memoria, no se le puede considerar un ordenador personal

Ian McKinnell

Cortesía de Personal Computer World

Maximus
Éste es un producto muy avanzado de MB Electronics. Aunque se parece a una calculadora, en realidad se trata de otro juego de "emparejamiento", como el *Simon*. El *Maximus*, sin embargo, posee capacidad para operar en diversas modalidades, lo que permite el emparejamiento de notas musicales, imágenes, ritmo, deletreo y formas

COURTESY OF MILTON BRADLEY LTD

Simon
El *Simon* de MB Electronics es una versión en microprocesador de un juego infantil al aire libre. La unidad genera una frase de notas musicales, cada una de ellas acompañada de una luz intermitente. Los cuatro cuadrantes de colores de superficie actúan como interruptores tanto para las luces como para los tonos. El objeto del juego es reproducir exactamente la frase musical

Texas Instruments
TI se introdujo en el mercado educativo a mediados de los años setenta con el *Little professor*, un dispositivo parecido a una calculadora que planteaba problemas de aritmética. Antes de que finalizara la década, Texas Instruments había empezado a comercializar un tutor para deletrear que utilizaba el chip TI para síntesis de voz. Pulsando una tecla, *Speak and spell* (Hable y deletree) solicita que se deletree una palabra. Más recientemente, estas técnicas se han aplicado a juegos de aritmética simple y a dispositivos muy elementales que narran cuentos a los niños más pequeños

Cortesía de Texas Instruments

Robo-1
El *Robo-1*, de la firma Tomy, es un brazo-robot de diseño convencional que se controla mediante dos palancas de mando. Es incapaz de operar bajo control por programa. Lo más sorprendente del *Robo-1* es su precio: menos del 10 % del precio del brazo-robot didáctico más barato (véase p. 314). Por supuesto, la construcción es mucho menos sólida, ya que no es de metal laminado sino de plástico. El brazo cuenta con la alimentación y el control visuales del usuario, en vez de utilizar motores de precisión a impulsos. Con algo de habilidad, el *Robo-1* se puede conectar a un ordenador personal

Robo-1, cortesía de Hamleys

Big Trak
Aunque por su aspecto se asemeja a algunos de esos vehículos de juguete resistentes para los niños más pequeños, el *Big Trak* es en realidad un robot móvil encubierto. Totalmente autocontenido, se programa dando entrada a los códigos de dirección y distancia en un teclado dispuesto sobre su superficie. Con un poco de habilidad se puede conectar el *Big Trak* a un microordenador personal por medio de una puerta en serie o en paralelo. El vehículo se podría guiar entonces bajo control de programa, lo cual abriría la posibilidad de bifurcar a un subprograma diferente en caso de que hubiera que hacer frente a determinadas condiciones

Cortesía de Milton Bradley Ltd

Ian McKinnell

Verificadores de ortografía

Muchos procesadores de textos disponen de programas verificadores de ortografía, y también están empezando a aparecer otros que se encargan del estilo y de la gramática

Aún está muy lejano el día en que los diseñadores de ordenadores puedan crear máquinas que posean capacidad de generar y manipular idiomas humanos. Una de las aplicaciones proyectadas para la quinta generación de ordenadores, que debería aparecer en el transcurso de la década de 1990, es la traducción mecánica entre, por ejemplo, el inglés y el japonés. Ya hay facilidades para la traducción mecánica de una prosa relativamente sencilla, como la de los expedientes e informes oficiales, si bien los borradores que producen los ordenadores de unidad principal invariablemente se han de corregir y pulir a mano. Las anécdotas sobre errores abundan: se cuenta que el dicho "El espíritu está pronto, pero la carne es débil", se tradujo del inglés al ruso y nuevamente al inglés mediante dos programas diferentes. El resultado final fue: ¡"El vino es agradable, pero la carne se echó a perder"!

Las anécdotas apócrifas de este tipo sirven para ilustrar un punto importante: las dificultades que surgen cuando un ordenador está procesando datos sin entender lo que éstos significan. A los estudiantes de informática se les suele plantear el problema de considerar cómo podría distinguir un ordenador los significados de estas oraciones en inglés:

TIME FLIES LIKE AN ARROW
(El tiempo vuela como una saeta)

FRUIT FLIES LIKE A BANANA
(A las moscas de la fruta les gusta el plátano)

La construcción de ambas oraciones es idéntica, pero en el primer caso FLIES es un verbo ("vuela"), mientras que en el segundo forma parte de un predicado nominal ("moscas"). Nosotros podemos reconocerlas como distintas a través de la experiencia. En un ordenador, la experiencia se puede simular, disponiendo de la suficiente memoria, pero esto cae en el campo de la inteligencia artificial, y la investigación en esta área no está muy avanzada. En realidad estamos aludiendo a la diferencia entre "sintaxis" y "semántica". La sintaxis, que son las reglas relativas a los procesos de construcción que se utilizan en un idioma, es algo que los ordenadores pueden manejar con bastante facilidad, como bien lo saben todos los programadores de ordenadores que se hayan encontrado con mensajes de SYNTAX ERROR? (¿error de sintaxis?). La semántica, en cambio, se refiere al significado que comunican aquellas frases y construcciones.

En los años cincuenta, Noam Chomsky desarrolló las bases de la teoría actual sobre los lenguajes humanos y las reglas de la gramática, y aunque no tuvo una relación directa con las ciencias de la informática, sus teorías son perfectamente aplicables tanto a la traducción mecánica como a la escritura de intérpretes y compiladores para lenguajes de programación.

Consecuencia directa de las investigaciones del lingüista norteamericano ha sido la creación de diversas herramientas de software para ayudar a la escritura de texto. Además de los paquetes para tratamiento de textos, auxiliares de la creación, la edición y la impresión de texto, hay programas para la corrección de documentos, para la detección de errores de ortografía y mecanografía, e incluso para verificar la gramática y el estilo de la escritura.

Todos los programas verificadores de ortografía se valen de un diccionario retenido en disco, que suele almacenar entre 25 000 y 50 000 palabras. La mayoría de los paquetes permiten que el usuario incorpore al diccionario nuevas entradas.

Constituye un problema, sin embargo, hallar el espacio de memoria adecuado para un diccionario completo. Se sabe que un byte de ocho bits puede retener un solo carácter alfanumérico empleando el código ASCII. Por consiguiente, aun concediendo un promedio muy optimista de apenas cinco caracteres por palabra, un diccionario de 30 000 entradas requeriría 150 Kbytes de almacenamiento, que es mucho más de lo que permiten la mayoría de las unidades de disco para ordenadores personales. Por suerte, se puede comprimir este tipo de datos mediante la utilización de dos técnicas.

La primera da por supuesto que el diccionario sólo incluirá letras en minúscula (una rutina del programa de ortografía manipulará las conversiones), y que no se necesitarán dígitos numéricos ni ciertos signos de puntuación, por lo que todos éstos se pueden eliminar del programa. En consecuencia, podríamos construir nuestro diccionario utilizando un máximo de 32 caracteres diferentes, en lugar de la gama ASCII completa de 128 (o 256, incluyendo los símbolos para gráficos). Por lo tanto, podríamos reducir las necesidades de almacenamiento de cada carácter de ocho bits a cinco. La palabra "computer", por ejemplo, se podría almacenar en un total de 40 bits, o cinco bytes. Los cinco primeros bits del primer byte especificarían la letra "c", y los tres bits siguientes, más los dos primeros del segundo byte, especificarían "o", y así sucesivamente.

La segunda técnica que se emplea en los verificadores de ortografía parte de la premisa de que ciertas combinaciones de caracteres aparecen con tanta frecuencia que se podrían representar en un único byte. Éste se señalaría de alguna forma mediante una "bandera" para indicar que se trata de un distintivo común para un grupo de caracteres y no de un solo carácter. Casi con toda seguridad su orde-

nador personal utiliza esta técnica en BASIC: cada palabra clave, como PRINTo NEXT, está almacenada en RAM en un solo byte para ahorrar espacio.

En un diccionario verificador de ortografía, este segundo sistema se emplea al comienzo de las palabras. Consideremos, por ejemplo, el gran conjunto de palabras que incluyen prefijos como "re", "in", "des" o "auto". El paquete VizaSpell, que se ejecuta con el procesador de textos VizaWrite en el Commodore 64, utiliza ambas técnicas para introducir un diccionario de 30 000 palabras en sólo 65 Kbytes de disco.

El trabajo que más dificultades entraña a un verificador de ortografía es, sin embargo, el de buscar en el diccionario todas las palabras que componen el documento. Se podría emplear una búsqueda binaria (véase p. 416), pero, tratándose de un documento de mil palabras, esto ocuparía horas. Lo ideal sería que el procesador de textos verificara cada palabra a medida que ésta se digitara, pero esto no es práctico en términos de programación y, en consecuencia, un documento por lo general se deberá verificar como un todo, ya sea en disco o (como en las máquinas mayores) en RAM. El programa funciona a lo largo del documento y compila en orden alfabético una lista de las palabras que contiene. A menudo, más del 50 % de un extenso informe consta de sólo 100 palabras diferentes.

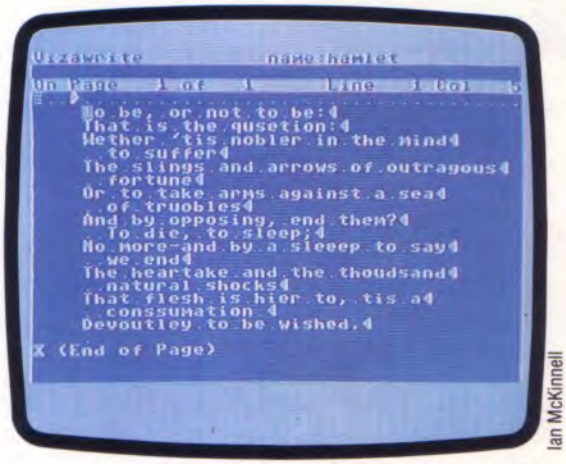
La mayoría de los verificadores de ortografía emplean este sistema para proporcionar un útil informe adicional acerca del empleo de palabras en su documento, lo que puede resultar de gran ayuda para detectar repeticiones innecesarias. Después, un algoritmo sencillo repasa esta lista y la lista del diccionario simultáneamente, buscando los emparejamientos. De este modo, el tiempo que lleva completar la búsqueda se reduce mucho y es constante: cuatro minutos en el caso del VizaSpell, independientemente de la longitud del documento.

Las palabras que no se encuentran en el diccionario se imprimen en forma de lista o bien se señalan en el mismo documento. Ante cada palabra señalada al usuario se le presentan tres opciones:

1) En la palabra hay un error de ortografía o de mecanografía que se debe corregir;
2) La palabra es correcta y debería agregarse al diccionario del programa;
3) La palabra es correcta, pero es poco probable que se vuelva a emplear en otra ocasión (p. ej., forma parte de unas señas), de modo que se debe dejar así, sin incorporarla al diccionario.

Los verificadores de gramática y de estilo funcionan de manera similar. Los primeros trabajan de acuerdo con una cantidad limitada de reglas (como cuidar que haya una letra mayúscula al comienzo de cada oración) y, por consiguiente, hay muchos errores gramaticales que no se pueden detectar. Los verificadores de estilo están aún en una etapa incipiente, y la mayoría de los paquetes habituales utilizan tan sólo un gran diccionario de ejemplos con el fin de identificar la sintaxis y las expresiones erróneas.

Popperfoto



Ian McKinnell

Escribir en BASIC un tipo sencillo de verificador de ortografía, de gramática o de estilo puede ser un ejercicio muy interesante incluso para un programador poco experimentado, aunque para ello se necesitará tener un conocimiento bastante profundo de las funciones de la máquina para manipulación de series. Al aumentar la sofisticación del software es de esperar que los paquetes para tratamiento de textos vengan con dichas funciones ya incorporadas. Desde luego, a cualquier escritor le encantaría: 'ORDEN > REDACTAR ARTICULO, LONGITUD 1200 PALABRAS, EMPEZAR'.

"Ser o no ser"
¡Imagínese cuánto más fácil sería la asignatura de Lengua y Literatura si se pudieran usar programas verificadores de ortografía en el aula de exámenes! Podemos utilizar el monólogo de *Hamlet*, de Shakespeare, para ilustrar cómo funciona uno de los programas de este tipo (el VizaSpell). Primero se digita el texto en el ordenador empleando un procesador de textos. Luego se llama al verificador de ortografía mediante un par de órdenes simples, y éste crea, en orden alfabético, una lista de todas las palabras utilizadas, incluyendo también su frecuencia de uso. Esta lista se coteja con el diccionario en disco, y se destacan los términos que no se reconocen. Cuando se lo emplea por primera vez, es probable que el programa "ilumine" algunas palabras aparentemente comunes, que se pueden incorporar al diccionario para una utilización posterior

Programas sofisticados

Los generadores de aplicaciones son parecidos a los generadores automáticos de programas, pero además de las aplicaciones de gestión poseen también aplicaciones para juegos

IAN McKINNELL

Pinball Construction Set
Este paquete es un tipo de generador de aplicaciones para juegos. El usuario diseña el trazado y la lógica para un juego de "millón" utilizando un menú de objetos y diversas herramientas representadas gráficamente para fijarlas sobre el tablero

En un capítulo anterior de MI COMPUTER analizamos una clase de programa para ordenador que, tras dar el usuario un conjunto de especificaciones, produce un programa capaz de llevar a cabo la aplicación requerida. Dichos generadores de programas se pueden adquirir para la mayoría de los micros de gestión y existen algunos paquetes para ordenadores personales, aunque el tipo de aplicaciones para los que son idóneos requiere al menos una unidad de disco.

Una forma mucho más común de generar programas para requerimientos específicos implica la utilización de paquetes denominados *generadores de aplicaciones*. A diferencia de los generadores de programas, éstos producen programas que no son autónomos sino que necesitan el paquete generador de aplicaciones original para poder ser ejecutados. Consideremos la creación de un programa para manipular la facturación con utilización de estos dos tipos de generadores, para ilustrar las diferencias entre ambos.

Si fuéramos a utilizar un generador de programas, primero el software se habría de cargar desde el disco al ordenador. Una vez el usuario hubiera respondido a todas las preguntas relativas a archivos, registros, campos, relaciones matemáticas, trazados de pantalla e informes impresos requeridos (es decir, una vez que hubiera especificado el programa de aplicaciones deseado), el generador le pediría que insertara en la unidad de disco un disco vacío. Entonces guardaría el nuevo programa que habría generado en este segundo disco. Este proceso se podría repetir y se podría hacer una copia del programa de facturación para cada sucursal de la empresa.

En comparación, un generador de aplicaciones parece inicialmente menos satisfactorio. Una vez cumplida la etapa de especificaciones, las rutinas necesarias se graban en el mismo disco que el generador. También podría grabar el programa en un disco separado, pero lo haría de forma tal que aún se necesitaría el disco del generador original para poder ejecutar la aplicación. Aunque se podría utilizar una única copia del paquete original para producir un número ilimitado de aplicaciones diferentes, se desprende que todas ellas se deben utilizar en la misma posición física que el disco del generador. Si usted quisiera poner su aplicación a disposición de otros usuarios, éstos necesitarían adquirir asimismo una copia del generador. Por supuesto, los generadores de este tipo emplean diversos métodos para proteger el programa, con el fin de que resulte muy difícil el poder copiarlo sin la debida autorización.

Un generador de aplicaciones es en realidad un sofisticado programa para fines generales. Cuando el usuario especifica su aplicación, sencillamente está asignando valores a una cantidad de variables importantes dentro del generador, denominadas *parámetros*. Éstos controlan el flujo del programa, la estructura de los datos y los trazados para la pantalla y la impresora. Cuando la aplicación se guarda en disco, lo que en realidad se está almacenando es una lista de estas variables o parámetros. Esta lista (a la que en ocasiones se alude como un "módulo de aplicación") actúa, por tanto, como un conjunto de instrucciones que le dicen al generador de aplicaciones cómo tiene que efectuar una aplicación determinada.

Algunos paquetes llegan más lejos y permiten que el usuario especifique su aplicación en forma de un lenguaje de muy alto nivel (similar al seudolenguaje que utilizamos por primera vez al desarrollar una nueva rutina en el curso de programación BASIC). Este listado lo interpretará el generador, y éste, a su vez, podría ser interpretado por el intérprete de BASIC si el programa de generación estuviera escrito en este lenguaje, lo cual constituye un caso especialmente interesante de jerarquía de software (véase p. 66).

No es raro que los módulos de aplicaciones los creen y comercialicen empresas distintas de la de los autores del generador original. Por ejemplo, dBase II (el más popular de todos los paquetes sofisticados de base de datos que existen para microordenadores) en realidad se puede considerar como un generador de aplicaciones, que contiene módulos consistentes en series de órdenes para base de datos de alto nivel. Los módulos para aplicaciones menos corrientes (como un sistema de contabilidad pensado para corredores de bolsa) se pueden construir sin tener que escribir el programa partiendo desde cero. Dadas las limitadas dimensiones del mercado, un paquete para corredores de bolsa que se ejecutara bajo el dBase II bien podría ser mejor que uno escrito en BASIC, porque el autor del programa tendría que haber concentrado todos sus esfuerzos en la *operación* del programa y no en la *escritura* del código. Las partes del programa más susceptibles de error (p. ej., la manipulación de archivos) serán escritas por los autores del generador y serán probadas por miles de usuarios en distintas aplicaciones.

Pero la diferencia principal entre un generador de programas y un generador de aplicaciones radica en su facilidad para el usuario. El programa final creado por un paquete generador de programas consistirá en código escrito artificialmente, por lo general en un lenguaje como el BASIC. Dicho código será inferior, tanto en eficacia como en estilo, al código generado por los humanos. En el caso del generador de aplicaciones, sin embargo, hasta un 99 % del programa final consistirá quizá en código escrito por la casa de software y éste estará, con toda probabilidad, también en código de lenguaje máquina. Éste es el caso del Silicon Office, uno de los generadores de aplicaciones más sofisticados y más fáciles de utilizar que existen para microordenadores de gestión. El programa resultante será más rápido y más eficaz, incorporará procedimientos de revisión para detectar posibles errores del operador y producirá visualizaciones en pantalla activadas por menú y de trazado claro.

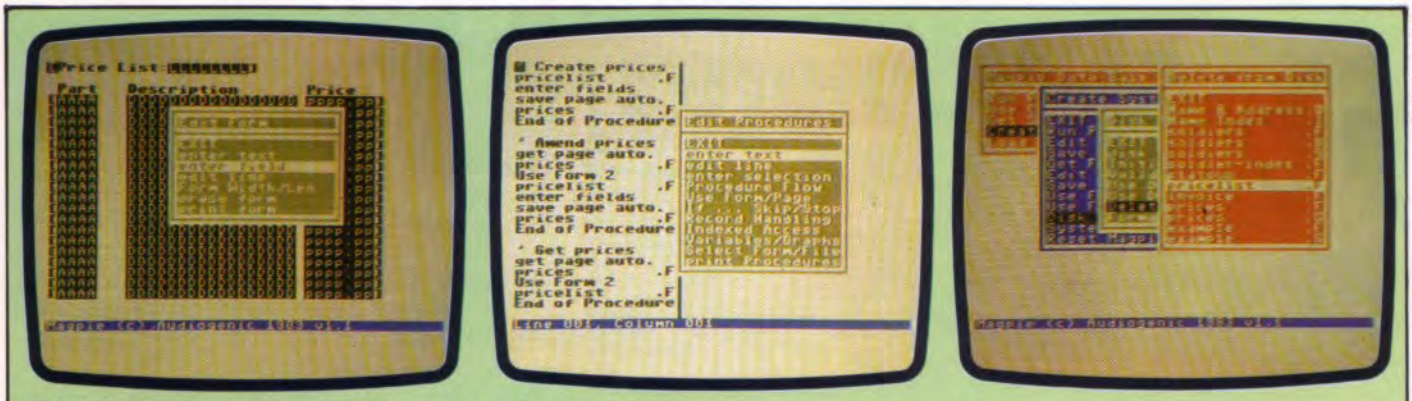
Además, los generadores de aplicaciones no están restringidos a los programas de gestión. Tal vez el mejor ejemplo de un paquete que no sea de gestión es el Pinball Construction Set (véase p. 241), en el cual el módulo de aplicaciones se especifica efectivamente trazando los elementos de la máquina de "millón" (billar electrónico) requerida.

Existen, de hecho, muchos puntos en común entre este tema y la programación orientada de objetos, de la que ya hemos hablado (véase p. 242), pero se pueden resumir en líneas generales en los siguientes términos: estimular al programador para que implemente sus aplicaciones especificando simplemente los objetos que del programa se requieren. Incluso los programas de hoja electrónica sencillos, existentes para ordenadores personales como el Spectrum de Sinclair, se pueden considerar generadores de aplicaciones: el usuario simplemente especifica la relación entre los diversos campos, y el paquete hace por él todo el trabajo rutinario.

Magpie (fabricado por Audiogenic para el Commodore 64 con una unidad de disco) es un generador de aplicaciones preparado para aplicaciones de gestión u otras aplicaciones serias. Se trata de otro paquete que hace buen uso de la programación orientada de objetos visuales: las relaciones entre datos de distintos registros se especifican al diseñar el trazado de esos registros.

A pesar de que no se consideran estrictamente generadores de caracteres, en la actualidad un número creciente de paquetes están incorporando algunos de estos principios. Al ejecutarlos por primera vez, los programas "activados por parámetros" de este tipo formulan al usuario una serie de preguntas y graban las respuestas en disco a lo largo del programa. Esta información determinará algunos de los detalles de la operación del programa. Un programa de facturación, por ejemplo, formularía preguntas relacionadas con la información que a la empresa le interesa incluir en cada factura, y los plazos de crédito estándar que concede. Un juego recreativo preguntaría al usuario con cuántos extraterrestres, bases y cohetes iniciará el juego, o incluso le permitirá diseñar los invasores.

El software se inclina cada vez más a evitar que el usuario tenga que aprender a programar, proporcionando al mismo tiempo un gran nivel de flexibilidad a la operación. Sería muy positivo que el propio software se adaptara a las exigencias del usuario (en vez de que sea el usuario quien deba adaptarse al software).



El Magpie para el Commodore 64

En la primera etapa de la creación de una aplicación se especifica el trazado de los modelos, transacciones e informes, como esta lista de precios. Llenando las columnas con letras (A, D, P), se especifican los campos de la base de datos a utilizar

A continuación se especifican todos los cálculos y procesos en forma de lista de instrucciones en un lenguaje de programación de alto nivel, que Magpie interpretará. Aquí vemos las rutinas para corregir los precios y para cargarlas (GET) desde el disco

Magpie es activado por menú. A medida que se selecciona una opción (p. ej., CREATE), aparece otra junto a ella mostrando todas las opciones CREATE. Esta pantalla visualiza el resultado de seleccionar CREATE, DISK, DELETE (borrar) y el archivo a borrar, en este caso PRICE LIST (lista de precios)

Cortesía de Soft

Ian McKinnell

Imitando instrumentos

La orden ENVELOPE del BBC Modelo B proporciona un control casi ilimitado

En uno de los capítulos anteriores se analizó el formato de la orden SOUND del BBC Micro. No obstante, para examinar con toda profundidad las capacidades de sonido del BBC, es necesario utilizarlo con la versátil orden ENVELOPE. Ésta permite al usuario formar cuatro sonidos, hasta el punto de que se pueden programar imitaciones bastante aceptables de instrumentos convencionales. Además, los efectos sonoros para juegos se pueden refinar para que suenen de manera muy parecida a las explosiones o los disparos.

ENVELOPE se construye de la siguiente manera:

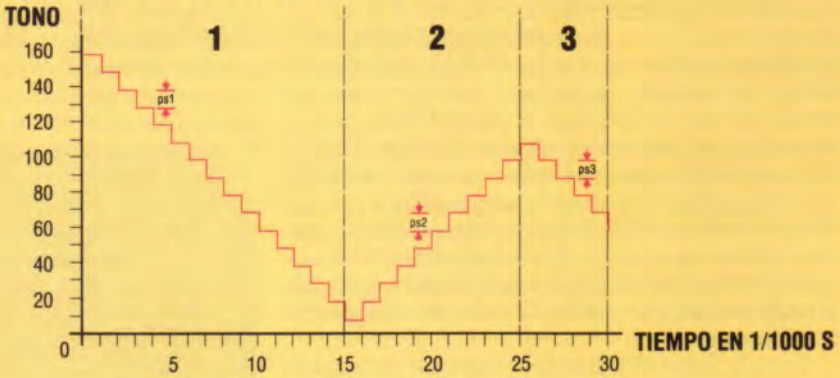
```
ENVELOPE N,T,PS1,PS2,PS3,NS1,NS2,NS3,
AR,DR,SR,RR,FAL,FDL
```

El primer parámetro, N, establece el número de envoltura y sirve para identificar ésta con las órdenes relacionadas SOUND o SOUND &. Se puede sustituir una de las cuatro envolturas para el volumen fijado (V) establecido por SOUND mediante un número negativo (de 0 a −15; véase p. 388).

T (de 0 a 127) y (de 128 a 255)

Éste es el control del tiempo para la orden. Establece la duración de cada intervalo de la construcción de la envoltura en centésimas de segundo. Por consiguiente, T = 5 significa que cada intervalo de envoltura dura cinco centésimas de segundo (0,05 segundos). Agregando 128 a la duración del intervalo requerida se suprimirá la repetición automática de la envoltura de tono, de modo que T establecido en 5 + 128 = 133 da una duración de intervalo de cinco centésimas de segundo para una envoltura de tono que se produce una sola vez en la nota. La utilización del término "envoltura de tono" puede parecer algo confusa dado que previamente la envoltura se ha empleado en términos de volumen, pero en este caso se refiere a la variación del tono en el transcurso de la duración de una nota. Esta configuración posee poco valor desde el punto de vista musical, a menos que se desee un "vibrato", pero puede ser útil para conferir a los efectos sonoros interesantes "trinos". Como se ve en el diagrama, la envoltura de tono se divide en tres secciones. La respuesta de cada sección se puede determinar mediante los números relacionados con PS y NS, del siguiente modo:

Envoltura de tono



De colores

La utilización de sprites en el Commodore 64

Una de las configuraciones más atractivas del Commodore 64 es su capacidad para el empleo de sprites. Los sprites se construyen del mismo modo que los caracteres definidos por el usuario, pero son mucho más grandes, ya que constan de 21 filas de 24 pixels. Los sprites no se visualizan en la matriz normal para caracteres en pantalla y ello permite que se pueda mover un pixel cada vez, en vez de exigir ocho pixels para mover un carácter de una celda a la siguiente. En la pantalla se pueden visualizar hasta ocho sprites a la vez, cada uno con estas características individuales de programación:

Forma y color

Un sprite se define de forma muy similar a un carácter de 8 × 8 pixels, pero se necesitan 63 bytes para retener los patrones codificados en forma binaria. Una vez que se ha definido la forma de esta manera, se obtiene un bloque de 63 posiciones consecutivas. Cada sprite posee un indicador de datos que señala la zona de la que proviene la forma del sprite, lo cual significa que puede haber más de un sprite que "mire" hacia la misma zona de memoria; es decir, que puede haber sprites idénticos. También se puede cambiar su forma haciendo que su indicador mire una zona diferente de memoria.

Cada sprite se puede colorear con alguno de los 16 colores existentes. Asimismo, se puede pintar de varios colores, con el inconveniente de que a veces la resolución horizontal se divide en dos.

Tamaño y movimiento

Los sprites se pueden ampliar horizontal y verticalmente, o en ambas direcciones, para duplicar su tamaño original. El sprite totalmente ampliado mide 48 × 42 pixels. En este caso también se ha de pagar un precio por ello: la resolución se divide por dos en la dirección de la ampliación.

PS1,PS2 y PS3 (de −128 a 128)

PS indica *pitch step* (intervalo de tono). La orden SOUND establece el tono al comienzo de la nota. PS1 establece la modificación, positiva o negativa, de tono por intervalo para la primera sección, PS2 para la segunda sección y PS3 para la tercera sección. PS, al igual que SOUND, se establece en cuartos de semitonos.

NS1,NS2 y NS3 (de 0 a 255)

NS quiere decir *number of steps* (número de intervalos) por sección; y, conjuntamente con PS, selecciona la velocidad a la cual se modifica el tono en una sección y también la duración de toda la envoltura de tono. Los valores PS y NS para el ejemplo anterior son los siguientes:

T = 1	PS1 = −10	NS1 = 15
	PS2 = +10	NS2 = 10
	PS3 = −15	NS3 = 5

En este caso, el tono se establece mediante SOUND = 160. El resultado es:

ENVELOPE 1,1,−10,10,−15,15,10,5,0,0,0,0,0,0

La duración de la envoltura viene dada por (NS1+NS2+NS3) × T, que en este caso es (15+10+5) × 1 = 0,3 segundos. Normalmente, la envoltura de tono se repetirá de manera automática en el transcurso de la duración de una nota, a menos que lo impida el parámetro de tiempo, T.

En el próximo capítulo de *Sonido y luz* volveremos a ocuparnos de las configuraciones de sonido del BBC Micro y explicaremos el funcionamiento de la envoltura de volumen.

Un sprite se puede desplazar un pixel cada vez y la posición antigua se borra de forma automática. Los sprites también se pueden mover para salir del área normal de visualización de la pantalla o para entrar en ella.

Prioridad y colisión

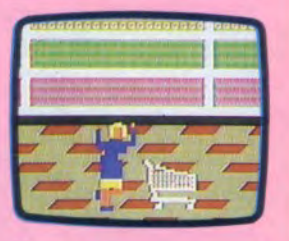
Cuando dos sprites se cruzan en su camino, uno aparece como si pasara por delante del otro. Si hay algún agujero en el sprite que pasa por delante, a través del mismo se verá el sprite que pasa por detrás. Se puede utilizar la prioridad para conseguir algunos interesantes efectos tridimensionales. A cada sprite se le da un número de 0 a 7 y la sencilla ley de la prioridad consiste en que los sprites de números menores pasan por delante de los sprites de números mayores. Por regla general, los sprites se mueven por delante de cualquier carácter normal que haya en la pantalla, pero también se pueden programar de tal manera que pasen por detrás de ellos.

Esta última característica se puede utilizar para dar la impresión de profundidad en la pantalla.

Cuando dos sprites se cruzan, se señala en un registro de colisiones. Mirando (PEEK) en este registro, el programador puede saber cuáles son los sprites implicados. Existe otro registro similar que señala cuándo un sprite ha entrado en colisión con algún carácter del fondo.

Gracias a estos recursos, se pueden escribir programas en BASIC para controlar juegos de movimiento rápido. Lamentablemente, no existen órdenes especiales en BASIC para controlar las configuraciones de los sprites; todo se debe hacer mediante una sucesión de sentencias POKE en la memoria del Commodore 64. Un método alternativo y más sencillo para crear sprites consiste en comprar un cartucho BASIC de Simon.

Basic de Simon
Se trata de un cartucho conectable para ampliar las capacidades de alta resolución y de manipulación de sprites de que dispone el programador en BASIC. El cartucho viene con un voluminoso manual en el que se detallan las 114 órdenes extras. Éstas incluyen órdenes para funcionar en modalidad de alta resolución, para seleccionar los colores del fondo y de primer plano, y para dibujar círculos, elipses, rectángulos y líneas rectas. El manual de instrucciones incluye ayudas para el diseño y la creación de sprites, órdenes para encender y apagar los sprites, y formas de colocarlos en la pantalla



Segundo paso
Estas líneas se pueden agregar al listado del programa de "supermercado" de la p. 359. Esta sección del programa utiliza tres sprites ampliados y multicolores: dos representan la figura humana y otro el carrito de la compra. Los señaladores de datos de los sprites están manipulados para que la forma de la mujer se modifique. Esto da el efecto de una silueta que se desplaza bailando por la pantalla. Para utilizar el programa de "supermercado" como una subrutina de este programa, cambie la línea 3270 para que diga: 3270 RETURN

```
90 REM **SPRITES 64**
100 PRINT " "
110 V = 53248
120 REM ----LEER DATOS SPRITE----
130 FOR I = 12288TO12350:READA:POKEI,A:NEXT
140 FOR I = 12352TO12414:READA:POKEI,A:NEXT
150 FOR I = 832TO894:READA:POKEI,A:NEXT
160 FOR I = 896TO958:READA:POKEI,A:NEXT
170 FOR I = 12416TO12478:READA:POKEI,A:NEXT
180 REM ----AMPLIAR SPRITES----
190 POKEV+23,7:POKEV+29,7
200 REM ----COLOR SPRITES----
210 POKEV+39,10:POKEV+40,10
220 POKEV+41,1
230 REM ----MULTICOLOR----
240 POKEV+28,3:POKEV+37,7:POKEV+38,9
300 REM ----INDICADORES MEMORIA----
310 POKE2040,192:POKE2041,193:POKE2042,194
320 REM ----ESTABLECER CORDENADAS Y----
330 Y0 = 150:Y1 = Y0+42:Y2 = Y0+34
340 POKEV+1,Y0:POKEV+3,Y1:POKEV+5,Y2
400 REM ----ENCENDER SPRITES----
410 POKEV+21,7
500 GOSUB3000:REM OMITIR A FALTA DE SUBRUTINA
1000 XO = 20
1010 POKE2040,13:POKE2041,14
1020 POKEV,XO:POKEV+2,XO:POKEV+4,XO+48
1030 FORI = 1TO500:NEXT
1040 POKE2040,192:POKE2041,193
1050 XO = XO+5
1060 POKEV,XO:POKEV+2,XO:POKEV+4,XO+48
1070 FORI = 1TO500:NEXT
1080 XO = XO+5
1090 IFXO>200THEN1110
1100 GOTO1010
1110 FORJ = 1TO10
1120 POKE2040,13:POKE2041,14
1130 FORI = 1TO50:NEXT
1140 POKE2040,192:POKE2041,193
1150 FORI = 1TO50:NEXT
1160 NEXT
1170 GOTO1170
9000 REM ----PARTE SUPERIOR MUJER----
9010 DATA0,0,0,0,21,0,0,21,0,0,22,0,0,86,0
9020 DATA0,86,0,0,86,0,0,40,0,0,252,0
9030 DATA15,255,0,255,255,0,255,255,0
9040 DATA195,255,0,195,255,0,195,243,254
9050 DATA207,243,254
9060 DATA143,240,0,143,252,0,15,252,0
9070 DATA15,252,0,15,252,0
9100 REM ----PARTE INFERIOR MUJER----
9110 DATA15,252,0,15,252,0,15,252,0
9120 DATA15,252,0,5,84,0,5,84,0,5,84,0
9130 DATA5,84,0,10,40,0,234,40,0,234,40,0
9140 DATA234,40,0,192,40,0,192,40,0,0,40,0
9150 DATA0,40,0,0,63,0,0,63,0,0,0,0,0,0
9160 DATA0,0,0
9200 REM ----PARTE SUPERIOR MUJER #2----
9210 DATA0,0,0,0,20,32,32,85,32,32,105,48,48,105,48
9220 DATA48,105,48,48,105,48,48,40,48,48,252,48
9230 DATA63,255,240,63,255,240,63,255,0
9240 DATA3,255,0,3,255,0,3,240,0
9250 DATA15,240,0
9260 DATA15,240,0,15,252,0,15,252,0
9270 DATA15,252,0,15,252,0
9300 REM ----PARTE INFERIOR MUJER #2----
9310 DATA15,252,0,15,252,0,15,252,0
9320 DATA15,252,0,5,84,0,5,84,0,5,84,0
9330 DATA5,84,0,10,40,0,58,168,0,58,168,0
9340 DATA58,0,0,58,0,0,10,0,0,10,0,0
9350 DATA10,0,0,15,192,0,15,192,0,0,0,0,0,0
9360 DATA0,0,0
9400 REM ----CARRITO----
9410 DATA192,0,0,224,0,0,118,0
9420 DATA0,55,192,0,32,60,0,53
9430 DATA87,240,32,0,15,53,85,85
9440 DATA32,0,3,53,85,85,0,0,3
9450 DATA21,85,85,31,255,255,24,0
9460 DATA0,12,0,0,12,0,0,31,255
9470 DATA240,31,255,255,1,0,2,7
9480 DATA0,14,7,0,14
```

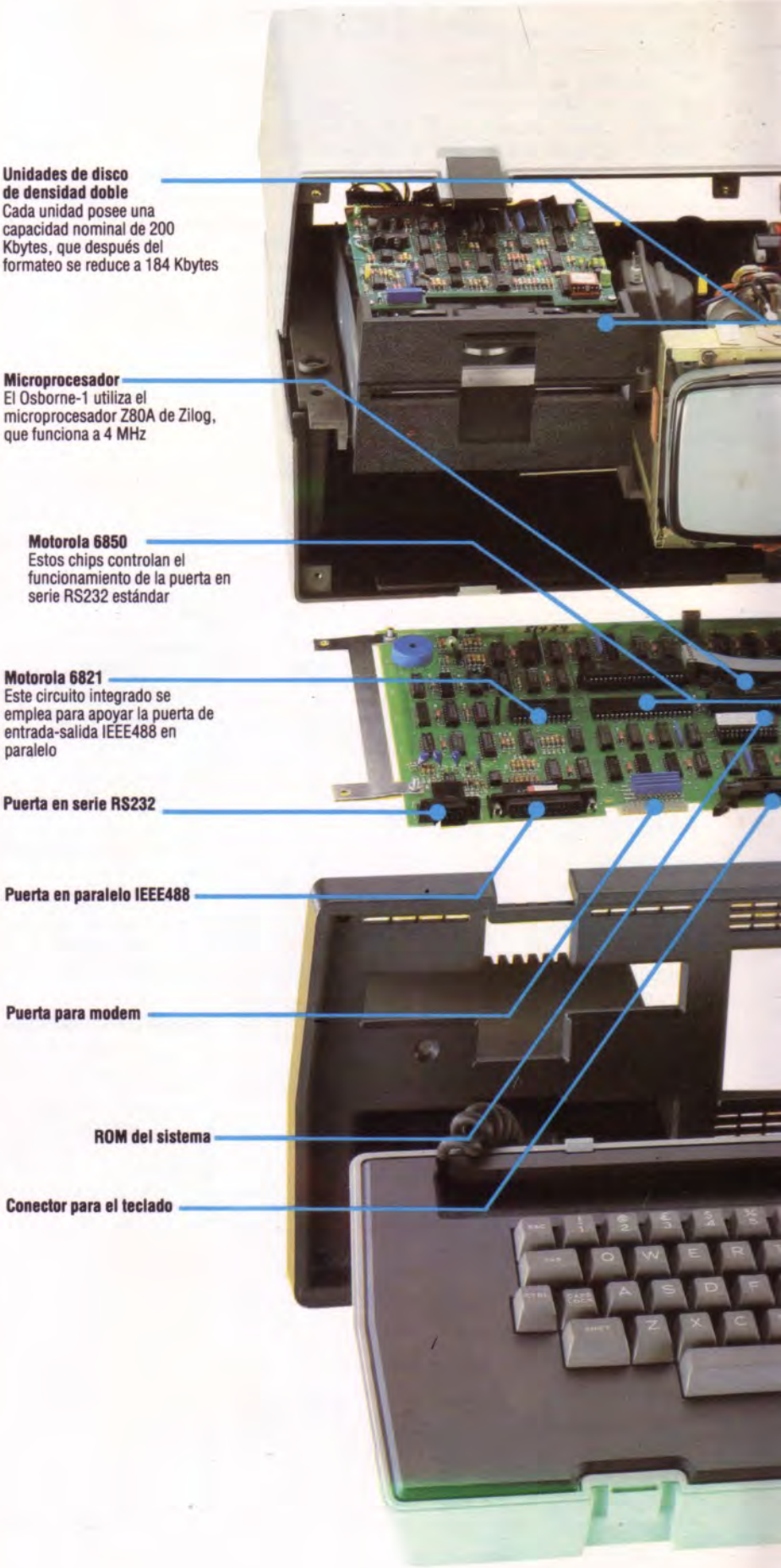
Osborne-1

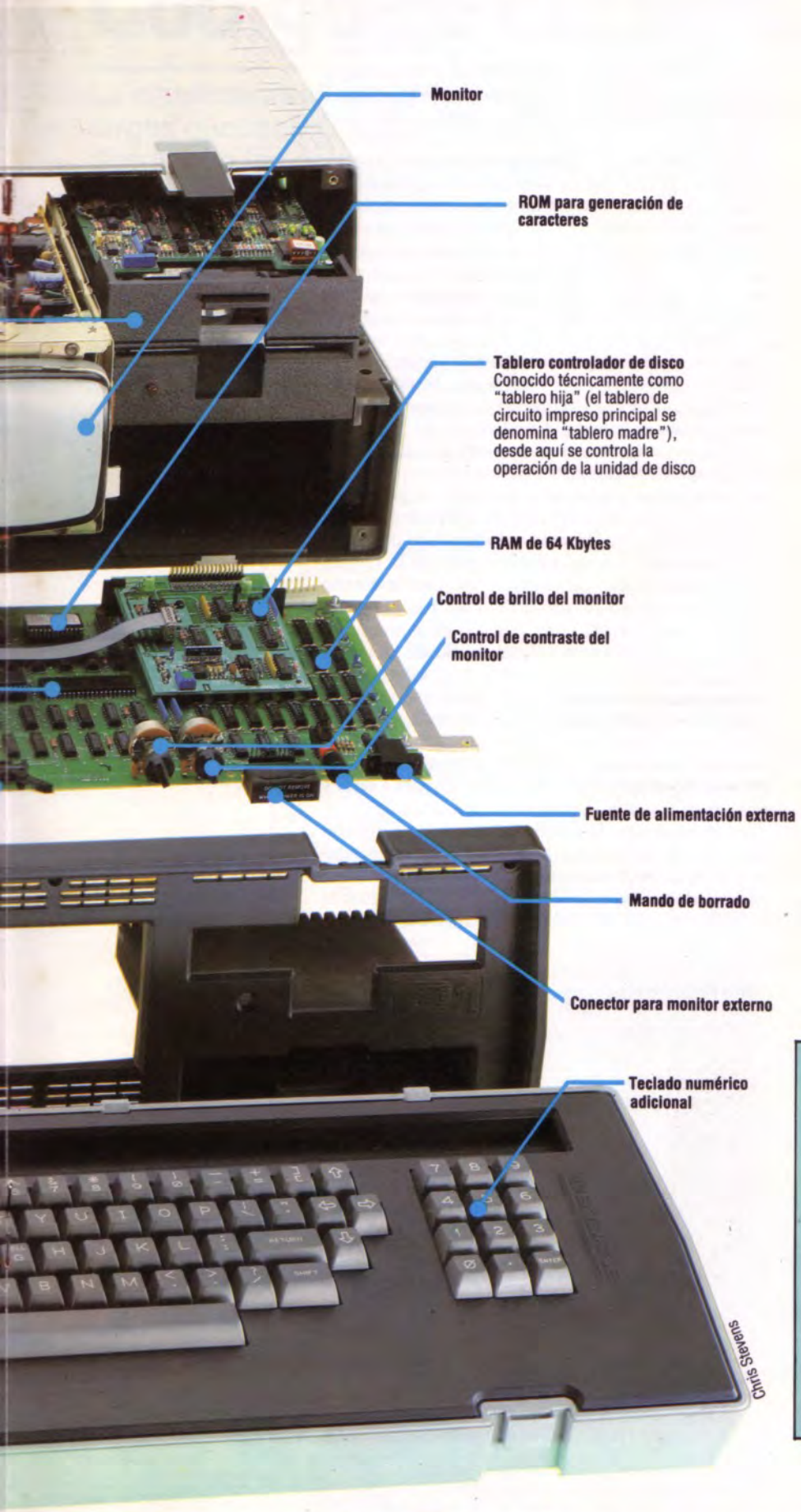
Éste es el primer micro portátil que se diseñó y fue también el primero que se vendió con software incluido

Aunque no entra estrictamente en la categoría de ordenador personal, el Osborne-1 es una máquina particularmente interesante porque fue el primer microordenador portátil totalmente autocontenido. Con sus dos unidades de disco incorporadas y su pequeño monitor, el Osborne le ofrece al usuario la posibilidad de trasladar consigo, vaya donde vaya, su propia capacidad para procesamiento de datos. La máquina sólo carece de un paquete de pilas interno, pero el fabricante pensó que ello incrementaría el peso global del ordenador hasta un límite que excedía lo razonable (ya pesa 10,5 kg). Hay, no obstante, un conector de corriente continua en el panel frontal, junto con las otras conexiones para interfaces. La máquina requiere tanto entradas de 12 v como de 5 v: la primera para las unidades de disco y la segunda para la lógica.

El otro factor que incide en contra de su utilización como ordenador personal es su alto precio, aunque en éste se incluyen algunos de los paquetes de software de gestión mejor establecidos que existen, como son: CBASIC Microsoft, una versión compilada del lenguaje BASIC, que posibilita una operación mucho más veloz de los programas; Supercalc, ampliamente reconocido como el mejor de los programas de hoja electrónica de la primera generación; Wordstar y Mailmerge, los paquetes para tratamiento de textos transportables (no limitados a ningún tipo de máquinas) más vendidos; y, quizá lo mejor de todo, el sistema operativo CP/M (Control Program/Monitor), de Digital Research, que permite ejecutar en cualquier máquina que lo utilice una amplia gama de paquetes de software.

El Osborne-1, al igual que el Apple II (véase p. 349), requiere cargar su sistema operativo desde disco. Además de supervisar la operación interna del ordenador, el sistema CP/M es capaz de realizar la mayoría de las labores de conservación: hacer copias de archivos y de discos enteros, iniciar nuevos discos, catalogar el contenido de éstos, etc. Pero el sistema CP/M posee también otros puntos fuertes. En primer lugar, se puede escribir software para el sistema operativo, independientemente de en qué máquina opere. Para la casa fabricante de software esto significa un mercado potencial mucho mayor y por ello se puede invertir muchísimo más dinero en la producción, lo que a su vez asegura un paquete de mayor calidad. En segundo lugar, para un usuario de CP/M experimentado, el tipo de máquina es casi independiente, y esto permite mejorar y perfeccionar el hardware sin la molesta tarea de volver a dar entrada a los archivos de datos y de convertir los programas. Durante un breve período, Osborne incluyó en el precio de compra el dBase II de Ashton-Tate, el más eficaz de todos los programas de administración de bases de datos basados en microordenador.





OSBORNE-1

DIMENSIONES
510 x 325 x 225 mm

PESO
10,5 kg

CPU
Z80A

VELOCIDAD DEL RELOJ
4 MHz

MEMORIA
RAM de 64 Kbytes
ROM de 4 Kbytes

VISUALIZACION EN VIDEO
24 filas de 52 caracteres en una visualización real de 128 x 32

INTERFACES
RS232, IEEE, modem

LENGUAJE SUMINISTRADO
BASIC, Ensamblador Z80

OTROS LENGUAJES DISPONIBLES
Todos aquellos que se ejecuten bajo CP/M.

VIENE CON
CP/M, Wordstar, CBASIC, MBASIC, Mailmerge, Supercalc, Manuales

TECLADO
Estilo máquina de escribir, 69 teclas incluyendo teclado numérico adicional

DOCUMENTACION
Adam Osborne le vendió su empresa editorial a McGraw-Hill con el fin de financiar la fabricación de los ordenadores Osborne, de modo que no es sorprendente que la documentación del manual sea muy buena. El único fallo es la falta de un índice global

Control Program/Monitor

Los ordenadores de unidad principal y los miniordenadores se han beneficiado de los sistemas operativos independientes de la máquina desde que a mediados de los años sesenta se introdujera la segunda generación de ordenadores; no obstante, habrían de transcurrir doce años antes de que los sistemas de control de este tipo llegaran a los microordenadores. El CP/M (Control Program/Monitor), de Digital Research, fue el primero de estos sistemas. Diseñado para los microprocesadores 8080 de Intel y Zilog Z80, posee una gama de programas de uso práctico y domésticos, y define asimismo las formas en que se pueden interrumpir y continuar los programas en ejecución. Otra importante ventaja radica en la definición de las estructuras y los trazados de los archivos, que también manipula el CP/M. Utilizando un programa de intercambio como el BSTAM, que reduce los archivos de cualquier clase a su forma básica, se pueden transferir programas escritos para CP/M entre diversas máquinas, independientemente de su tipo o especificación. Esto significa que el usuario de CP/M dispone de una inmensa cantidad de software

Lamentablemente para Osborne, la mayoría de las empresas comerciales de Estados Unidos concentraron su atención en el ordenador personal de IBM, máquina de 16 bits basada en el microprocesador 8088 de Intel. Concebido como una solución provisional (ostenta un direccionamiento de 16 bits, pero la transferencia de datos es de sólo ocho bits), el 8088 se convirtió en un estándar industrial de facto sencillamente por el hecho de haberlo escogido la IBM para su primera incursión en el mercado de microordenadores.

El ordenador personal IBM utiliza un sistema operativo diseñado especialmente que se denomina PC-DOS. Digital Research lanzó dos nuevas versiones del sistema operativo CP/M: Concurrent CP/M, que permite una verdadera multiprogramación multiusuario, y el CP/M86, diseñado para el chip 8086 de Intel, que incorporaba un direccionamiento de 16 bits y también una transferencia de datos de 16 bits.

Lamentablemente, todos estos desarrollos llegaron demasiado tarde como para impedir que el Osborne-1 se hundiera en los avatares del mercado, y en 1983 la Osborne Computer Corporation (la empresa principal de Estados Unidos) presentó una liquidación voluntaria. Con su memoria de 64 Kbytes (60 Kbytes disponibles para el usuario) y sus unidades de disco gemelas de 183 Kbytes, el Osborne-1 continúa siendo un ordenador razonablemente eficaz. Contribuyen a ello sus puertas RS232 e IEEE incorporadas, la puerta para modem y su capacidad para funcionar con un paquete de pilas, y resulta fácil comprender por qué el ordenador fue un éxito de venta instantáneo y por qué continúa gozando de popularidad entre los usuarios aun después de la desaparición de su primer fabricante.

Una configuración muy interesante del Osborne-1, que hasta cierto punto la comparte con el Epson HX-20 (véase p. 169), es la provisión de una "pantalla virtual" de tamaño más de tres veces mayor que la visualización proporcionada, de 52 columnas por 24 líneas. La utilización de la tecla de control (una exigencia del CP/M estándar) y las teclas del cursor permite que la visualización se mueva a través de la memoria de pantalla real. En gran medida esto elimina la mayoría de los inconvenientes que plantea el pequeño tamaño (8,75×6,6 cm) de la pantalla, aunque quienes no son usuarios de la máquina suelen manifestar su sorpresa ante el hecho de que una visualización cuyos caracteres miden sólo 2 mm de altura pueda ser legible e incluso cómoda de utilizar.

De hecho, son pocos los usuarios que no consiguen adaptarse a esta miniaturización, si bien Osborne proporcionó un conector para monitor externo que reproduce el contenido de la pantalla pequeña en una unidad adicional de mayores dimensiones. En realidad, lejos de considerar que el tamaño de los caracteres era demasiado pequeño, hubo una considerable demanda por parte de los usuarios en el sentido de que toda la pantalla virtual de cuatro Kbytes (128 columnas por 32 filas) se visualizara en todo momento, y Osborne fabricó una modificación para satisfacer esa especificación. Ésta le permite al usuario elegir entre tres anchos diferentes: 52, 96 o 128 caracteres, e incluso en la mayor densidad de caracteres, éstos siguen siendo definidos y legibles.

El teclado del Osborne-1, que se sujeta al panel frontal del ordenador como una "tapa", haciéndolo a prueba de intemperie, es una unidad de 69 teclas. Posee teclas normales estilo máquina de escribir, con la adición de teclas de control CTRL y escape ESC, más un relleno numérico de 12 teclas en el lado derecho que incluye teclas extras de punto y aparte y ENTER. Utilizando un programa CP/M denominado SETUP, las funciones de las teclas numéricas (empleadas conjuntamente con la tecla de control) las puede definir el usuario hasta un máximo de 96 caracteres. Esta configuración es especialmente útil si se ha de usar con frecuencia una palabra, una frase o una orden. Los resultados del programa SETUP se escriben en cada disco, en vez de almacenarse en la memoria, de modo que las funciones se pueden preprogramar por separado para cada paquete diferente de software. El ordenador establece automáticamente sus funciones cada vez que se carga el sistema operativo.

Además de los 96 caracteres estándar de mayúsculas y minúsculas, existen 32 caracteres predefinidos para gráficos, aunque a éstos sólo se puede acceder mediante un programa de aplicaciones.

Debido a que el Osborne-1 utiliza chips de apoyo de la serie 6800 y no sus equivalentes de la familia 8080 (como sería de esperar en una máquina CP/M), el método de sondeo del teclado es ligeramente distinto. Existe una porción de la memoria reservada para interpretar las pulsaciones de las teclas, y la ROM del sistema verifica constantemente si se ha pulsado alguna. En el teclado propiamente dicho no hay ninguna lógica de decodificación. Es esta característica lo que permite una programación sencilla de las teclas de función, y como las funciones se almacenan en la RAM, cabe acceder a ellas y modificarlas desde dentro de un programa.

A pesar de que la Osborne Computer Corporation entró en liquidación voluntaria, la filial británica creó una empresa separada y continuó su actividad comercial. Sea cual fuere el futuro que aguarde a esta máquina, su calidad no admite duda.

Para llevárselo
Además de su excelente relación calidad-rendimiento, el Osborne-1 posee la ventaja adicional de su portabilidad. Con sus 10,5 kg no es un peso ligero, pero es autocontenido y está bien equilibrado, de modo que se transporta sin dificultad.
Una de las condiciones básicas en el momento de precisar sus dimensiones físicas fue su adaptabilidad al limitado espacio disponible debajo del asiento en un avión

Ian McKinnell

Código de clasificación

La clasificación Shell es mucho más eficaz para las matrices largas que la clasificación burbuja o por inserción. Funciona dividiendo los datos en una serie de "cadenas"

En la p. 286 analizamos dos métodos para clasificar por orden una matriz: las clasificaciones burbuja y por inserción. Por lo general, la clasificación burbuja es más fácil de realizar, pero la clasificación por inserción es más rápida. La experiencia con estos dos procedimientos demuestra que lo que más tiempo ocupa es cambiar los naipes de un lugar a otro a través de distancias cortas: suele ser preferible cambiar un naipe de lugar una sola vez a través de una distancia larga que cambiarlo varias veces en desplazamientos cortos.

```
7999 REM * * * * * * * * * * *
8000 REM *       SHELL       *
8001 REM * * * * * * * * * * *
8025 PRINT "CLASIFICACION SHELL - ADELANTE!!!!!"
8050 LET LK = LT
8100 FOR Z = 0 TO I STEP 0
8150 LET LK = INT(LK/II)
8200 FOR LB = I TO LK
8250 LET LL = LB+LK
8300 FOR P = LL TO LT STEP LK
8350 LET D = R(P)
8400 FOR Q = P TO LL STEP-LK
8450 LET R(Q) = R(Q-LK)
8500 IF D< = R(Q) THEN LET R(Q) = D: LET Q = LL
8550 NEXT Q
8600 IF D>R(LB) THEN LET R(LB) = D
8650 NEXT P
8700 NEXT LB
8750 IF LK = I THEN LET Z = I
8800 NEXT Z
8850 PRINT "CLASIFICACION SHELL - STOP!!!!!"
8900 RETURN
```

Para agregar esta rutina al programa de demostración de clasificación de la p. 287, cambie la línea 350 por:

```
350 LET I = 1: LET 0 = 0: LET II = +: LET TH = 3
```

y cambie la línea 900 por:

```
900 ON SR GOSUB 6000,7000,8000
```

Un procedimiento mejor que estos dos es el denominado *clasificación Shell* (llamado así en honor de su creador, D. Shell). Este método disminuye el desorden de la matriz al comenzar la clasificación (de modo que los datos no están muy lejos de sus posiciones verdaderas) y permite que las permutaciones operen a través de distancias relativamente grandes. He aquí un método para esta clasificación:

1) Disponga los naipes de un mismo palo en cualquier orden. Éstos habrán de clasificarse por orden descendente, de manera que el Rey será el naipe situado más hacia la izquierda y el As el situado más a la derecha. Cuente los naipes, divida el número (en este caso, 13) por dos, ignorando la cantidad que sobre, y escriba el resultado (es decir, 6) en un trozo de papel bajo el título "el eslabón".

2) Coloque una moneda de una peseta debajo del naipe situado más a la izquierda (a ésta la llamará posición 1) y una moneda de cinco pesetas en la posición eslabón (es decir, en posición 6 en el primer caso). Todos los naipes desde la posición 1 hasta la posición eslabón serán los naipes situados al extremo izquierdo de una serie de "cadenas" de cartas. El número de cadenas será igual al valor en curso del eslabón. Cada cadena se forma empezando por el naipe situado más a su extremo, sumando el eslabón al número de posición del naipe del extremo para obtener la posición del naipe siguiente, y así sucesivamente hasta que se alcanza o se supera el extremo de la matriz. La primera cadena comprende los naipes de las posiciones 1, 7 y 13; la segunda cadena consta de los naipes de las posiciones 2 y 8; la tercera, de los de las posiciones 3 y 9. La última es la de los naipes de las posiciones 6 (el valor actual del eslabón) y 12.

3) Ahora, después de haber marcado los límites con las monedas de una y de cinco pesetas, desplace los naipes que constituyen la primera cadena fuera de la matriz de modo que usted los pueda ver en solitario, y póngalos en orden utilizando ya sea la clasificación burbuja o la clasificación por inserción tal como las describimos en la p. 286.

4) Vuelva a desplazar la cadena ordenada de nuevo en los espacios de la matriz y repita el punto anterior con la siguiente cadena, y después con la siguiente, hasta que se hayan clasificado todas las cadenas cuyos naipes más sobre la izquierda estén dentro de las monedas de una y cinco pesetas.

5) Cuando haya clasificado todas las cadenas, divida el eslabón por dos. Si éste es menor que uno, la matriz está ordenada. Si no, repita el proceso desde el paso 2 con el nuevo valor del eslabón.

Panel de clasificación Shell

Posición n.º									Valor eslabón	Comentarios
1	2	3	4	5	6	7	8	9		
2	8	9	3	D	5	K	6	7	(9/2)=>4	Empezar pasada
*	+	@	$	*	+	@	$	*		Formar cadenas
D			7					2		Clasif. cadena 1
	8				5					Clasif. cadena 2
		K				9				Clasif. cadena 3
			6				3			Clasif. cadena 4
D	8	K	6	7	5	9	3	2		Empezar pasada
D	8	K	6	7	5	9	3	2	(4/2)=>2	Fin pasada
*	+	*	+	*	+	*	+	*		Formar cadenas
K		D		9		7		2		Clasif. cadena 1
	8		6		5		3			Clasif. cadena 2
K	8	D	6	9	5	7	3	2		Fin pasada
K	8	D	6	9	5	7	3	2	(2/2)=>1	Empezar pasada
*	*	*	*	*	*	*	*	*		Formar cadena 1
K	D	9	8	7	6	5	3	2		Fin pasada

CLAVE	
*	Miembro cadena 1
+	Miembro cadena 2
@	Miembro cadena 3
$	Miembro cadena 4

Clasificación Shell
El ejemplo de la clasificación Shell que vemos en el papel, para una mano reducida, demuestra su método exclusivo de dividir la matriz en una serie de cadenas (con espaciaciones basadas en el número de eslabón corriente). Estas cadenas se clasifican por separado, en este caso utilizando el método por inserción, antes de que se complete una pasada.
El listado del programa que aquí damos para una clasificación Shell se debe emplear conjuntamente con el programa *testbed* de la p. 287. Cuando lo probamos, se observó una significativa mejora respecto a los otros métodos de clasificación cuando el número de ítems a clasificar superaba los 40

Con una sola mano

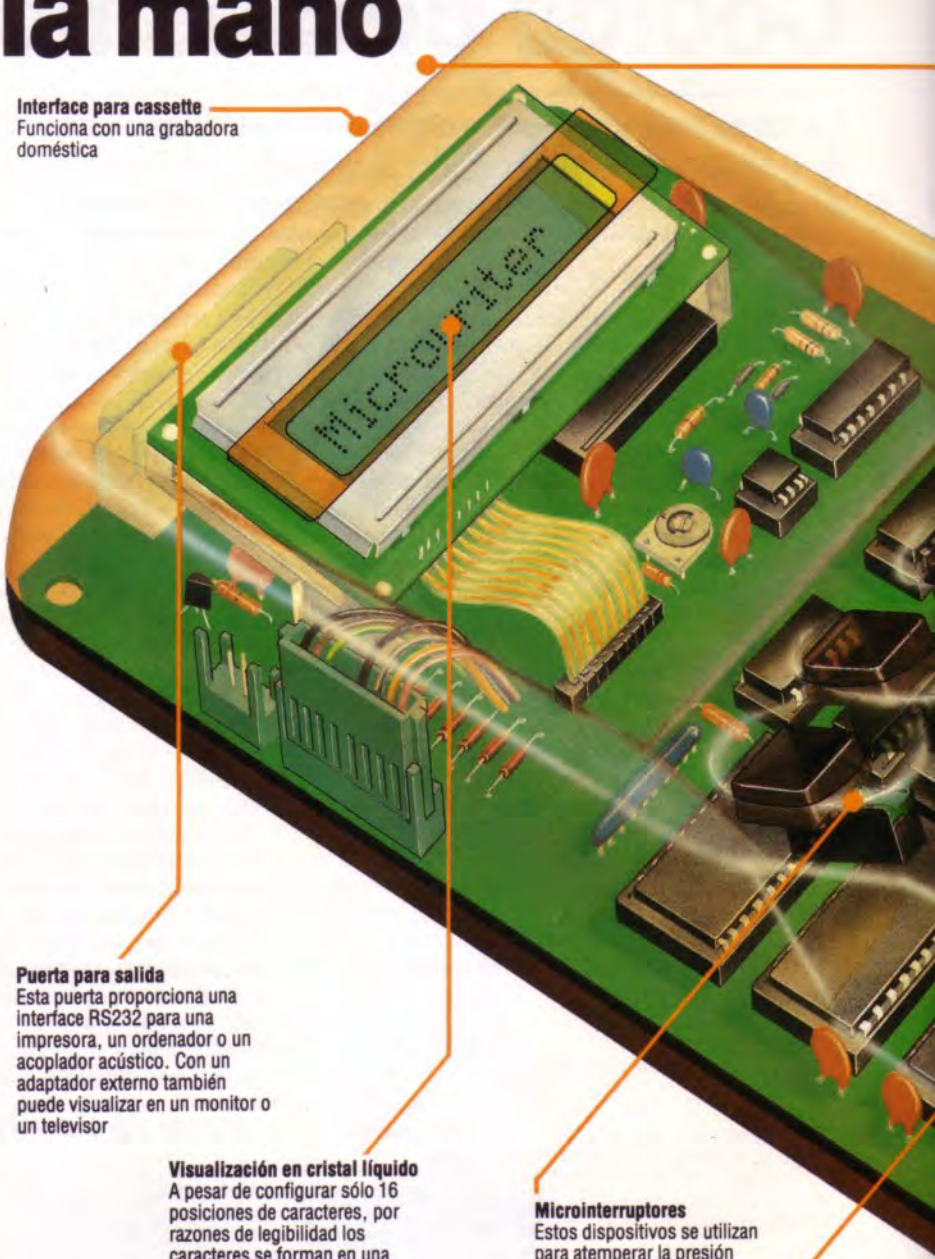
El Microwriter es un procesador de textos portátil que se puede manejar con una sola mano. El teclado de seis botones quizá se utilice pronto en los ordenadores

Tener un procesador de textos en la oficina, o un ordenador personal con un programa para tratamiento de textos, puede ser una buena idea. Aparte de eliminar el fatigoso trabajo de escribir cartas o documentos rutinarios, puede ayudar con la documentación de programas, elaborar con toda rapidez copias de avisos o cuidar de una agenda de direcciones. Sin embargo, el problema surge cuando usted desea llevarse notas de su casa o de la oficina en una forma que el ordenador pueda comprender.

Existe un mercado en crecimiento para los sistemas informáticos portátiles como el Tandy 100 y el Epson HX-20. Si bien éstos poseen la ventaja de que pueden actuar como procesadores de textos portátiles, o como terminales remotos de sistemas mayores, no son tan manuales como un bloc de notas o un dictáfono. ¿Qué diría usted de un sistema de tratamiento de textos que fuera lo suficientemente pequeño como para llevarlo en el bolsillo? Un sistema que fuera tan compacto como para funcionar a pilas y que se pudiera manejar con una sola mano, pudiéndose conectar con una impresora o incluso con otro ordenador.

Hace aproximadamente cuatro años que existe este tipo de dispositivo y se denomina Microwriter. Concebido originalmente por el norteamericano Cy Enfield, cambia el teclado QWERTY en favor de un sistema exclusivo de teclas múltiples con sólo seis teclas de botón. El concepto surgió originalmente del deseo de crear un juego manual basado en palabras, para el cual hasta un teclado en miniatura hubiera resultado demasiado grande y demasiado caro. La respuesta evidente era la creación de un tipo especial de teclado que utilizaba sólo unas pocas teclas con suficientes combinaciones para especificar todos los símbolos alfanuméricos. El sensacional avance se produjo con la invención de un sistema de código simbólico que es exclusivo del Microwriter.

A primera vista parece imposible que las letras del alfabeto, por no hablar de los símbolos numéricos y los signos de puntuación, se puedan crear a partir de combinaciones entre apenas seis teclas, pero éstas son más que suficientes. Y aprenderse todas las combinaciones comunes lleva apenas unas pocas horas. En realidad, como con cierta justificación afirman los fabricantes, es mucho más fácil de aprender que un teclado QWERTY. La combinación de teclas necesaria para cada una de las letras se basa en la forma física de la letra, un código que suele resultar más fácil de aprender a quienes no saben mecanografía. Debido a que sólo se necesita una mano, el Microwriter también les abre las puertas del tratamiento de textos a los minusválidos, que no pueden manipular las múltiples pulsaciones de teclas que a menudo se requieren para generar órdenes en un teclado convencional.

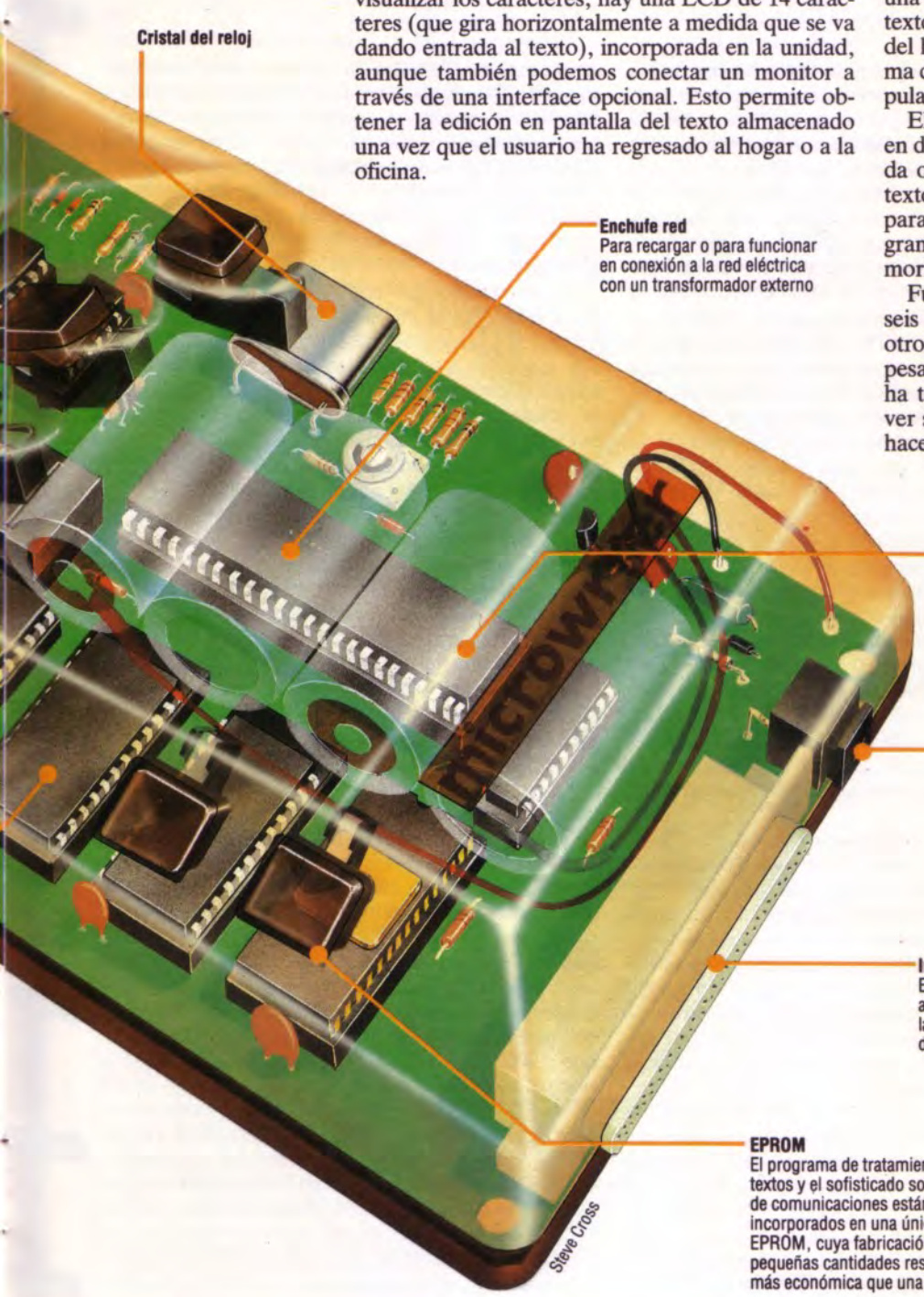


Internamente, el Microwriter está diseñado para ser lo más portátil posible. Tanto el microprocesador interno como su memoria son dispositivos CMOS (Complementary Metal Oxide Silicon) que ayudan a reducir el consumo energético. Un paquete de pilas níquel-cadmio recargable proporciona la energía suficiente para 30 horas de uso. Para visualizar los caracteres, hay una LCD de 14 caracteres (que gira horizontalmente a medida que se va dando entrada al texto), incorporada en la unidad, aunque también podemos conectar un monitor a través de una interface opcional. Esto permite obtener la edición en pantalla del texto almacenado una vez que el usuario ha regresado al hogar o a la oficina.

Además de contar con una interface en serie RS232 para conectar a una impresora, el Microwriter está equipado con una interface para cassette, que permite guardar permanentemente o volver a cargar el texto almacenado en su memoria. Asimismo, la interface en serie permite que el Microwriter se pueda utilizar como una terminal manejable con una sola mano de un ordenador o un procesador de textos normales. Los documentos digitados fuera del hogar o la oficina se pueden cargar en un sistema de tamaño natural para una edición o una manipulación más completas.

El texto dentro del Microwriter se puede dividir en documentos separados, y ello permite dar entrada o manipular por separado diversos bloques de texto sin relación entre sí. Existen sencillos resortes para agregar o eliminar texto y hasta desplazar grandes bloques utilizando una cassette como memoria intermedia.

Fue intención del diseñador que el teclado de seis teclas del Microwriter pudiera incorporarse a otros dispositivos electrónicos. No obstante y a pesar de sus muchas cualidades, el Microwriter sólo ha tenido una aceptación limitada y aún está por ver si los fabricantes de ordenadores personales se hacen eco de la idea.



Interruptor
Al encenderlo o apagarlo no se pierden datos y se puede continuar con la escritura del mismo documento

Cristal del reloj

Enchufe red
Para recargar o para funcionar en conexión a la red eléctrica con un transformador externo

CPU
Tanto la CPU como la RAM son dispositivos CMOS (Complementary Metal Oxide Silicon) para reducir el consumo energético

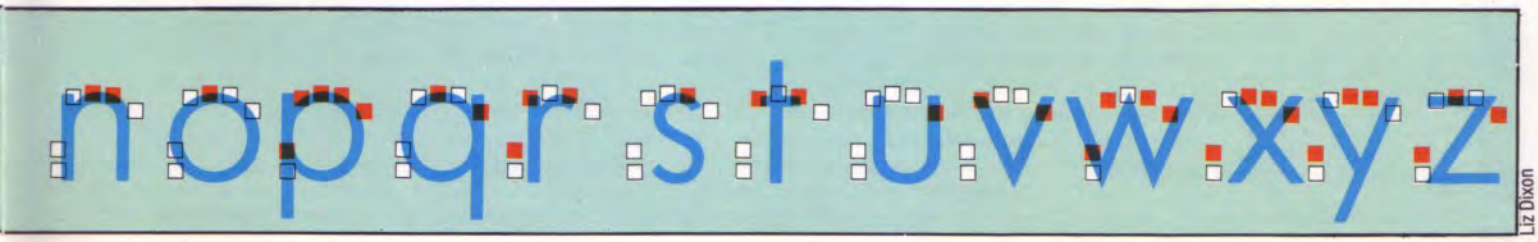
Pilas de Ni-Cad
Estas pilas de níquel-cadmio se recargan mediante un transformador externo

Interface para ampliación
En vistas a una futura ampliación, esta puerta incluye las líneas de dirección y de datos del microprocesador

EPROM
El programa de tratamiento de textos y el sofisticado software de comunicaciones están incorporados en una única EPROM, cuya fabricación en pequeñas cantidades resulta más económica que una ROM

Grupos de cinco
La documentación del Microwriter incluye notas mnemotécnicas e ilustraciones para ayudar al usuario a aprender las distintas combinaciones de teclas necesarias para crear el alfabeto. La sexta tecla se utiliza en combinación con las otras para proporcionar las órdenes de puntuación y edición

Steve Cross



Liz Dixon

Búsqueda binaria

El tiempo necesario para localizar un registro se reduce mucho merced a la "búsqueda binaria", si el archivo ya se ha clasificado en el orden adecuado

Las tres actividades más importantes del programa de la agenda de direcciones (agregar registros nuevos, guardar el archivo en cinta o disco y leer el archivo de datos cuando se ejecuta por primera vez el programa) ya las hemos desarrollado. Pero una agenda de direcciones no sirve de nada si usted sólo puede agregar información irrecuperable. Se necesita, pues, una rutina para hallar un registro.

Probablemente la actividad más frecuente será hallar el registro completo de un nombre, y ése es el motivo por el cual la primera opción del menú de opciones (*ELECCN*) es HALLAR REGISTRO (DE UN NOMBRE). Buscar es una actividad muy importante en la mayoría de los programas para ordenadores, en especial en los programas para bases de datos donde a menudo se necesita recuperar datos de un archivo. En términos generales, existen dos métodos de búsqueda: el lineal y el binario. El lineal observa cada uno de los elementos de una matriz, empezando desde el principio, y continúa hasta encontrar el dato deseado. Si los datos de la matriz están sin clasificar, la única búsqueda que ofrece garantías de funcionamiento es la lineal. El tiempo necesario para localizar el dato utilizando una búsqueda lineal en una matriz de N datos posee un valor promedio proporcional a N/2. Si son pocos los datos a través de los cuales se debe buscar, N/2 puede ser perfectamente aceptable; pero a medida que aumenta el número de datos, el tiempo que ocupa realizar la búsqueda también aumenta, hasta resultar excesivo.

No obstante, si se sabe que los datos del archivo ya están clasificados, existe un método de búsqueda mucho más eficaz que se denomina *búsqueda binaria*, y funciona de la siguiente manera. Supongamos que se desea hallar la definición de la palabra "lepidóptero" en el diccionario. Desde luego no comenzará por la primera página a ver si está allí y, de no hallarla, pasará a la segunda página y así sucesivamente a través de todo el diccionario hasta que encuentre la palabra deseada. Lo que hará es abrir el libro aproximadamente por la mitad, elegir una página y mirar lo que hay en ella. Si resulta que comienza por la palabra "metatarso", usted sabe que se ha pasado, de manera que la segunda mitad del libro es descartada, pues la palabra que busca está en algún lugar de la primera mitad. Volverá entonces a repetir el proceso, considerando ahora la página por donde abrió el diccionario la primera vez como si fuera la última página. Vuelve a dividir en dos esta parte del diccionario y a escoger una página, que ahora comienza con "involución". Esta vez deduce que la página seleccionada es demasiado "baja" y (a los fines de nuestra búsqueda de "lepidóptero") se puede considerar como si fuera la primera página: las anteriores son descartadas y se dice que son "bajas" en el sentido de que la *l* va después que la *i*. Ahora de "primera" y "última" páginas del diccionario pueden hacer aquellas que empiezan con "involución" y "metatarso" respectivamente. De nuevo abre por la mitad de la sección restante y encuentra "juncal" iniciando página. Una vez más es demasiado "baja", de modo que la palabra que estamos buscando debe estar entre ésta y la página de "metatarso". La suficiente repetición de este proceso es una garantía para localizar la palabra que estamos buscando (¡siempre que esté en el diccionario!).

En el ejemplo que acabamos de considerar, "lepidóptero" era la *clave de búsqueda*. La clave de búsqueda es la entrada que estamos intentando hallar. Cada vez que examinamos un registro, comparamos la clave de búsqueda con la *clave de registro* para localizar el "objetivo" o "víctima". Junto con la clave de registro esperamos hallar lo que se denomina *información adicional*, con la suficiente lógica. La información adicional para la clave de registro "lepidóptero" sería la definición que da el diccionario para la palabra, en este caso, insecto llamado corrientemente "mariposa".

La analogía es perfecta con la búsqueda de un registro-objetivo a través de un archivo en una base de datos, siempre y cuando los registros se hayan clasificado previamente como se han clasificado las entradas de un diccionario. ¡Piense lo difícil que sería utilizar un diccionario si las entradas estuvieran consignadas por el orden en que se le hubieran ocurrido al lexicógrafo!

La rutina de búsqueda requerida para nuestra agenda de direcciones resultará algo más complicada de lo que podríamos imaginar en un primer momento, por motivos que más adelante se harán evidentes. Lo primero que hará la rutina de búsqueda (de momento la llamaremos *BUSREG) será solicitar el nombre que ha de hallar. Es la clave de búsqueda. Supongamos que en algún lugar del archivo hay un registro para una persona llamada María Gómez. El registro para esta persona tendrá un campo (con el nombre en forma estandarizada), que contenga GOMEZ MARIA. La rutina de búsqueda podría presentarse con un mensaje como ¿A QUIEN ESTA BUSCANDO? y nosotros le responderíamos MARIA GOMEZ, o quizá M. GOMEZ o Mari Gómez. Antes de que todo esto se complique demasiado, vamos a suponer que nosotros le respondemos con el nombre completo, María Gómez. Lo primero que hará la rutina de búsqueda será adaptar esta respuesta a la forma estandarizada, GOMEZ MARIA. A continuación, comparará nuestra entrada, la clave de búsqueda, con los diversos contenidos de los campos MODNOM$. Si el programa estuviera utilizando una búsqueda lineal, la clave de búsqueda se compararía, de forma secuencial, con cada campo MODNOM$, hasta que se encontrara un nombre igual o se descubriera que no hay otro igual.

Sin embargo, como ya hemos apuntado, una búsqueda lineal no es eficaz en comparación con una búsqueda binaria cuando los datos ya están cla-

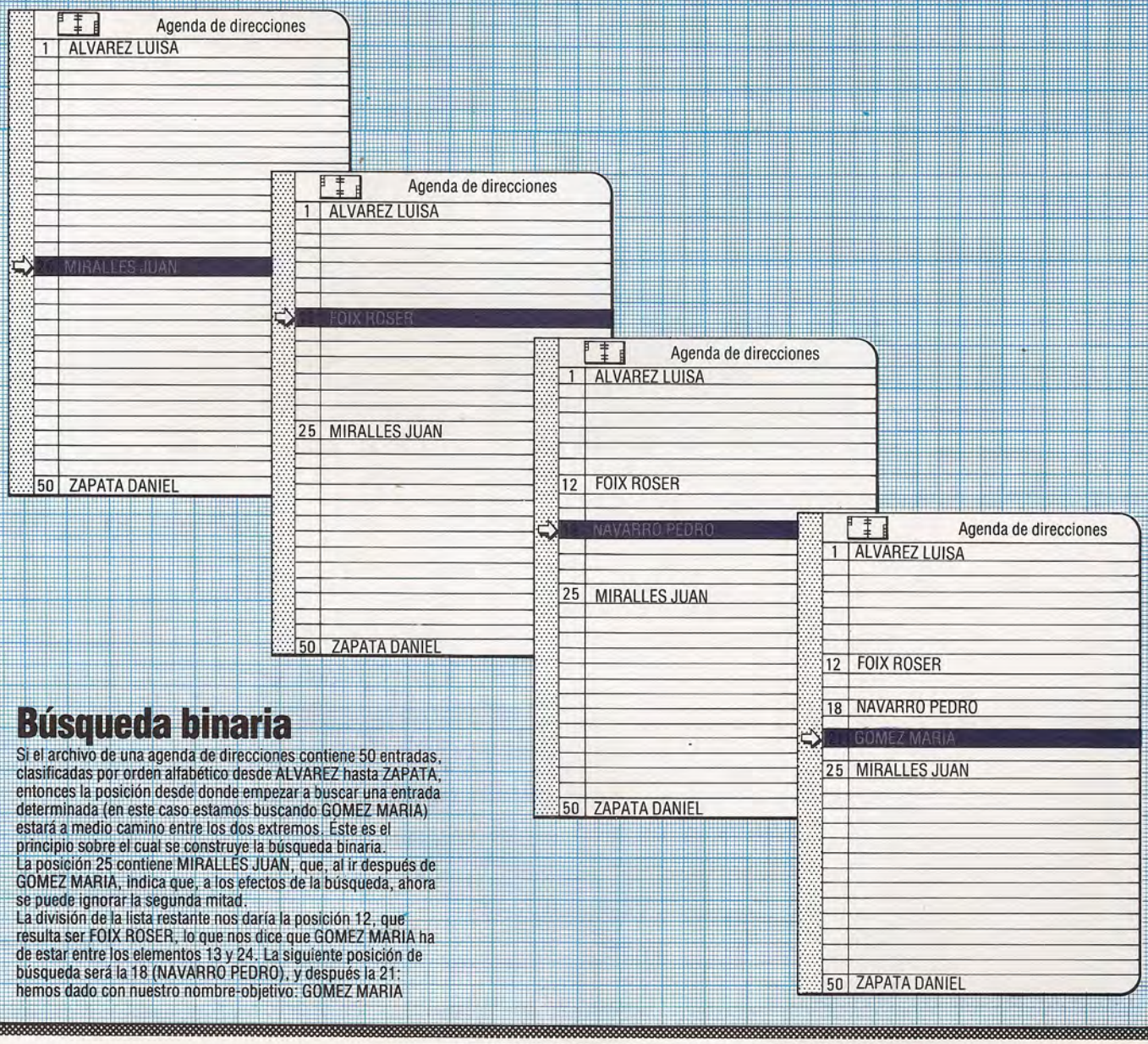
sificados. La rutina de búsqueda puede asegurar que los registros estén clasificados empezando con una IF RMOD = 1 THEN GOSUB *BUSREG*. El programa sabe que el elemento más bajo de la matriz a buscar será MODCAM$(1) y que el más alto será MODCAM$(TAMA-1). Para conducir la búsqueda necesitaremos tres variables: INF, para la parte inferior de la matriz (MODCAM$(1) al principio); SUP, para la parte superior de la matriz (MODCAM$(TAMA-1) al principio); y MED, para el valor correspondiente al elemento medio.

Ahora, utilizando de nuevo la analogía del diccionario, podemos suponer que INF = MATRIZ(1) y SUP = MATRIZ(TAMA-1). En otras palabras, la matriz que hemos considerado para la búsqueda empieza por el elemento "más grande". Por consiguiente, podemos dejar que LET INF = 1 y LET SUP = TAMA-1 (recuerde que TAMA siempre es mayor en uno que el número de registros que existe habitualmente en la agenda de direcciones).

Supongamos que en la agenda de direcciones hay 21 entradas válidas. TAMA tendrá un valor de 22. INF tendrá un valor de 1. SUP tendrá un valor de 21. El valor de MED, es decir, la posición del elemento medio, se puede obtener en BASIC mediante INT((INF + SUP)/2); si el valor de INF es 1, y el valor de SUP es 21, el valor de MED será, entonces, 11.

Para realizar una búsqueda binaria, primero suponemos que todo el archivo es válido y hallamos el punto medio INT(INF + SUP)/2) dentro de un bucle que se termina si se ha hallado el objetivo o bien si no existe igualdad. Luego verificamos para ver si la clave de búsqueda (BUSCLV$) resulta ser igual al valor MED de la matriz. Si el valor MED de la matriz es demasiado pequeño, sabemos que MATRIZ(MED) es la parte más baja de la matriz que necesitamos considerar, de modo que INF se podría establecer en MED. Sin embargo, es ligeramente más eficaz establecer INF en MED + 1, dado que ya sabemos que MATRIZ(MED) no es igual a la clave de

Búsqueda en la agenda de direcciones



Búsqueda binaria

Si el archivo de una agenda de direcciones contiene 50 entradas, clasificadas por orden alfabético desde ALVAREZ hasta ZAPATA, entonces la posición desde donde empezar a buscar una entrada determinada (en este caso estamos buscando GOMEZ MARIA) estará a medio camino entre los dos extremos. Este es el principio sobre el cual se construye la búsqueda binaria.
La posición 25 contiene MIRALLES JUAN, que, al ir después de GOMEZ MARIA, indica que, a los efectos de la búsqueda, ahora se puede ignorar la segunda mitad.
La división de la lista restante nos daría la posición 12, que resulta ser FOIX ROSER, lo que nos dice que GOMEZ MARIA ha de estar entre los elementos 13 y 24. La siguiente posición de búsqueda será la 18 (NAVARRO PEDRO), y después la 21: hemos dado con nuestro nombre-objetivo: GOMEZ MARIA

búsqueda. Del mismo modo, por otra parte, SI MATRIZ(MED) > BUSCLV$, SUP se establecería en MED – 1.

Como paso provisional hacia el desarrollo de una rutina que funcione por completo, el programa mostrado puede tomar una entrada ficticia (que necesita estar exactamente en el mismo formato que los campos de MODCAM$), o bien imprimir NO HALLADO REGISTRO si no hubiera encontrado igualdad o EL NUMERO DE REGISTRO ES (MED) si la hubiera hallado. Como la rutina empieza con el número de línea 13000, se puede agregar al final del programa tal como lo presentamos en la p. 399, y funcionará siempre que la línea 4040 se cambie por IF OPCN = 1 THEN GOSUB 13000.

La línea 13240 contiene la sentencia STOP. Ésta interrumpirá temporalmente el programa tan pronto como se visualicen los mensajes NO HALLADO REGISTRO o EL NUMERO DE REGISTRO ES (MED). El programa se puede reemprender por el mismo número de línea, sin pérdida de datos, digitando CONT. Sin STOP, el programa se apresuraría a la sentencia RETURN de la línea 13250 y el mensaje aparecería durante un lapso demasiado breve como para resultar legible.

Consideremos con mayor detalle este fragmento de programa. La línea 13100 establece INF en 1, la posición del menor elemento de la matriz MODCAM$. SUP se establece en TAMA-1 en la línea 13110. Ésta es la posición de las matrices MODCAM$ donde está situado el elemento mayor. La línea 13120 inicializa un bucle que sólo se terminará cuando se encuentre una igual o bien cuando se sepa que no existe igualdad alguna.

La línea 13130 halla el punto medio de la matriz dividiendo por dos la suma del índice inferior y superior de la matriz (se emplea INT para redondear la división, de modo que MED no pueda asumir un valor como 1,5). Existe la posibilidad de que el contenido de MODCAM$ (MED) sea el mismo que la clave de búsqueda (BUSCLV$); pero si no lo es, como es probable que ocurra, L se establecerá en 0, asegurando que el bucle se repita. Si fracasa la condición de la línea 13140, MODCAM$ (MED) tendrá un valor menor o mayor que BUSCLV$. El valor de INF se establecerá entonces en uno más que el antiguo valor de MED (línea 13150), o se establecerá el valor de SUP en uno menos que el antiguo valor de MED. La razón por la cual no se utiliza el propio valor de MED es que el fracaso de la condición de la línea 13140 ya ha demostrado que MODCAM$(MED) no es el objetivo que estamos buscando y que no tiene sentido volver a examinar ese elemento de la matriz en el próximo ciclo del bucle.

De no hallarse ninguna igualdad, el valor de INF finalmente superará al valor de SUP. El bucle se puede terminar (línea 13170) e imprimirse el mensaje NO HALLADO REGISTRO (línea 13200).

Este fragmento de programa se ofrece con fines de demostración y para permitir la verificación de la rutina de búsqueda. Tal como está, su utilización es más bien limitada. Sin el STOP de la línea 13240 no tendríamos tiempo ni siquiera para ver el mensaje centelleando en la pantalla. Lo que se necesita es una visualización del registro completo, tal como se digitó originalmente. Una vez que se conoce el número del registro resulta sencillo recuperar cualquier información adicional que se necesite: NOMCAM$, CLLCAM$, etc. Debajo de la visualización del registro probablemente desearíamos un mensaje como PULSE BARRA ESPACIADORA PARA CONTINUAR (regreso al menú principal) y quizá otras opciones como PULSE "I" PARA IMPRIMIR

Lamentablemente, ya no es tan fácil decidir cómo manipular la entrada de *ENCREG*. En el fragmento de programa, la entrada esperada (en la línea 13020) ha de estar en la forma estandarizada: GOMEZ MARIA, por ejemplo. Esto, sin duda, no es todo lo conveniente que sería de desear. La gente no piensa los nombres en orden inverso, y tener que darles entrada en letras mayúsculas representa una incomodidad para el usuario. Además, a la más mínima desviación respecto al nombre al que originalmente se dio entrada nos encontraríamos con NO HALLADO REGISTRO. Cabría esperar que los dos primeros problemas los manipulara *MODNOM*. El tercer problema, el de cómo hacer frente a una pareja aproximada, es muchísimo más interesante pero, al mismo tiempo, mucho más difícil de resolver.

Antes de considerar este problema, veamos por qué *MODNOM* no solucionará los dos primeros problemas. Si usted vuelve hacia atrás y analiza *MODNOM*, que empieza en la línea 10200, descubrirá una buena ilustración de una de las trampas más comunes en las que caen los programadores: falta de generalidad. Esta subrutina debería ser capaz de manipular conversiones entre nombres "normales" y nombres "estandarizados" cada vez que fuera necesaria esta operación. Aun cuando se escribió como una subrutina separada, se hizo pensando claramente en *INCREG*. Da por sentado que el nombre a convertir siempre residirá en NOMCAM$(TAMA) y que después de la conversión el nombre modificado siempre se almacenará en MODCAM$(TAMA). Frente a esta situación, el programa tiene tres opciones: reescribir *MODNOM* por completo para hacerla general, lo que a su vez implicaría introducir cambios en otras partes del programa; escribir una rutina casi idéntica sólo para manipular la entrada para *ENCREG*, lo que representaría realizar un esfuerzo injustificado y ocuparía más memoria; o recurrir a alguna mala técnica de programación para permitir que se pueda utilizar la rutina *MODNOM* sin modificarla. Esta última alternativa es, en varios sentidos, la menos conveniente. Solucionaría el problema, pero el verdadero funcionamiento de la parte del programa que se haya modificado probablemente no estaría claro ni siquiera para el programador, y sería una pesadilla para cualquier otra persona que intentara utilizar el programa.

La moraleja de la historia es: hacer que las subrutinas sean lo más generales posible, de modo que cualquier parte del programa las pueda llamar.

Para ilustrar una técnica de programación mala o, como a menudo se la suele llamar, programación "chapuza", y para mostrar lo poco claro que puede hacer que resulte el programa, consideremos la línea 13020 del fragmento de programa INPUT "DE ENTRADA A LA CLAVE"; BUSCLV$ y analicemos después la modificación o el "arreglo" que posibilitaría la utilización de *MODNOM*:

```
13020 INPUT "DE ENTRADA A LA
      CLAVE";NOMCAM$(TAMA)
13030 GOSUB 10200: REM SUBRUTINA
      *MODNOM*
13040 LET BUSCLV$ = MODCAM$(TAMA)
13050 ...
```

Afortunadamente, el valor de TAMA siempre es mayor en uno que el más alto de los registros válidos. En otras palabras, en las matrices no hay ningún registro en la posición TAMA, de manera que este arreglo no modificará ninguno de los registros existentes. Pero sin algunas otras observaciones REM que expliquen lo que está sucediendo, ¡imagínese lo confusas que le resultarían estas tres líneas a alguien que no hubiera estado implicado en el desarrollo del programa!

Volvamos a un problema más interesante, el de tratar con los "cuasi yerros". Supongamos que hemos dado entrada al nombre de alguien como Mari Gómez durante una operación de *INCREG*, pero como María Gómez durante *ENCREG*. Éstos se convertirían a las formas estandarizadas GOMEZ MARI y GOMEZ MARIA, respectivamente, y durante la búsqueda no se encontraría ningún emparejamiento, incluso aunque el registro que deseáramos estuviera allí. Nosotros no intentaremos resolver este problema, porque una solución satisfactoria representaría un gran trabajo de programación. No obstante, he aquí algunas indicaciones para aquellos lectores interesados en experimentar:

```
EMPEZAR {buscar matriz para pareja exacta}
  IF se halla igualdad
    THEN PRINT registro completo
    ELSE buscar matriz para pareja aproximada
      IF se halla pareja aproximada
        THEN PRINT registro de pareja aproximada
        ELSE PRINT "NO HALLADO REGISTRO"
      ENDIF
  ENDIF
END
```

El procedimiento para pareja aproximada se podría parecer a las siguientes líneas:

```
EMPEZAR pareja aproximada
  Buscar matriz para igualdad de apellido
  IF igualdad de apellido
    THEN buscar nombres de pila aproximados
    PRINT registro aproximado
    ELSE buscar apellidos aproximados
      IF se halla apellido aproximado
        THEN PRINT registro aproximado
      ENDIF
  ENDIF
END
```

El procedimiento para "aproximados" se podría definir en líneas generales como hallar la variable objetivo que posea el máximo número de caracteres en común con aquellos de la variable clave. Podría aceptar una situación en la que la variable clave estuviera totalmente contenida en la variable objetivo, o viceversa. No existen soluciones sencillas, pero el panorama es muy amplio para una programación emprendedora.

En el fragmento de programas presentado existe un "efecto colateral". Supongamos que se produjera la siguiente secuencia de acontecimientos. Usted ejecuta el programa y después utiliza *INCREG* para agregar un registro nuevo, seguido de *ENCREG* para localizar un registro. Cuando por último se ejecute *SAPROG*, para guardar el archivo y dar por terminado el programa, no se guardará el registro que usted agregó (aunque sí se guardarán todos los otros registros). Esto es consecuencia directa de algo que sucedió durante la ejecución de *ENCREG*. ¿Se da cuenta por qué no se guardará el registro agregado?

En el próximo capítulo de nuestro curso explicaremos cómo evitar esta pérdida de datos; mostraremos para qué se utiliza la variable CURS y describiremos cómo borrar o modificar un registro. Otras opciones del menú principal (*ENCCIU*, etc.) son bastante similares a las rutinas que ya hemos elaborado. Se deja a los propios lectores su creación, en el caso de que sean necesarias.

Por último, veamos qué sucedería si hubiera 50 registros en el archivo de datos y se usara la rutina *ENCREG* modificada (que llama a *MODNOM*). (Una pista: TAMA tendrá el valor 51.)

Complementos al BASIC

SPECTRUM

Modificaciones para el Spectrum:

```
13000 REM VERSION PARA PROBAR
      *ENCREG*
13010 IF RMOD = 1 THEN GOSUB 11200
13020 INPUT "DE ENTRADA A LA
      CLAVE";B$
13100 LET INF = 1
13110 LET SUP = TAMA - 1
13120 FOR L = 1 TO 1
13130 LET MED = INT((INF + SUP)/2)
13140 IF M$(MED) <> B$ THEN LET L = 0
13150 IF M$(MED) < B$ THEN LET
      INF = MED + 1
13160 IF M$(MED) > B$ THEN LET
      SUP = MED - 1
13170 IF INF > SUP THEN LET L = 1
13180 NEXT L
.
.
.
13200 IF INF > SUP THEN PRINT "NO
      HALLADO REGISTRO"
13210 IF INF < = SUP THEN PRINT "EL
      NUMERO DE REGISTRO ES  ";MED
.
.
.
13240 STOP
13250 RETURN
```

De nuevo el problema de las variables alfanuméricas de una letra en el Spectrum: aquí B$ se ha sustituido por BUSCLV$.

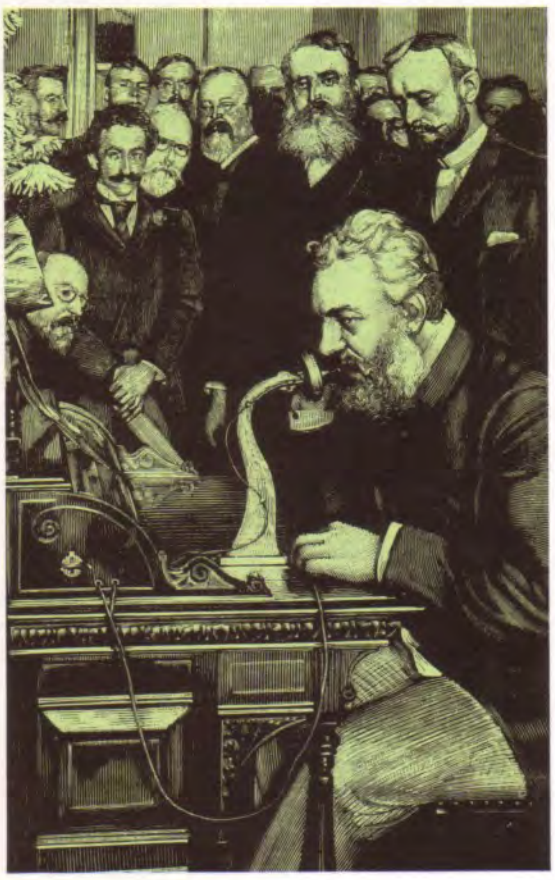
```
13000 REM VERSION DE *ENCREG* PARA PROBAR
13010 IF RMOD = 1 THEN GOSUB 11200
13020 INPUT "DE ENTRADA A LA CLAVE";BUSCLV$
13030 REM
13040 REM
13050 REM
13060 REM
13070 REM
13080 REM
13090 REM
13100 LET INF = 1
13110 LET SUP = TAMA-1
13120 FOR L = 1 TO 1
13130 LET MED = INT((INF+SUP)/2)
13140 IF MODCAM$(MED)<>BUSCLV$ THEN L = 0
13150 IF MODCAM$(MED)<BUSCLV$ THEN INF = MED+1
13160 IF MODCAM$(MED)>BUSCLV$ THEN SUP = MED-1
13170 IF INF>SUP THEN L = 1
13180 NEXT L
13190 REM
13200 IF INF>SUP THEN PRINT "NO HALLADO REGISTRO"
13210 IF INF<=SUP THEN PRINT "EL NUMERO DE REGISTRO ES ";MED
13220 REM
13230 REM
13240 STOP
13250 RETURN
```

Los Laboratorios Bell

En la historia del ordenador, este centro de investigación ha realizado numerosas aportaciones, tanto en el campo del hardware como en el del software

Hace cien años, la reina Victoria quedó encantada con un nuevo invento que le permitía hablar desde la isla de Wight con sus ministros, en Londres. El teléfono ha sido objeto de notables mejoras desde aquellos tiempos del aparato a manivela a través de la investigación y el desarrollo, y una de las consecuencias de este trabajo ha sido el ordenador. En los primeros años de la historia del teléfono, la American Telephone and Telegraph Company decidió crear una organización que investigara la forma de perfeccionar el sistema telefónico. Así nacieron en 1925 los Laboratorios Bell (apodados *Ma Bell*) en Murray Hill, Nueva Jersey (Estados Unidos).

Laboratorios Bell es una institución insólita porque se dedica exclusivamente a la investigación y, sin embargo, es propiedad de una compañía con fines lucrativos. A los científicos se les mantiene deliberadamente alejados de los problemas de ingeniería que plantea día a día el funcionamiento de una empresa de estas características, porque la Bell considera que la investigación supone una inversión rentable a largo plazo. Resumiremos aquí algunos aspectos de su labor, los más decisivos para el desarrollo del ordenador.



"¡Venga aquí, señor Watson: lo necesito!"
Los Laboratorios Bell han tomado su nombre de Alexander Graham Bell (1847-1922), a quien se atribuye la invención del teléfono, en el año 1876. Se dice que las primeras palabras que se transmitieron por medios eléctricos a través de cables fueron las que dirigió Bell a su ayudante, que estaba en el cuarto contiguo; dichas palabras fueron: *"Come here, Mr. Watson, I want you!"* ("¡Venga aquí, señor Watson: lo necesito!")

A mediados de la década de los treinta, los sistemas telefónicos se iban automatizando y haciéndose cada vez más complejos. Los mensajes se enviaban en modo analógico a través de los cables telefónicos y las llamadas se conectaban utilizando la información contenida en un código de discado digital. El número discado se convertía primero, en la central telefónica, de una señal analógica a una secuencia de impulsos digitales. Ésta se almacenaba temporalmente en una memoria compuesta por interruptores de relé hasta completar la conexión mediante un banco de interruptores en cruz. Éstos contaban los impulsos del código de discado y los convertían en coordenadas sobre un cuadro conmutador electromecánico. Todos los ingredientes de un ordenador ya estaban ahí: sólo esperaban a que la persona adecuada los recogiera.

George Stibitz, matemático que trabajaba en la Bell, observó la similitud entre los impulsos "contadores" y la suma de todos ellos juntos. Trabajando en su casa, sobre la mesa de la cocina, con algunos antiguos interruptores en cruz y relés electromecánicos, construyó los primeros circuitos de ordenador de relé.

Stibitz comenzó entonces a trabajar con un experimentado ingeniero de interruptores, Samuel B. Williams, que llevaba 25 años construyendo circuitos interruptores, y los dos hombres crearon una calculadora de números complejos (los números complejos comprenden los llamados números imaginarios, raíces cuadradas de números negativos, y son necesarios para obtener soluciones completas de una ecuación polinómica). Comenzaron la tarea en 1937, y el dispositivo consumió 450 relés y 10 interruptores en cruz. Operaba en notación binaria y podía dividir dos números de ocho dígitos en 30 segundos. La calculadora de números complejos entró en funcionamiento el 8 de enero de 1940, y en septiembre del mismo año se exhibió ante la American Mathematical Society, en el Dartmouth College (donde posteriormente se idearía el BASIC). La calculadora tenía capacidad de acceso remoto y múltiple a través de teclados de máquinas de escribir conectados con el mecanismo de cálculo de Nueva York mediante cables telefónicos. Impresionó al público, especialmente por su forma "humana" de funcionar: ¡después de que se le formulaba una pregunta a la calculadora, parecía hacer una pausa de algunos segundos, como pensando, antes de dar la respuesta!

Muchos dispositivos menores de hardware también se originaron en Bell, como los colchones de aire flotantes que se emplean en las cabezas de cinta magnética, y los amplificadores de feedback negativo. Pero el invento más famoso fue el transistor, que crearon Bardeen, Brattain y Shockley en 1947 (véase p. 47). Fue el transistor lo que hizo posible la segunda generación de ordenadores.

Diseños de calidad

El diseño auxiliado por ordenador pide difíciles cálculos para sus refinados productos. Muchos de sus principios son aplicables por ordenadores personales

Cortesía de Soft

Ian McKinnell

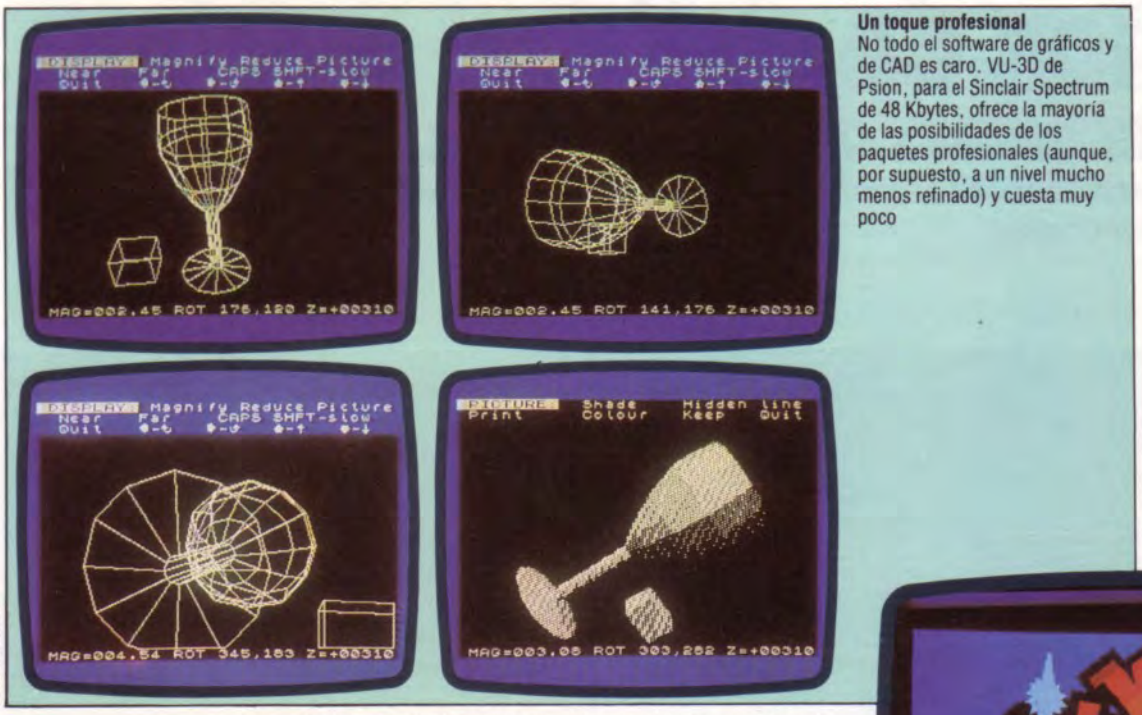
La flor del manzano
Por muy sofisticado que sea el software, el elemento más importante de un paquete de diseño auxiliado por ordenador es el diseñador que lo utiliza. Versawriter, cuyos resultados vemos en la fotografía, funciona en un Apple II y exige dos unidades de disco y un digitalizador. Un usuario experimentado tardó cierto tiempo en dar entrada a la flor en la máquina

La idea de aplicar el ordenador al proceso del diseño industrial surgió por primera vez en la década de los sesenta (en el Massachusetts Institute of Technology). No obstante, habría de transcurrir una década más hasta que la tecnología del ordenador le permitiera al diseñador ver una representación gráfica de su trabajo e interactuar con ella. Una vez presente en la pantalla de un monitor, se accede a su modificación mediante un digitalizador o un lápiz óptico, exactamente igual que si se estuviera frente a un tablero de dibujo. Estos periféricos esenciales (el digitalizador, el lápiz óptico y el plotter) son las herramientas básicas del arte del diseño auxiliado por ordenador. Con ellos se pueden crear imágenes de la misma forma como las crea un animador (véase p. 181), "dibujando" en una tablilla de digitalización. El diseñador puede modificar dicha imagen, tal vez utilizando un lápiz óptico, incorporando componentes predibujados y subensamblajes, y después sacar una copia del dibujo acabado con un plotter. El ordenador se convierte en un sistema de delineación en principio similar a un procesador de textos, pero trabajando con imágenes en lugar de letras.

Hemos visto que la calidad de la imagen producida en el monitor de un ordenador depende de dos cosas: de la capacidad de resolución del monitor (es decir, las dimensiones de un elemento o pixel individual de la imagen) y de la potencia y las dimensiones de la memoria del ordenador que acciona el monitor. Cuando analizamos las imágenes generadas por ordenador, nos encontramos en gran medida con las mismas exigencias: un monitor que pudiera resolver algo así como 1 000 × 1 200 pixels, y un ordenador capaz de procesar estos 1,2 millones de elementos de la imagen en menos de 1/24 segundos.

Nos será útil continuar con la analogía entre el paquete para diseño auxiliado por ordenador y el procesador de textos. En lugar de trasladar párrafos, oraciones o palabras individuales a través de un trozo de texto (corrigiendo, insertando o borrándolos a voluntad) el programa CAD (*Computer Assisted Design*: diseño auxiliado por ordenador) se puede emplear para trasladar a través de la página los elementos de un dibujo. El efecto podrá ser diferente, pero el principio es exactamente el mismo.

Desde el punto de vista del programa, se trata de cómo almacenar esta imagen de forma tal que se la pueda alterar y manipular. Si fuéramos a realizar el modelo físico de un objeto, utilizaríamos uno de dos métodos básicos: aditivo o sustractivo. El método aditivo es como modelar en arcilla: se va construyendo el objeto pieza tras pieza hasta que llegamos a la forma final. El método sustractivo se parece al que siguen los escultores, que sacan la obra

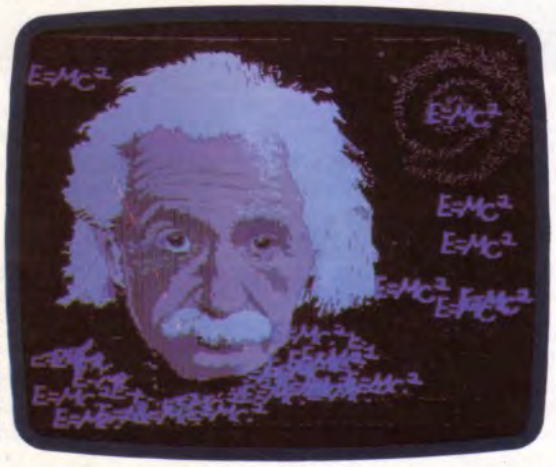


Un toque profesional
No todo el software de gráficos y de CAD es caro. VU-3D de Psion, para el Sinclair Spectrum de 48 Kbytes, ofrece la mayoría de las posibilidades de los paquetes profesionales (aunque, por supuesto, a un nivel mucho menos refinado) y cuesta muy poco

Ian McKinnell

desbrozando la materia. Por analogía, en el ordenador el bloque de materia sólida es una matriz tridimensional. Por consiguiente, adquieren importancia las dimensiones y el rendimiento del ordenador utilizado. Si la matriz es lo suficientemente grande como para permitir que se destine todo un byte para la definición de cada pixel o elemento de la imagen, entonces la cantidad de información que podremos retener acerca de cada elemento es bastante grande (256 partes separadas al emplear un procesador de ocho bits, y muchas más todavía para los dispositivos de 16 o 32 bits). Pero los problemas que plantea la creación de tanto espacio de almacenamiento son prácticamente insolubles, de modo que nos vemos obligados a asumir un compromiso, que por lo general es bastante aceptable. En lugar de destinar un byte entero para cada elemento, es suficiente destinar un único bit, si todo lo que deseamos hacer es indicar la presencia o la ausencia de un elemento en esta posición del modelo.

Fórmula eficiente
El sistema Pluto, de Io Research, lleva la generación de imágenes en alta resolución a una amplia gama de pequeños microordenadores, añadiendo, además, un procesador veloz y memoria extra. El sistema básico no es excesivamente caro y proporciona ocho colores fijos y un poder de resolución de 670 × 576 pixels



Cortesía de Io Research

El software para diseño auxiliado por ordenador comparte muchas de las características de los paquetes de imágenes generadas por ordenador: atenuación de curvas, eliminación de elementos ocultos, sombreado, relleno y recoloreado de zonas, por ejemplo. Para formar una curva sólo se requiere la solución repetida de una ecuación simple para una serie de valores. Especificando los puntos de comienzo y final para una línea determinada, y la distancia máxima que alcanzará la curva respecto a dicha línea recta, proporcionamos una solución para la ecuación. A partir de esta solución podemos trabajar a la inversa, para deducir la ecuación en sí misma, y después proceder a resolverla para el resto de la serie de valores, formando así la curva.

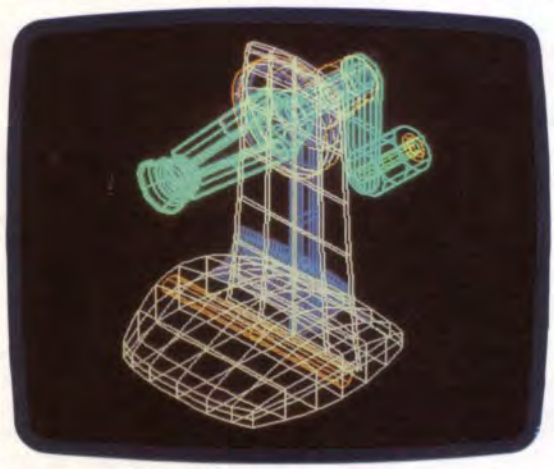
Esta capacidad para componer un dibujo a partir de partes estándar de los componentes es el verdadero punto fuerte de los sistemas CAD. Ya no es necesario volver a dibujar los componentes individuales comunes. Una vez que éstos se han defini-

do, se puede volver a llamar dicha definición cada vez que se la requiera e incorporarla a nuevos dibujos. Un ejemplo destacable de esto es el uso de ordenadores para auxiliar el diseño de las futuras generaciones de ordenadores.

El diseño de un circuito impresor, por ejemplo, es muy complicado, por lo que requiere la aplicación de técnicas de optimización para acomodar los componentes y sus pasos de interconexión de la forma más económica posible (teniendo en cuenta que las vías de conexión nunca se han de cruzar entre sí). El diseñador a menudo debe remitirse al ensayo y error, y es aquí donde los paquetes CAD son especialmente útiles. Todos los componentes individuales se almacenan como imágenes predefinidas y se les llama cuando se necesitan. Resulta muy sencillo probar un diseño determinado en la unidad de visualización de datos, para ver si responde a todas las exigencias, antes de trasladarlo al papel como parte de un dibujo en ejecución. Siguiendo este método se puede esbozar un diseño e incluso probar diferentes soluciones, empleando el tiempo que llevaría completar un boceto.

Los circuitos integrados se diseñan casi exactamente de la misma manera, pero debido a la densidad de componentes y vías de conexión, se requiere además otra configuración de software: la capacidad para ampliar una parte del dibujo, trabajar en él a escala aumentada y luego devolverlo a su posición dentro del diseño global. Este efecto es hoy una pieza básica del repertorio CAD y ha representado una considerable mejora en cuanto a la eficacia del sistema. Mediante su utilización se puede retener la especificación de un objeto completo en un solo dibujo, y se puede ajustar la escala para satisfacer las necesidades de quien lo mira.

Y no es sólo la escala lo único que se puede variar de este modo. Si consideramos el caso de un



Cortesía de Applicon

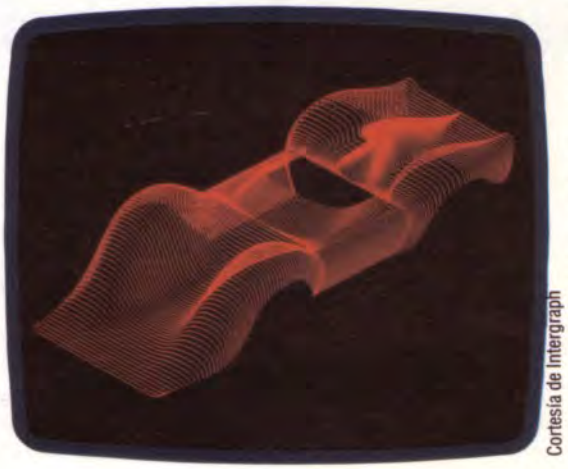
objeto más complejo (un coche, p. ej.), estamos ante múltiples subsistemas que, juntos, conforman el todo: el sistema eléctrico, el sistema hidráulico, el de escape, la suspensión, etc. Mientras que al diseñador de estética le interesará más el paquete global, a los ingenieros individuales probablemente lo que más les interese sea un solo subsistema. Es muy sencillo hacer que cada subsistema vaya en un color distinto para extraer después del dibujo global, todos los objetos de un color determinado. Esto no equivale a afirmar que el dibujo siempre ha de estar constituido por una mezcla de colores; la codificación se puede suprimir a voluntad cuando no se la considere necesaria.

Cortesía de Siggraph

Técnica e imaginación
El fondo de esta vista de la producción de Lucasfilms *Road to Point Reyes* se compuso en gran medida mediante "fractals", una nueva e ingeniosa técnica de CAD. Los "fractals" son fenómenos cuya complejidad aumenta según vamos contemplándolos más cerca. Las colinas y las montañas que aparecen al fondo se crearon como polígonos simples, descritos dentro de la memoria de un ordenador. Luego, cada polígono se fue haciendo progresivamente más complejo mediante la adición de su propia forma a cada uno de sus lados, repitiéndose el proceso con un cierto grado de aleatoriedad. El desarrollo de la forma de un copo de nieve, que vemos abajo, a partir de un simple triángulo, sirve para ilustrar este principio

El avance más sensacional es la capacidad para retener la especificación *completa* de un objeto (no sólo su forma y su aspecto, sino también información relativa al material en el que está construido, su peso, costo, etc.). Recabar información acerca de la forma y el tamaño del objeto es sólo una función del sistema, que se puede considerar como una base de datos orientada hacia la visualización. Formulando distintas preguntas a dicha base de datos se pueden ordenar pedidos a los proveedores, planificar la fabricación de subensamblajes y componentes, integrar las cadenas de producción para asegurar que los componentes lleguen exactamente a donde y cuando se les necesita, analizar costos y controlar la eficacia de la fabricación, entre otras muchas cosas. Estamos tentados de imaginar que el paso siguiente será un sistema regular de control directo de la fabricación por ordenador.

La mayoría de las aplicaciones que hemos analizado aquí requieren ordenadores de unidad principal o bien miniordenadores muy potentes, pero ello no quiere decir que incluso los microordenadores pequeños no puedan desempeñar un papel útil en el proceso de diseño. Existe una amplia gama de software para CAD destinado a máquinas que trabajan bajo CP/M, por ejemplo, y la mayoría de los fabricantes ofrecen al menos un paquete, incluso para ordenadores tan poco sofisticados, relativamente, como el Sinclair ZX81. Como hemos apuntado, el tamaño y la velocidad del ordenador determinan la calidad de la imagen almacenada, pero es poco probable que las necesidades del usuario de un ordenador personal sean las de un diseñador profesional, de modo que es factible conseguir brillantes resultados por un desembolso modesto.



Cortesía de Intergraph

Estructura básica
La primera etapa en la creación de una imagen o diseño tridimensional se conoce como "estructuración lineal". La imagen se define como una serie de coordenadas de puntos, unidas adecuadamente mediante líneas rectas. Estas líneas se pueden manipular utilizando el algoritmo de suavización de curvas, eliminando las líneas ocultas y después rellenando con color y sombreando, según sea necesario, para aumentar la ilusión de profundidad

La cinta teórica

La máquina Turing es un dispositivo puramente teórico, que se utiliza para decidir si un problema es informatizable o no

En *Mi Computer* solemos seleccionar temas prácticos y cosas que usted puede hacer con su ordenador personal. Sin embargo, en este apartado vamos a echar una mirada al lado teórico de los ordenadores: al campo que se denomina *ciencia de los ordenadores*. Tal ciencia es a la informática lo que las matemáticas puras son a la ingeniería, es decir, un campo que se caracteriza por ser extremadamente teórico pero del cual derivan las ideas prácticas.

La máquina Turing, por ejemplo, es una idea puramente teórica que desarrolló Alan Turing (véase p. 200) como ayuda para estudiar los algoritmos y la posibilidad de informatización. Se trata en realidad del "mínimo ordenador posible", de modo que si se puede demostrar que un problema determinado no se puede resolver utilizando una máquina Turing, entonces se puede afirmar que dicho problema no es "informatizable". Turing pensó que un ordenador mínimo de este tipo necesitaría tres configuraciones: un almacenamiento externo para grabar y almacenar la información de entrada y de salida; un medio para leer dicho almacenamiento y para escribir en él, y en tercer lugar una unidad de control que permitiese determinar las acciones a emprender.

Por definición, una máquina Turing posee, entonces, una cinta (si le sirve de ayuda, imagínesela como una cinta magnética) de longitud infinita (es decir: sea cual fuere la cantidad de cinta necesaria para solucionar un problema, siempre habrá la suficiente). La cinta está dividida en cuadrados, que o bien estarán en blanco o contendrán un símbolo. A lo largo de la cinta se mueve una "cabeza", que puede leer o escribir los símbolos en los cuadrados y que recibe sus instrucciones desde una unidad de control que le proporciona una doble indicación: cuáles son los símbolos que debe escribir y en qué dirección se ha de mover a continuación.

La unidad de control posee un programa de ejecución, y en este sentido se puede decir que cada máquina Turing se "fabrica" específicamente para realizar una aplicación, ya que en la especificación no existen medios para cargar o alterar un programa. Hemos entrecomillado la palabra fabricar porque las únicas máquinas Turing que se han construido físicamente lo fueron con fines puramente educativos. Sin embargo, escribir un programa en BASIC que imite el funcionamiento de una máquina Turing en un ordenador personal es un ejercicio relativamente sencillo.

El programa de control de una máquina Turing consta de un conjunto de "quíntuplos", o sentencias que contienen cinco elementos. Qué quíntuplo se ejecuta en cada etapa depende de dos factores: el símbolo que contenga el cuadrado que esté debajo de la cabeza de cinta, y el "estado" o "condición" de la máquina. Dicho estado es una cualidad estrictamente arbitraria: podemos especificar que la máquina se ponga en marcha en el estado S_A y que cuando llegue al estado especial H entonces se detenga, dándose por acabado el cálculo. Entretanto, el estado cambiará muchas veces de acuerdo con las instrucciones de los "quíntuplos". El estado tan sólo refleja lo que ha sucedido hasta el momento en el cálculo y ayuda a seleccionar qué "quíntuplo" se ejecutará a continuación (nuevamente, si le sirve de ayuda, imagíneselo como una variable de bandera en programación BASIC).

Los cinco elementos de que consta cada quíntuplo son:

1) El estado en curso de la máquina;
2) El símbolo del cuadrado de la cinta que se halle debajo de la cabeza;
3) El símbolo a escribir en ese cuadrado, que será el mismo que en 2) si no se requiere ninguna modificación de los datos;
4) El estado en que la máquina ha de entrar ahora;
5) La dirección en la que se debe mover la cabeza de cinta (izquierda o derecha).

El "quíntuplo" (S_A,5,3,S_B,D), por ejemplo, se ejecutará siempre que la máquina esté en estado S_A y que la cabeza de cinta lea un 5. El 5 se reemplazará luego por un 3, la máquina pasará del estado S_A al S_B y la cabeza de cinta se desplazará un cuadrado hacia la derecha.

Diseñar una máquina Turing teórica para efectuar una tarea determinada implica especificar el formato en el cual se le presentarán a la máquina sus datos de entrada en cinta, el formato de los datos de salida en cinta cuando se termine el cálculo (es decir, con la máquina en estado H), y el grupo de quíntuplos requeridos para ejecutar el algoritmo.

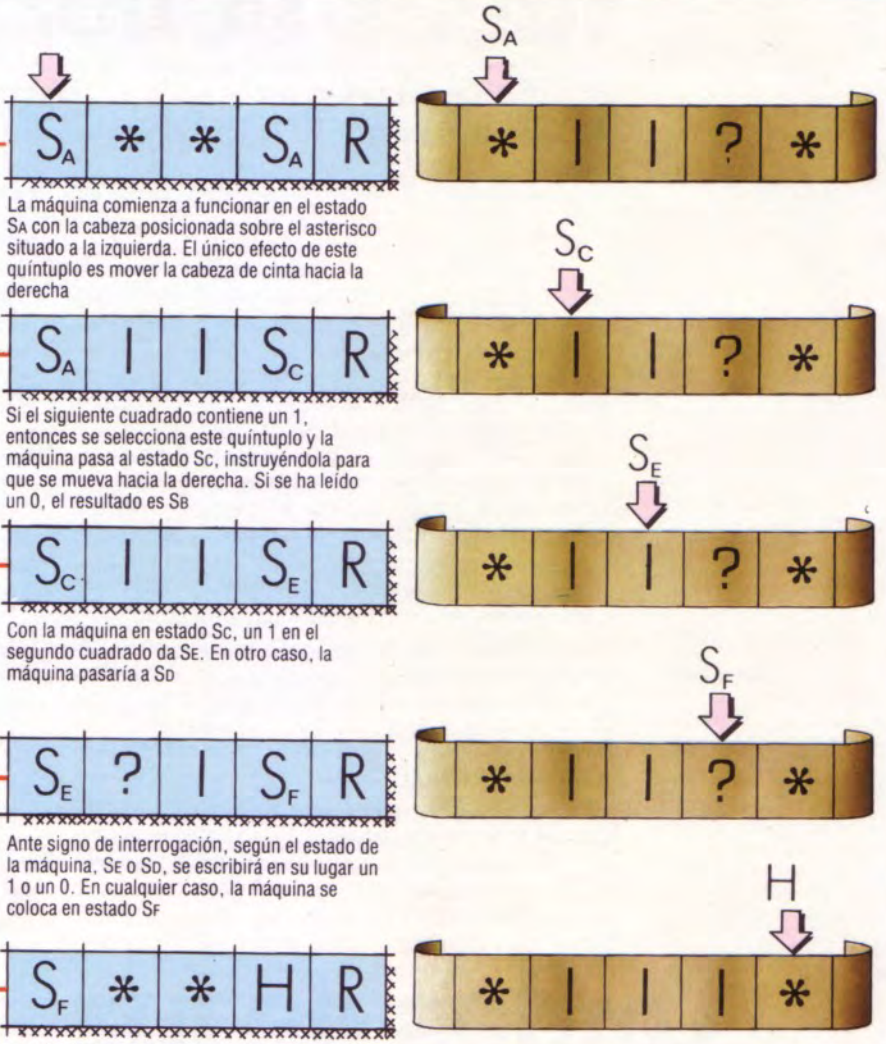
En el recuadro hemos diseñado una máquina Turing para realizar la función AND. Estableceremos los dos bits de entrada (cada uno un 1 o un 0) en cuadrados adyacentes, seguidos de un signo de interrogación, que se ha de reemplazar por la respuesta (nuevamente, un 1 o un 0, dependiendo de las dos entradas). Por conveniencia, hemos agregado un asterisco a los extremos del área de datos, y pondremos la máquina en funcionamiento en el estado S_A encima del asterisco situado más a la izquierda, acabando sobre el situado más a la derecha.

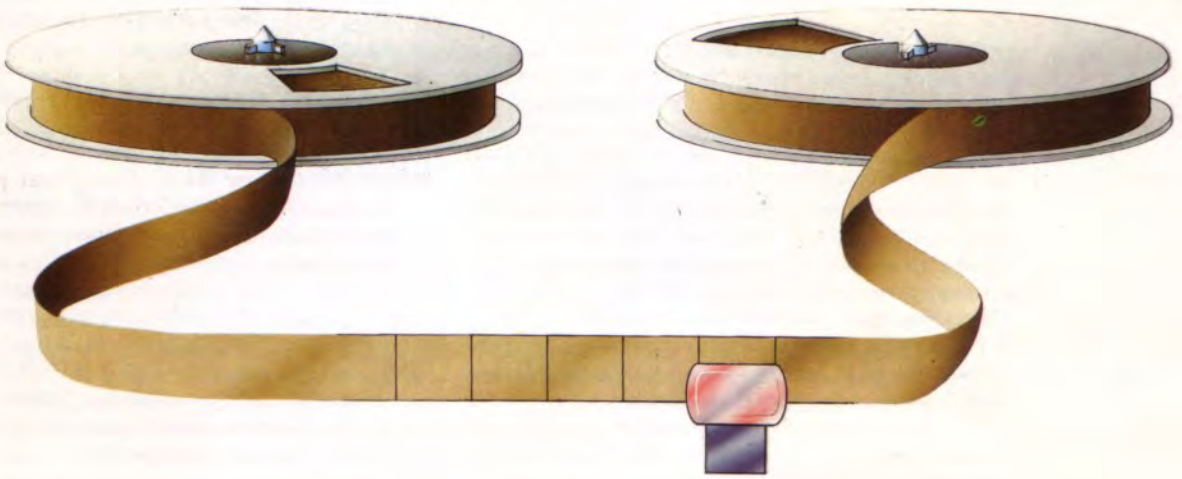
Se necesita un total de diez quíntuplos para especificar esta máquina, si bien, como se podrá ver en el ejemplo con que trabajamos (1 AND 1 = 1) se utilizan sólo cinco para cualquier ejecución. Si usted prueba la misma máquina para, supongamos, 0 AND 1, descubrirá que de los diez quíntuplos se selecciona un grupo de quíntuplos diferente.

La máquina Turing

Este ejemplo muestra la construcción de una máquina Turing para realizar la función AND. Los dos bits de entrada están establecidos en cuadrados adyacentes, seguidos por un signo de interrogación que será reemplazado por el resultado. Flanquean el área de datos dos asteriscos para que actúen como bordes. Los diez quíntuplos reseñados abajo especifican la operación de esta máquina, aunque para cualquier ejemplo con el cual se trabaje (en este caso 1 AND 1), sólo se utilizarán cinco de los diez

S_A	*	*	S_A	R
S_A	0	0	S_B	R
S_A	1	1	S_C	R
S_B	0	0	S_D	R
S_B	1	1	S_D	R
S_C	0	0	S_D	R
S_C	1	1	S_E	R
S_D	?	0	S_F	R
S_E	?	1	S_F	R
S_F	*	*	H	R





Kevin Jones

Ideas sonoras

Un ulterior análisis de la sofisticada orden ENVELOPE del BBC Micro

En el capítulo anterior de *Sonido y luz* presentamos la orden ENVELOPE del BBC Micro. Es una de las órdenes más poderosas de que dispone el programador de BASIC, cuando se utiliza junto con la orden SOUND, analizada en la p. 358. Vamos a ahondar nuestra explicación de ENVELOPE viendo qué es una *envoltura de volumen*.

En la siguiente línea de parámetros, los N se refieren a la envoltura de tono que ya conocemos (véase p. 408):

```
ENVELOPE N,T,PS1,PS2,PS3,NS1,NS2,NS3,AR,DR,
SR,RR,FAL,FDL
```

Los restantes parámetros aluden a la envoltura de volumen, estableciendo los volúmenes máximos y la velocidad de cambio de volumen en la ejecución de una nota requerida por la orden SOUND.

AR y DR (de −127 a 127); FAL y FDL (de 0 a 126)

AR (*Attack Rate*) establece la velocidad de subida de volumen. Aunque el software admite un valor negativo, en la práctica la escala va de 1 a 127. Indica el número de cambios de volumen por unidad de tiempo subiendo hasta alcanzar el FAL (*Final Attack Level*: máximo nivel de elevación), que da paso a la fase de apagamiento. La velocidad de apagamiento la controla DR (*Decay Rate*) de forma similar, pero con valores negativos, haciendo que el volumen decaiga hasta alcanzar el FDL (*Final Decay Level*: máximo nivel de apagamiento).

Aunque para los niveles máximos el software admite una escala de 0 a 126, el hardware normal sólo admite de 0 a 16, de modo que un valor FAL de 50 se adaptaría automáticamente a un volumen 6.

SR y RR (de −127 a 0)

La velocidad del sostenido (SR, *Sustain Rate*) y la velocidad del final (RR: *Release Rate*) aluden asimismo a cambios de volumen por unidad de tiempo. Ambos deben tomar valores negativos. El sostenido se prolonga hasta que se completa la duración establecida por la orden SOUND. Ello significa que si el tiempo de subida y el tiempo de apagamiento sumados son mayores o iguales que el tiempo de duración establecido, no habrá fase de sostenido aun siendo programada. El final comienza una

Ondas de luz

Los gráficos de Atari marcaron una senda que han seguido otros fabricantes

Los ordenadores personales Atari 400 y 800 son muy conocidos por sus sistemas de cartucho enchufable, pero las máquinas en sí mismas también poseen unas configuraciones para gráficos muy refinadas y disponibles en BASIC. Estas configuraciones, que son comunes a ambas máquinas, admiten nueve niveles de visualización en pantalla: tres modalidades para texto (ofreciendo diferentes tamaños de caracteres) y seis modalidades para gráficos. La resolución máxima que se puede obtener es de 320 × 192 puntos.

En los ordenadores Atari se puede escoger entre 16 colores, pero el número máximo que se puede visualizar simultáneamente es de cinco. Existen los juegos de caracteres ASCII estándar en mayúscula y minúscula, así como 37 caracteres para gráficos especiales de Atari. Estos caracteres se pueden utilizar en sentencias PRINT para construir visualizaciones y tablas en baja resolución. Los Atari también permiten controlar el movimiento del cursor desde un programa en BASIC. Esto se consigue utilizando los caracteres de control del cursor dentro de sentencias PRINT, para posicionar el texto que sigue en la pantalla a continuación. Los caracteres de control del cursor permiten mover el cursor arriba-abajo o atrás-adelante.

Una de las configuraciones más atrayentes de los Atari es su capacidad para emplear gráficos al estilo sprite, que se denominan gráficos PM (*Player-Missile*), que permiten que el usuario escriba juegos recreativos de acción rápida en BASIC. Sin embargo, no existen órdenes especiales en este lenguaje para utilizar los gráficos PM, y todo el trabajo necesario se ha de efectuar manipulando las posiciones de memoria de la RAM, mediante PEEK y POKE. Los gráficos PM serán analizados con mayor detalle en un próximo capítulo.

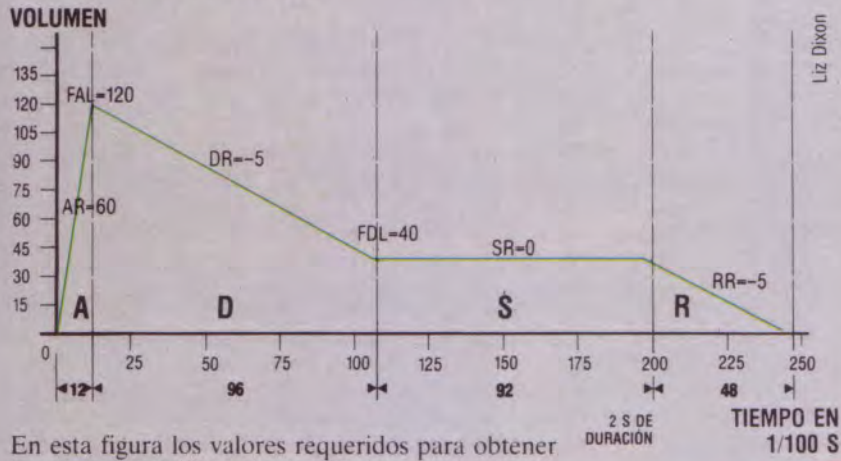
Modalidades de visualización

Las modalidades 0, 1 y 2 son para visualización de textos. Cuando se enciende la máquina, la visualización se establece en la modalidad 0 y a la pantalla se le da un formato de 24 filas, cada una de las cuales contiene 40 espacios de caracteres. En esta modalidad los caracteres de visualización se basan en el formato estándar ASCII de ocho por ocho. Los caracteres que se imprimen (PRINT) en la modalidad 1 son dos veces más anchos que los caracteres de la modalidad 0, pero conservan la misma altura; por su parte, los caracteres de la modalidad 2 son dos veces más altos y anchos que los de la modalidad 0.

A excepción de la modalidad 0, todas las modali-

vez completa la duración. El volumen baja hasta cero a la velocidad establecida, a menos que por el mismo oscilador no se dé comienzo a una nueva nota, lo que significa que el final se descarta, fenómeno evitable estableciendo "H" en "1" mediante una nueva orden SOUND &.

Envoltura de volumen



En esta figura los valores requeridos para obtener la envoltura similar a la de un piano serían los siguientes:

T = 6 AR = 60 SR = 0 FAL = 120
DR = −5 RR = −5 FDL = 40

duración SOUND = 40 (dos segundos)

Que expresaríamos así:

ENVELOPE 1,6,0,0,0,0,0,0,60,−5,0,−5,120 40

El siguiente programa emplea todas las órdenes de sonido en BASIC BBC para ejecutar una frase musical muy conocida con la envoltura de volumen propia del piano, y una breve envoltura de tono triangular repetida en el acorde final.

```
10 REM **COSMICO**
20 ENVELOPE 1,6,0,0,0,0,0,0,60,−5,0,−5,120,40
30 ENVELOPE 2,6,1,−1,1,1,2,1,60,−5,0,
   −5,120,40
40 FOR I = 1TO4: READ N
50 SOUND 1,1,N,20:REM **TOCAR LA SI SOL
   SOL**
60 SOUND &1001,0,0,5:NEXT I
70 SOUND &201,2,77,40: REM **ULTIMO**
80 SOUND &202,2,89,40: REM **RE MAYOR**
90 SOUND &203,2,109,40: REM **ACORDE**
100 DATA 137,145,129,85: REM **LA SI SOL
    SOL**
```

dades para gráficos poseen una pantalla dividida, reservándose las pocas líneas inferiores para datos diversos, como los mensajes de error. Para imprimir en el cuerpo principal de la pantalla en las modalidades 1 y 2, se ha de especificar un número de dispositivo. PRINT #6 permite imprimir texto en la

Mod.	Tipo	Filas	Cols.	Colores
0	texto	24	40	2
1	texto	20	20	5
2	texto	10	20	5
3	gráficos	20	40	4
4	gráficos	40	80	2
5	gráficos	40	80	4
6	gráficos	80	160	2
7	gráficos	80	160	4
8	gráficos	160	320	1

parte de gráficos de la pantalla. Las modalidades de la 3 a la 8 son para gráficos y permiten trazar en la pantalla puntos y líneas con grados variables de resolución y una selección de colores. La tabla que ofrecemos refleja la gama completa de las opciones que el usuario tiene a su disposición.

Órdenes en BASIC

El BASIC de Atari dispone de cierto número de órdenes para ayudar con los gráficos. Estas órdenes también funcionan, en forma modificada, en las tres modalidades para texto.

SETCOLOR a,b,c

Existen cinco registros de color para controlar su utilización en la pantalla, pero no todos ellos se emplean en todas las modalidades. SETCOLOR sirve para seleccionar los colores utilizados por estos cinco registros. En esta orden, a es el número del registro de color, 0-4; b es el número del color a utilizar, 0-15; y c permite que cada color se visualice en uno de los ocho niveles de brillo, escogiendo un número par entre 0 y 14.

COLOR n

Esta orden funciona de dos maneras, según se haya seleccionado una modalidad para texto o para gráficos. En las modalides 0, 1, y 2, n es un número de la escala entre 0 y 255. En su forma binaria, este número consta de ocho bits: los seis primeros aluden al código ASCII del carácter que se está trazando (PLOT), y los otros dos están reservados para la información de color relativa al carácter.

En las modalidades para gráficos, n asume un valor entre 0 y 3, y se utiliza para seleccionar un determinado registro de control de color cuando se traza (PLOT) un punto.

PLOT x,y

El origen de la pantalla Atari está situado en el rincón superior izquierdo de la pantalla. PLOT ilumina el punto de gráficos con las coordenadas (x,y). Del mismo modo, la orden POSITION:

POSITION x,y

coloca un cursor invisible en el punto (x,y) de la pantalla.

DRAWTO x,y

dibuja una línea recta (o lo más recta posible en las modalidades de resolución más baja) desde la antigua posición del cursor hasta (x,y). La línea:

X10 18, #6,0,0,"S:"

emplea la orden de Atari X10 para entrada-salida, que le permite al usuario rellenar o pintar una forma dibujada en la pantalla. Es bastante complicada, pero si se la emplea con cuidado puede dar buenos resultados. Una vez que se ha dibujado en la pantalla un área cerrada, se debe establecer el cursor en el rincón inferior izquierdo de la zona a colorear. La coloración comenzará desde la parte superior del área e irá rellenándola, entre los límites, hasta que la posición del cursor alcance la parte inferior. El color se establece mediante POKE 765,C, donde C es 1, 2 o 3, como en la orden COLOR.

Tamaño XL

Los gráficos de Atari son muy interesantes, pero no fáciles de utilizar, aunque tienen la ventaja de su amplia gama de modalidades para texto. El siguiente programa le demuestra al usuario la utilización de los caracteres de tamaño doble, junto con la orden POSITION, para imprimir en la pantalla un mensaje familiar:

```
10 REM *LETRAS
   GRANDES*
20 GRAPHICS 2+16
30 SETCOLOR 0,3,6
40 FOR X = 19T08
   STEP−1
50 POSITION X,1
60 FOR J = 1 TO 100:
   NEXT J
70 PRINT #6; "CURSO "
80 NEXT X
90 FOR X = 19T06
   STEP−1
100 POSITION X,3
110 FOR J = 1 TO 100:
    NEXT J
120 PRINT #6; "MI "
130 NEXT X
140 FOR X = 13T07
    STEP−1
150 POSITION X,9
160 FOR J = 1 TO 100:
    NEXT J
170 PRINT #6;
    "COMPUTER "
180 NEXT X
190 SETCOLOR 0,5,5
200 FOR Y = 9T05
    STEP−1
210 POSITION 7,Y
220 PRINT #6;
    "COMPUTER "
230 NEXT Y
240 GOTO 240
```

Observe que cuando se selecciona una modalidad, se puede contrarrestar el efecto de la pantalla dividida agregando 16 al número de modalidad

Jerga

Algunos de los términos que se utilizan en el mundo de la informática nacieron del modo más pintoresco

Muchas de las palabras que definen aspectos de la informática tienen un extraño origen. Toda profesión cuenta con un lenguaje especializado propio (palabras o frases que utilizan las personas que desempeñan esa actividad), pero ninguna ha creado tantos términos como la industria informática. Incluso se ha acuñado un vocablo para referirse a ellos: *buzzwords*.

Buzzwords

La palabra **BUZZWORD** se utilizó por primera vez a finales de los años sesenta, cuando a alguien del departamento de publicidad de la Honeywell le dio por crear un juego denominado *generador de "buzzwords"*. El juego se basa en tres columnas de diez palabras cada una, numeradas del 0 al 9. La primera columna es una lista de sustantivos, las dos restantes contienen adjetivos o modificadores indirectos que pueden ir gramaticalmente unidos a los primeros. Piense usted un número de tres cifras, busque las palabras correspondientes y se encontrará con una frase tan carente de significado como: "matriz heurística de diagnóstico". Si la dice en una conversación, imagine el desconcierto y hasta el asombro que causará.

Boot

BOOT es un apócope de *bootstrap*, como este vocablo lo es de *"to pull oneself up by one's bootstraps"* (pararse uno por sus propios pies). Un cargador de bootstrap es una rutina que se ejecuta automáticamente cada vez que a un ordenador se le proporciona alimentación eléctrica (el usuario de ordenadores especializado no se conforma con decir "encendido"). En aquellas máquinas que no poseen un sistema operativo en ROM, la rutina boot debe contener instrucciones para llamar a ese sistema operativo desde el disco.

Bit

Aunque la mayoría de los diccionarios afirman que se trata de una contracción de **BI**nary Digi**T** (dígito binario), parece igualmente probable que no sea más que una ampliación de su significado en inglés: brizna. Sin embargo, vale la pena tener presente que en el *argot* norteamericano un bit es asimismo la octava parte de un dólar y siempre se emplea de a pares: "dos bits", por ejemplo, equivalen a un cuarto (25 centavos).

Con frecuencia *bit* aparece como prefijo: como en *bit-slicing*, término que se utiliza para indicar que ciertos microprocesadores bastante sofisticados se pueden construir a partir de "bloques de construcción" de dos, cuatro u ocho bits, lo que da lugar a dispositivos de hasta 32 bits de capacidad. Igualmente, el sentido común informático nos dice que los programas que permanecen durante mucho tiempo sin utilizar originan errores adicionales e insolubles: este previsible fenómeno se conoce como *bit-decay*.

Turnkey

Para cuando una persona recibe un equipo informático, quizá por primera vez, la jerga ha reservado una nueva palabra. Muchas organizaciones comerciales trabajan con una firma de consultores de informática para instalar el hardware y el software, de modo que el cliente lo pueda recibir listo para funcionar. Esto se conoce como operación **TURNKEY,** porque todo lo que el cliente ha de hacer es sólo dar vuelta a la llave y empezar a servirse de la máquina.

Basic

BASIC significa **B**eginners' **A**ll-purpose **S**ymbolic **I**nstruction **C**ode (código de instrucciones simbólico para principiantes). Al igual que sucede con muchos acrónimos, el término está tan extendido que uno llega a creer que primero vino la palabra y después la frase.

Hardware

Software

HARDWARE y **SOFTWARE** son palabras bien conocidas (*hard* significa tangible y *soft*, lo contrario), pero existen asimismo otros dos tipos de *ware*: **FIRMWARE**, que significa que el software está encapsulado en el hardware (tal como sucede en la ROM o en la EPROM), y **LIVEWARE**, vocablo genérico ¡que engloba a todas aquellas personas que tienen la suerte de trabajar con ordenadores y de usarlos!

Baudio

La palabra **BAUDIO** (la velocidad a la cual se transmiten los datos) se escogió en honor de Emile Baudot, inventor de un código telegráfico que llegó a competir con el desarrollado por Samuel Morse.

Byte

BYTE es un vocablo informático que aparece con mucha frecuencia y, aunque sólo tiene 30 años, sus orígenes ya se han perdido entre las tinieblas. Hasta que apareció el microprocesador de ocho bits, los bits de un byte eran suficientes para codificar un único carácter, en algunas ocasiones seis, otras ocho. En aquel entonces, era muy raro que los ordenadores utilizaran una palabra de menos de 24 bits; y las máquinas diseñadas para aplicaciones específicas, llegaban hasta 64 bits. La evocación semántica de byte (mordisco) ha llevado a la acuñación del término **NYBBLE** (bocadito): ¡medio byte! Abusando todavía más de la analogía, un **GULP** (sorbo) es un pequeño grupo de bytes.

Basura

BASURA es una palabra que está presente en diversas frases del lenguaje de los usuarios de ordenadores. Por ejemplo, el acrónimo **GIGO** significa **G**arbage **I**n, **G**arbage **O**ut (basura entra, basura sale), y éste es en realidad un recordatorio de que los ordenadores sólo procesan información y que, por lo tanto, uno no puede esperar resultados exactos si en primer lugar no se provee a la máquina de datos pertinentes.

ACUMULACIÓN DE INFORMACIÓN INSERVIBLE es la expresión con la que se designa el proceso interno que se podría muy bien utilizar en su ordenador personal si éste empleara una versión de BASIC que admitiera series dinámicas (es decir, series cuya longitud puede alterarse durante un programa). Cada vez que una serie aumenta en longitud, en la RAM se hace una nueva copia completa. De modo que si hubiera varias sentencias de la forma `LET A$ = A$+"*"` (en especial dentro de bucles), entonces al cabo de poco tiempo la memoria se llenaría por completo. En este punto, la ejecución del programa se interrumpirá temporalmente de forma automática y una rutina de la ROM denominada *recolector de información inservible* ordenará la zona de la serie y eliminará todas las secciones de series que hubieran quedado de un tratamiento anterior. Aunque el programa continuará cuando el recolector haya terminado su tarea, el proceso puede tardar segundos e incluso minutos, durante los cuales el ordenador no operará.

Bombas lógicas

Caballos de Troya

Los medios de comunicación se han apresurado siempre a hacer suyos los imaginativos términos de la jerga y en los últimos años se han dedicado a utilizar algunos de propio cuño. El tema de los delitos informáticos es un terreno especialmente fértil para el nacimiento de estos vocablos: **BOMBAS LÓGICAS** y **CABALLOS DE TROYA** son dos de los métodos supuestamente utilizados con fines fraudulentos. El primero da idea de unas instrucciones que se escriben en un programa de aplicaciones pero que permanecen inactivas (o sea, sin efecto alguno) hasta que el programa se ha ejecutado durante un tiempo suficiente para que el fraude (transferir dinero de una cuenta a otra, tal vez) no se pueda detectar. El término caballo de Troya alude a un programa disfrazado de otro para introducirse en el sistema.

Apretón de manos

Un acuerdo comercial suele sellarse con un apretón de manos entre las partes interesadas; por analogía, en el lenguaje informático, un **APRETÓN DE MANOS** es el nombre que se le da a la señal electrónica que avisa que se ha completado un intercambio de datos.

Bomba de relojería

Una expresión parecida, pero que se refiere a una práctica legítima, es **BOMBA DE RELOJERÍA**. Alude a una técnica particularmente ingeniosa para proteger al software de gestión contra la piratería. Se trata de una codificación introducida dentro del propio paquete, la cual queda inhabilitada cuando el sistema lo instala un comerciante honrado. Sin embargo, en una copia pirata, la bomba entrará en funcionamiento al llegar una fecha determinada, probablemente en el momento en que la compañía dependa en gran medida del paquete. Al día siguiente de la "explosión" de la bomba, no sólo se habrán convertido en "basura" los archivos del usuario, sino que también se habrá destruido la copia del programa (a menos que el disco estuviera protegido contra una reescritura).

0. Red	0. Integrada	0. De datos
1. Capacidad	1. Situacional	1. Interactiva
2. Base	2. Vertical	2. De sistemas
3. Expresión	3. Digitalizada	3. De diagnóstico
4. Palanca	4. Estocástica	4. Direccional
5. Matriz	5. Lineal	5. Aleatoria
6. Impresora	6. Heurística	6. Gráfica
7. Aplicación	7. Relativa	7. Alfanumérica
8. Jerarquía	8. Habitual	8. Esquemática
9. Imagen	9. Programable	9. Modular

Generador de "términos"
El término *buzzword* se utilizó por primera vez para describir un juego sencillo consistente en crear frases en jerga tecnológica sin ningún significado, pero muy "convincentes". Usted puede desarrollar su propio "generador de *buzzwords*" ideando tres columnas de diez palabras cada una, como nosotros hemos hecho aquí. La elección de un número al azar de tres dígitos "generará" un término muy erudito (a la violeta)

Commodore PET 4032

El Commodore PET fue el primer ordenador personal. Desde su introducción en el mercado, su hardware ha mejorado sensiblemente

En muchos sentidos, el Commodore PET (acrónimo de *Personal Electronic Transactor*) fue la máquina que inició el *boom* del microordenador. Cuando se lanzó al mercado, en 1977, estableció un estándar tan elevado que, en comparación, algunas máquinas más recientes casi pueden considerarse retrocesos. La carcasa metálica de la máquina original es buena muestra de su superioridad. Aparte de Memotech y de las máquinas para gestión más caras, las carcasas de la mayoría de los ordenadores recientes están moldeadas en plástico y van desde las aún aceptables hasta las decididamente malas. La fuente de alimentación eléctrica incorporada del PET es otro detalle que lo distingue claramente de muchos de sus competidores del mercado personal.

Chris Stevens

El teclado y el monitor del PET
El teclado de los primeros PET no era estándar; el de los más recientes se aproxima en mayor medida al estilo de los de las máquinas de escribir e incorpora los símbolos para gráficos en el frente de las teclas (exceptuando los modelos de gestión). Todos los PET poseen monitores incorporados: los últimos tienen pantalla de 12" (30 cm), con visualizaciones en verde sobre negro y una opción de columnas de 40 u 80 caracteres

Si bien al menos dos años antes de que se lanzara el PET al mercado ya existían máquinas de 8 bits e incluso de 16, éstas eran *kits* para montar o simples "sistemas mínimos" compuestos sólo por chips sobre un circuito impreso. El PET fue el primer microordenador en salir a la venta del que realmente se podía afirmar que para usarlo sólo había que enchufarlo. Las primeras versiones del PET tenían una grabadora en cinta incorporada, con control de motor, un monitor incorporado y BASIC en la ROM. Todo cuanto debía hacer un nuevo usuario para comenzar a trabajar con él era enchufarlo y encenderlo, y aparecía un mensaje tranquilizador:

```
COMMODORE BASIC VER. 1.0
7167 BYTES FREE
READY
```

Chip sincronizador
Cuando se enciende un ordenador, los circuitos tardan unos instantes en estabilizarse. Este sincronizador espera durante una fracción de segundo, al cabo de lo cual repone el microprocesador para que comience el intérprete de BASIC

Conexión dispositivos
Esta interface contiene un número de líneas útiles, incluyendo una puerta en paralelo de ocho bits y conexiones para interface de monitor externo. Es particularmente adecuada para conectar proyectos electrónicos de diseño casero

Puerta IEEE488
El PET fue el único de los primeros microordenadores que incluyó esta interface en paralelo. Debido a que podía direccionar hasta 15 periféricos, se utilizaba para activar tanto discos como impresoras. La IEEE488 también es la puerta estándar que se emplea para conectar en interface equipos científicos de laboratorio

6522
Este adaptador versátil para interfaces es parecido al 6520, pero contiene un registro para realizar desplazamientos para convertir datos de en paralelo a en serie y viceversa, así como dos sincronizadores programables que se pueden utilizar para controlar equipos externos

El usuario podía entonces empezar a digitar y su trabajo podía ser almacenado en una cassette a toda prueba, sin necesidad de tener que enchufar diversos componentes entre sí ni de cargar de una cinta programas para el sistema de carga (o, peor aún, de tener que darle entrada mediante un teclado HEX, que era algo común en aquellos días).

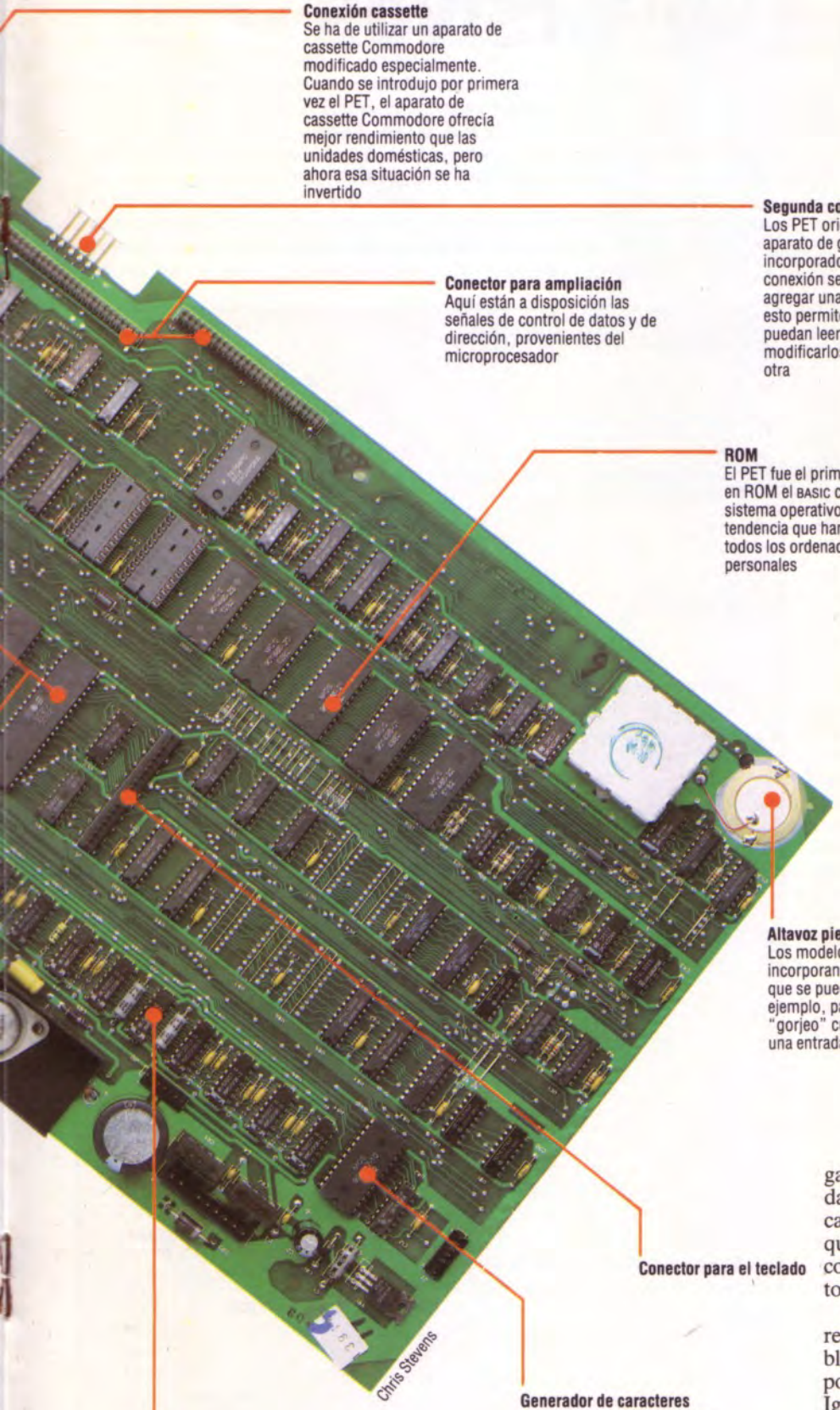
A lo largo de su existencia, el BASIC de Commodore ha sido objeto de varias revisiones, y la última versión (4) se ha ampliado tanto como para convertirse en una nueva.

Otra configuración importante y exclusiva del PET es el juego de caracteres. Consta tanto del juego ASCII completo como de una gran variedad de gráficos de bloques; los usuarios de PET los han empleado de manera notablemente creativa, a pesar de la algo baja resolución de los caracteres. No obstante, un problema importante de la máquina era que los códigos generados por el teclado no se correspondían con el juego ASCII ni estaban acomodados a ningún orden estandarizado.

La intensa utilización de estos gráficos de bloques se ha reforzado merced a la existencia de una

6520
Estos PIA (*Peripheral Interface Adaptors*: adaptadores de interfaces de periféricos) se encargan de la mayoría de las interfaces, incluyendo cassette y teclado

6502
El PET lo diseñó Chuck Peddle, de Commodore, de manera que no sorprende que se base en un microprocesador 6502, también diseñado por él. Aunque los ordenadores de gestión han optado por otros procesadores, el 6502 sigue gozando de popularidad entre los usuarios de ordenadores personales



Conexión cassette
Se ha de utilizar un aparato de cassette Commodore modificado especialmente. Cuando se introdujo por primera vez el PET, el aparato de cassette Commodore ofrecía mejor rendimiento que las unidades domésticas, pero ahora esa situación se ha invertido

Segunda conexión cassette
Los PET originales tenían un aparato de grabación incorporado. Ahora esta conexión se puede utilizar para agregar una segunda unidad y esto permite que los datos se puedan leer en una cinta, modificarlos, y luego escribir en otra

Conector para ampliación
Aquí están a disposición las señales de control de datos y de dirección, provenientes del microprocesador

ROM
El PET fue el primero en colocar en ROM el BASIC completo y el sistema operativo, iniciando una tendencia que han seguido casi todos los ordenadores personales

Altavoz piezoeléctrico
Los modelos posteriores fueron incorporando este dispositivo que se puede programar, por ejemplo, para producir un "gorjeo" cuando el usuario hace una entrada errónea

Conector para el teclado

Generador de caracteres
Además de los 64 caracteres alfanuméricos, el PET puede generar 64 símbolos para gráficos. El texto se puede visualizar en mayúsculas y minúsculas, a voluntad

RAM
Los PET poseen de 8 a 32 Kbytes como estándar. Mediante una modificación especial, esta cifra se puede ampliar a 96 Kbytes

COMMODORE PET

DIMENSIONES
480 x 440 x 300 mm

CPU
6502

VELOCIDAD DEL RELOJ
1 MHz

MEMORIA
32 Kbytes de RAM
20 Kbytes de ROM

VISUALIZACION EN VIDEO
25 líneas de 40 caracteres. Monitor de fósforo verde de 12" (30 cm) incorporado. 256 caracteres y símbolos para gráficos visualizables, o gráficos en baja resolución (50 x 80)

INTERFACES
IEEE488, puerta en paralelo de 8 bits para el usuario, cassette (2)

LENGUAJE SUMINISTRADO
BASIC, monitor de lenguaje máquina

OTROS LENGUAJES DISPONIBLES
PASCAL, COMAL, LISP

VIENE CON
Manual de instrucciones

TECLADO
Teclado estilo máquina de escribir; posee 64 teclas individuales con símbolos para gráficos inscritos en el frente. Un relleno de teclado numérico separado incluye teclas de función de calculadora

DOCUMENTACION
Commodore nunca ha gozado de mucho prestigio por la calidad de su documentación, aunque ésta ha mejorado mucho desde los primeros días

gama de impresoras que los reproducen sin necesidad de la complicada programación de bits de la cabeza de impresión. Por supuesto, esto significa que el número de impresoras aptas para emplearlos con el PET es limitado y la mayoría, aunque no todas, son de la propia casa Commodore.

Como consecuencia de estas características diferenciadoras, y a pesar de que existe una considerable cantidad de software para la máquina, es muy poco el que se ha traducido para otras máquinas. Igualmente, son pocos los programas de otras máquinas que se han convertido al BASIC del PET, porque la conversión por lo general implica un esfuerzo excesivo y resulta más sencillo simplemente volver a escribirlos. Por consiguiente, en cierto sentido la máquina se ha quedado un poco "aislada" en su pequeño mundo propio y las modificaciones que se producen en la industria en general apenas si inciden en ella.

Elementos subversivos

Con una cuidadosa planificación y un enfoque paso a paso se acorta el tiempo necesario para eliminar los errores de un programa

A medida que usted vaya adquiriendo mayor experiencia en la escritura de programas, también irá empeñándose cada vez más en "depurarlos". Los errores sintácticos y los errores de lógica, en los que incurren hasta los programadores de ordenadores más experimentados, se van haciendo cada vez menos frecuentes y menos problemáticos a medida que su experiencia aumenta. Vamos a dar algunas sugerencias para ayudarle a evitar los errores de programación y para aumentar su eficacia en la depuración del código.

Comencemos por donde un programa empieza: ¡por el cerebro de usted! Si desde el principio elabora equivocadamente el concepto de un programa, entonces lo más seguro es que al escribirlo quede plagado de errores.

Una idea mucho mejor es la de comenzar a escribir un programa enunciando primero el problema, a uno mismo o a otras personas, con la mayor claridad posible. Después, divida el problema en partes completas desde el punto de vista de la lógica (entrada, salida, algoritmos, estructuras de datos, procesos, etc.) y considere cada una de estas partes como un problema separado. De ser necesario, divida cada uno de estos problemas en subproblemas, y así sucesivamente, hasta que el problema original sea un conjunto estructurado de pequeños problemas, cada uno de los cuales le resultará sencillo de programar. Un enfoque formal, como utilizar un seudolenguaje o un diagrama de flujo, es esencial en la etapa de diseño para registrar (y preservar) la estructura del programa como un todo. Debe tratar de permanecer alejado del teclado hasta que honestamente considere que está en condiciones de saber cómo programar cada una de las partes del problema. Éste es el enfoque *top-down* de la programación, método que reduce notablemente el tiempo que se consume en la depuración.

Dividir los problemas en tareas resolubles le llevará a escribir programas que realmente sean conjuntos de subrutinas o procedimientos unidos por un programa principal esquematizado. Así resulta más fácil hallar los errores y usted puede construir una biblioteca de subrutinas a prueba de errores para utilizarlas en programas ulteriores. De otro modo, estaría continuamente inventando el fuego: cada vez que escribiera, por ejemplo, un programa para clasificar datos, volvería a resolver el problema de escribir una rutina de clasificación.

En la medida en que el BASIC se lo permita, trate en todo caso de utilizar nombres adecuados para las variables, aun cuando tenga que abreviarlos. NETO = BRUTO−IMPUESTO, por ejemplo, se explica por sí solo, y NT = BR−IM no es un mal sustituto; pero N = B−I es sumamente ambiguo y no le sugiere indicación alguna acerca de cuáles son las variables en cuestión. Es un buen hábito llevar una tabla de variables, que le indique todas las variables utilizadas en el programa y para qué sirven. Esto puede conducirle a normalizar la utilización de las variables (o sea, denominar ciertas variables siempre con la misma letra, p. ej., las contadoras de bucles) e impide que usted emplee la misma variable con distintas finalidades. Del mismo modo, es una buena costumbre almacenar los valores cons-

Controle los errores

```
100 GOTO 200:X$="THAT'S ALL FOLKS"
120 I=12:K=1984
140 FOR K=1 TO LT
160 PRINT"WHO NEEDS STRUCTURE ?:N$:NEXT
180 RESTORE
190 FOR L=1 TO I
200 INPUT"ENTER YOUR NAME";N$
220 INPUT"ENTER YOUR AGE";LT
240 GOSUB 100
260 PRINT IF YOU'RE";LT;"NOW"
280 PRINT"YOU WERE BORN IN";K-LT
300 YR$=K-LT
320 LY=INT(YR)/4*4
340 IF LY=YR THEN IF INT(LY/100)*100=LY THEN GOTO 370
380 PRINT YR WAS A LEAP YEAR":GOTO 420
390 PRINT "YR WAS NOT A LEAP YEAR"
400 NEXT
420 PRINT X$
440 STEP
```

- Esta sentencia no se ejecutará nunca, porque la orden GOTO hace que se la salte
- El orden de estas dos líneas es incorrecto. La línea 100 debería decir: GOTO 190
- Acabamos de dar a K un número, 1984, y esta sentencia se lo arrebata
- Falta cerrar las comillas; por este motivo el NEXT no se ejecutará
- Debería decir: RETURN
- Error de sintaxis: los dos puntos (:) deberían ser punto y coma (;)
- Esto acarreará un gran problema. Probablemente debería decir: GOSUB 140
- Faltan comillas
- Dará un número sin sentido, porque desde la línea 120 el valor de K no es el deseado 1984
- Error de sintaxis: debe decir YR = K−LT
- El paréntesis está mal colocado: hará un cálculo erróneo. Debería ser: INT(YR/4)*4
- ¡No hay línea 370!
- No es (') sino (;"). Además GOTO 420 implica salir del bucle FOR...NEXT
- Falta el nombre de la variable del bucle: es decir, NEXT L
- X$ no se ha inicializado, de manera que esta sentencia no hará nada
- Error de sintaxis: debería decir STOP

tantes en variables al principio del programa y remitirse a partir de entonces a estas variables. Esto hace que el programa sea más rápido y pulido, y el usuario puede cambiar estos valores sin tener que buscar por el programa cada vez que se producen.

Ni siquiera con este enfoque formal es fácil eliminar los errores. De modo que es muy importante adoptar un método disciplinario para hallarlos y erradicarlos. Los errores más comunes son los de tipo sintáctico, y por lo general el usuario los puede corregir según los vaya encontrando. Pero no siempre es éste el caso. Consideremos lo siguiente:

```
10 PRINT "**PULSE LA BARRA ESPACIADORA**"
20 PRINT "**CUANDO ESTE LISTO**"
```

Las líneas de esta clase suelen producir un mensaje de error cuando se ejecutan si no se las digita como dos líneas separadas. La línea 10 contiene 40 caracteres, de modo que al digitarla en una pantalla de 40 columnas, el cursor acaba donde empieza la segunda línea de la pantalla, con lo que fácilmente usted puede que se olvide de pulsar RETURN en la línea 10 antes de comenzar a digitar la línea 20. De ser así, entonces lo que en su programa parecerían ser dos líneas perfectas en realidad sería una sola línea con un error de sintaxis (el número 20) en la mitad. Una forma de captar estos errores consiste en listar todas las líneas sospechosas individualmente en vez de hacerlo como si fueran una parte del fragmento de un programa.

Los mensajes de error, cuando no son del todo incomprensibles, pueden inducir a confusión. Tomemos el ejemplo:

```
25 DATA 10.2,34,56.9,0.008,15.6
30 FOR K = 1 TO 5: READ N(K): NEXT K
```

La ejecución de estas líneas, al fallar, nos puede hacer pensar que hay un error de sintaxis en la línea 30, cuando en realidad el error está en la línea 25 (uno de los ceros se ha digitado equivocadamente con la letra O).

Los errores de codificación que no son de sintaxis son los más comunes, y por lo general son también los más difíciles de hallar. En este caso, resulta vital ser metódico. Empiece por tratar de descubrir aproximadamente en qué lugar del programa está el error. Esto es bastante fácil en los programas modulares bien estructurados y puede resultar aún más fácil mediante la orden TRACE, que hace que se imprima en la pantalla el número de línea en curso del programa mientras éste se ejecuta. Si su máquina no admite TRACE, entonces puede crear sentencias TRACE periódicamente a lo largo del programa (tal como PRINT "LINEA 150" al comienzo de la línea 150, p. ej.). Del mismo modo, puede usar la orden STOP para interrumpir la ejecución del programa en los lugares significativos del mismo, de manera que pueda examinar los valores de las variables cruciales. Puede hacerlo en la modalidad directa utilizando PRINT, o bien puede escribir una subrutina al final de su programa:

```
11000 REM IMPRIMIR LAS VARIABLES
11100 PRINT "MARCADOR,TAMAÑO,BANDERAS"
11200 PRINT MR;TM;B1;B2
11300 PRINT "MATRIZ TABLERO"
11400 FOR K = 1 TO 10: PRINT TB $(K): NEXT K
```

En consecuencia, cuando el programa llegue a una orden STOP usted puede digitar GOTO 11000 y obtener la visualización del estado actual de las variables. Incluso puede modificarlas (digitando, pongamos por caso, TM = 17 y pulsando RETURN), y restaurar el programa con la orden CONTinuar.

El primer "bug"

Para los programadores novatos los errores suelen asumir rasgos zoomórficos: se esconden del programador y burlan a placer todo esfuerzo por dar con ellos. En inglés se utiliza la palabra "bug" (bicho) para indicar que existe un error. El primer *bug* (al menos aquel del cual proviene el término) era realmente animado. Mientras intentaba eliminar el error de un programa que estaba desarrollando en 1945 para el Harvard Mrk II, Grace Hopper descubrió que una enorme polilla atrapada en el sistema electromecánico del ordenador era la causante del supuesto error

Tony Lodge

Una vez que haya descubierto que el error está escondido en ciertas líneas, o en una variable determinada, en ese momento ya estará cerca de poder eliminarlo, ¡pero vaya con cuidado! Pruebe los remedios uno a uno, para que pueda observar cuál es su efecto exacto en la ejecución. Es muy sencillo introducir diversas modificaciones entre una y otra ejecución, con la idea de librarse de un error y crear con ellas varios más, intentar corregir éstos ¡y, finalmente, olvidarse de cómo empezó todo!

Los bucles y las bifurcaciones, sobre todo cuando están "anidados", pueden ser terreno particularmente propicio para cometer errores y exigen especial atención tanto al escribirlos como al depurarlos. Consideremos este trozo de código:

```
460 IF SM < 0 AND SC <> -1 THEN IF SC > 0 OR
    SM = SC-F9 THEN LT = 500
470 FOR C1 = 1 TO LT:FOR C2 = LT TO C1
    STEP -1
480 SC = SM + SC*C2
490 NEXT C2: SM = 0: NEXT C1
```

¿Y qué significa todo esto? Aun cuando usted supiera lo que se ha de hacer, ¿está seguro de conseguirlo? Colocar sentencias dentro de un bucle cuando éstas deberían estar fuera de él es una forma segura de facilitar la aparición de errores. Y lo mismo puede decirse cuando no se cubren todas las condiciones posibles al escribir sentencias IF... THEN. En este sentido, constituye un caso especial el que se produce cuando se escriben sentencias múltiples después de IF...THEN. Por ejemplo:

```
655 IF A$ = " " THEN GOTO 980:A$ = B$
660 PRINT A$
```

La sentencia A$ = B$ no se ejecutará jamás, porque o bien A$ = " ", en cuyo caso el control pasaría a la línea 980, o A$ <> " ", en cuyo caso se ignoraría el resto de la línea 655.

El mejor maestro para la depuración de errores es la experiencia, pero un enfoque paso a paso y un método disciplinado son ayudas inestimables. Tómeselo con tiempo y ¡NO SE ATEMORICE!

El show del láser

La tecnología del disco óptico (láser) hace accesibles al ordenador personal dos importantes aplicaciones: el video interactivo y el almacenamiento masivo de datos

La capacidad de almacenamiento interno del ordenador es importante, pero la capacidad de su sistema de almacenamiento masivo de datos a largo plazo resultará decisiva. Al cabo de un par de meses, el entusiasta usuario de un ordenador personal habrá acumulado una considerable cantidad de cassettes o varias cajas de discos. Como la mayoría de estos programas no se modificarán nunca, estarían mejor almacenados en cartuchos de ROM que en delicados medios magnéticos. Lo que sería muy útil es alguna forma de sistema de almacenamiento digital que fuera sólo de lectura, como un cartucho, pero que tuviera muchísima más capacidad.

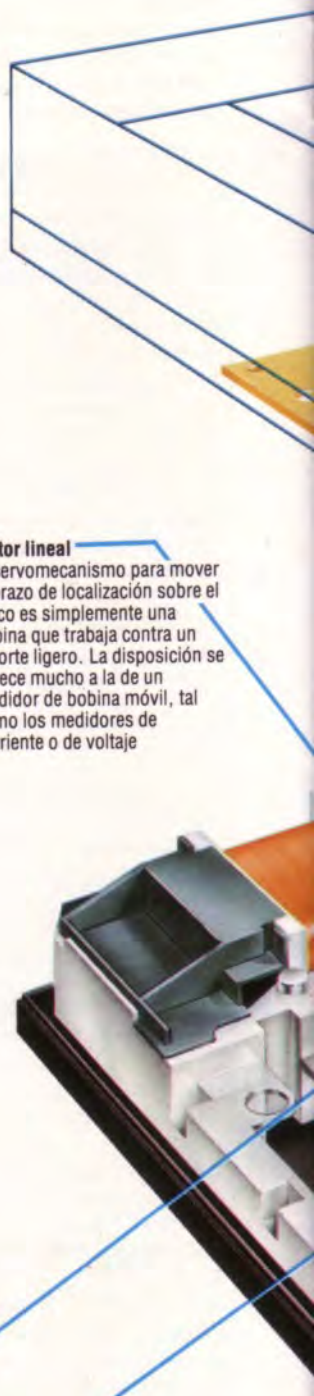
Dicho sistema existe y es el disco láser óptico. Lo corriente, sin embargo, es que este sistema se utilice en el hogar sólo como una alternativa a la grabadora de videocassette, para mostrar el material pregrabado. Otra utilización de la misma tecnología es el disco compacto de audio.

La diferencia entre estos dos tipos de sistemas (aparte de los diámetros de sus discos) estriba en sus métodos de operación. Mientras que un videodisco es un sistema analógico, un disco de audio compacto almacena su información en forma digital, es decir, como una secuencia de unos y ceros. Esta información se vuelve a convertir en la señal de audio original mediante un convertidor digital-analógico, que es el opuesto electrónico del proceso que en primer lugar creó la información. Dado que existen tantos campos eléctricos dispersos en el entorno doméstico, no es práctico utilizar los medios magnéticos como los discos flexibles para la grabación de video. En cualquier caso, la cantidad de información de un disco óptico puede ascender a millones de megabytes, y esto es mucho más de lo que puede retener hasta un disco Winchester.

Existen en el mercado varios sistemas de discos láser ópticos, pero hasta el momento el que ha obtenido mayor éxito es el que ha introducido Philips. Este sistema usa un disco plástico de 14″ (35 cm) que en realidad no es sino una envoltura de protección. La información propiamente dicha está oculta dentro del plástico en forma de una serie de "agujeros" en una lámina metálica. Al igual que en un disco flexible, la información almacenada está catalogada en el videodisco, de manera que, con un tocadiscos adecuado, se puede pasar instantáneamente a cualquier unidad de información individual. Una vez que la cabeza de lectura está en la posición deseada, el rayo láser lee la información del disco. La luz pasa a través del plástico y cae sobre la superficie de la chapa metálica. Una célula fotosensible lee entonces la información a medida que los agujeros de la chapa van reflejando la luz. La información se graba en una única pista en espiral, a fotograma de video por revolución. Esto da un total de 54 000 fotogramas por cada cara del disco, o 36 minutos de ejecución.

Las principales aplicaciones potenciales de los discos ópticos en el campo de los ordenadores van por dos caminos. El primero, que ya existe en el mercado, es el del *video interactivo*. Un programa transmitido por televisión no es interactivo: el espectador no puede controlar el orden en que se le presentan las imágenes. Pero, con el video interactivo, la información visual y textual se almacena en un videodisco que está conectado con un ordenador. El disco se puede entonces utilizar como una biblioteca de referencia, con el texto visualizado superpuesto contra las imágenes de video en la pantalla de un televisor convencional. En respuesta a las indicaciones del ordenador, el usuario puede seleccionar "pistas" o "escenas" específicas del videodisco para que se reproduzcan. Por otro lado, el disco se puede emplear como medio de entrenamiento, visualizando en un televisor las acciones en movimiento o las imágenes fijas y dando entrada en el ordenador a las respuestas de quien se está entrenando a las preguntas importantes, pudiendo el ordenador controlar e informar sobre el rendimiento del usuario. Aún no están muy difundidas las interfaces entre un videodisco doméstico y un ordenador personal. Así y todo, Philips comercializa un modelo profesional de su *Laser Vision*, que puede hacer frente por sí mismo al video interactivo o se puede conectar en interface con un ordenador mediante una puerta IEEE488 o una RS232.

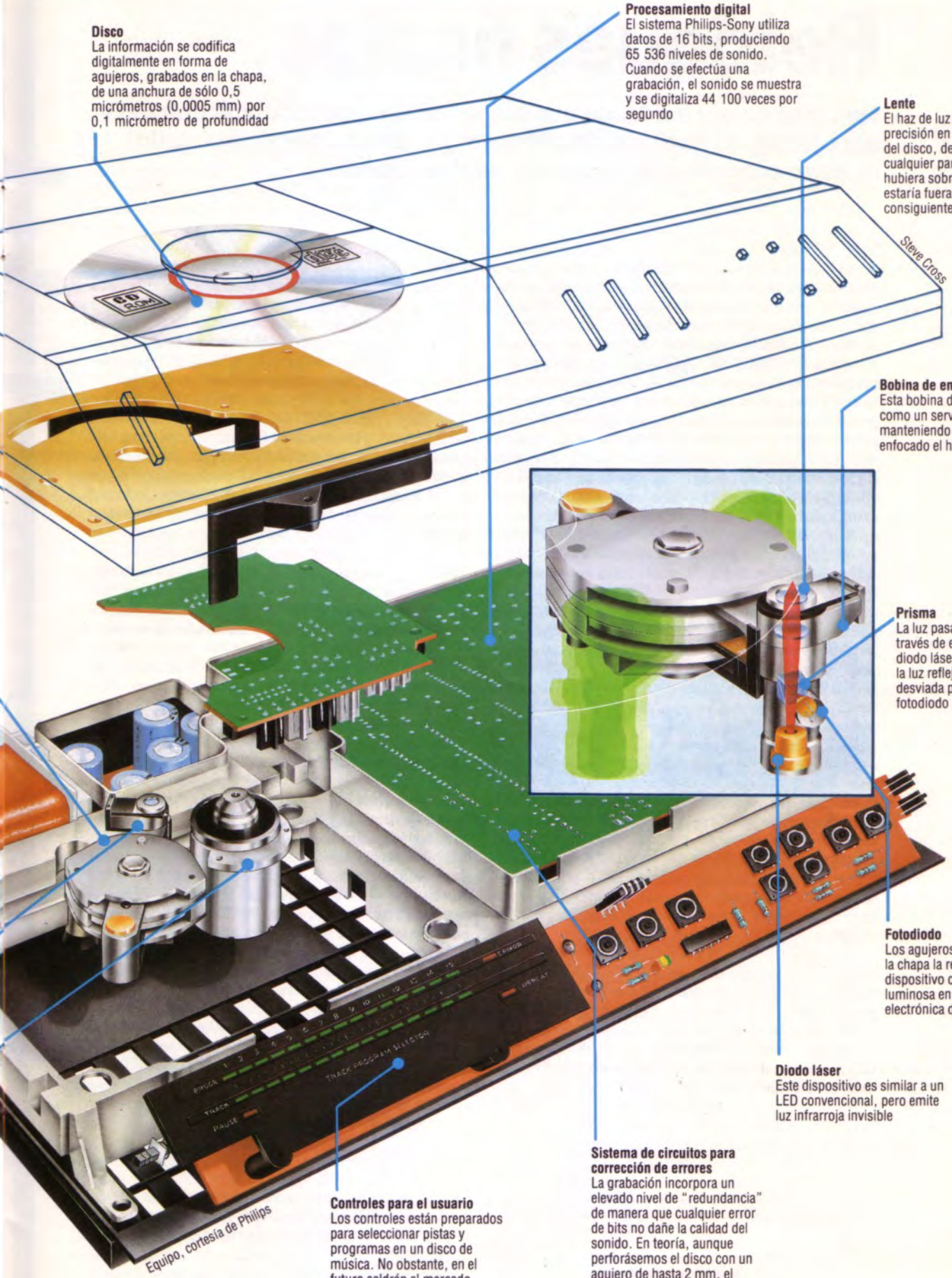
El otro campo en el que es probable que se explote la tecnología del disco óptico es en la provisión de software para ordenadores. Imagínese, por ejemplo, las ventajas de suministrar un ordenador con todo su software de sistemas (tratamiento de textos, base de datos, hoja electrónica y varias docenas de juegos) en un solo disco "incorruptible". Es probable que éste asuma el formato del disco compacto, pero hasta el momento no se ha equipado ningún tocadiscos compacto con una interface para ordenador. Con un mercado potencial tan enorme, es razonable esperar que en un plazo muy breve aparezcan los tocadiscos compactos domésticos con interfaces de este tipo, así como reproductoras exclusivas de discos compactos para ordenadores personales. Sony y Philips ya han hecho pública su intención de producir un tocadiscos exclusivo para ordenadores, denominado CDROM.



Motor lineal
El servomecanismo para mover el brazo de localización sobre el disco es simplemente una bobina que trabaja contra un resorte ligero. La disposición se parece mucho a la de un medidor de bobina móvil, tal como los medidores de corriente o de voltaje

Brazo de localización
El brazo gira centralmente sobre un pivote, está cuidadosamente equilibrado y es de giro libre. Por consiguiente, la cabeza de lectura describe un arco a través del disco

Motor
La velocidad de rotación del disco se controla con gran precisión utilizando un sistema de circuitos de realimentación. A medida que el brazo va del centro del disco al borde, la velocidad cambia de 500 a 200 r.p.m. para mantener constante la densidad de grabación



Disco
La información se codifica digitalmente en forma de agujeros, grabados en la chapa, de una anchura de sólo 0,5 micrómetros (0,0005 mm) por 0,1 micrómetro de profundidad

Procesamiento digital
El sistema Philips-Sony utiliza datos de 16 bits, produciendo 65 536 niveles de sonido. Cuando se efectúa una grabación, el sonido se muestra y se digitaliza 44 100 veces por segundo

Lente
El haz de luz se enfoca con gran precisión en la bobina interior del disco, de modo que cualquier partícula de polvo que hubiera sobre su superficie estaría fuera de foco y, por consiguiente, se ignoraría

Bobina de enfoque
Esta bobina de miniatura actúa como un servomecanismo, manteniendo rígidamente enfocado el haz luminoso

Prisma
La luz pasa directamente a través de este prisma desde el diodo láser hasta la lente, pero la luz reflejada desde el disco es desviada por el prisma hacia el fotodiodo

Fotodiodo
Los agujeros dispersan la luz, y la chapa la refleja. Este dispositivo convierte la señal luminosa en una secuencia electrónica de unos y ceros

Diodo láser
Este dispositivo es similar a un LED convencional, pero emite luz infrarroja invisible

Sistema de circuitos para corrección de errores
La grabación incorpora un elevado nivel de "redundancia" de manera que cualquier error de bits no dañe la calidad del sonido. En teoría, aunque perforásemos el disco con un agujero de hasta 2 mm, el sonido no quedaría afectado

Controles para el usuario
Los controles están preparados para seleccionar pistas y programas en un disco de música. No obstante, en el futuro saldrán al mercado periféricos exclusivos para ordenador utilizando tecnología de disco compacto

Equipo, cortesía de Philips

Retoques finales

Nos falta salvar los fallos que surjan de unir los módulos entre sí y agregar algún que otro detalle para que demos por completado nuestro programa de la agenda de direcciones

En el capítulo anterior de nuestro curso de programación, dejamos a los lectores con el problema de resolver por qué al ejecutar el programa de la agenda de direcciones, agregar luego un registro (utilizando *INCREG*), localizar después un registro (empleando *ENCREG*) y salir a continuación del programa (utilizando *SAPROG*) se tiene como resultado que no se guarde el registro agregado. El problema surgió a causa de la variable RMOD utilizada como una bandera que indica si se ha modificado un registro (lo que significa que el archivo pudiera estar desordenado). La subrutina *CLSREG* clasificaría el archivo en orden alfabético y después establecería RMOD en 0 en la suposición de que el archivo estuviera en orden. Ejecutando *SAPROG* se verificaba para ver que el archivo estuviera en orden (RMOD = 0) y no se molestaba en guardar el archivo si éste estaba en una situación clasificada.

Agregando un registro nuevo (mediante *INCREG*) establece RMOD en 1 (dado que se habría modificado un registro, es decir, se habría agregado un registro nuevo), pero *CLSREG* establecería RMOD en 0, indicando que el archivo se había clasificado. Sin embargo, lo que realmente se necesita, independientemente de que el archivo se haya clasificado o no, es una bandera para señalar la modificación de un registro y otra bandera aparte para indicar si el archivo está o no clasificado. Luego, las subrutinas que necesitan saber si el archivo está clasificado pueden verificar la bandera "clasificado", y las que necesiten saber si se ha modificado algún registro pueden verificar la bandera "modificado".

Los nombres adecuados para las dos banderas serían RMOD, para mostrar si se ha modificado un registro, y CLAR, para mostrar si el archivo se ha clasificado.

Cuando se presentó el programa en la p. 399, la línea 1230 contenía la sentencia LET SVED = 0. La variable SVED no se ha utilizado hasta el momento, pero cuando se incluyó la línea se comprendió que RMOD sola no sería suficiente. Se escogió para la variable el nombre SVED con la idea de que antes de que fuera necesario guardar algo (en cinta o disco) tendrían que cumplirse ciertas condiciones.

Un nombre más apropiado para esta bandera sería CLAR (para indicar que el archivo está clasificado). La línea 1230 original se ha cambiado por:

```
1230 LET CLAR = 1
```

Ahora hay cuatro estados posibles respecto a la situación del archivo de datos. Éstos son:

RMOD	CLAR	
0	0	No modificado, no clasificado (ilegal)
1	0	Modificado, no clasificado
0	1	No modificado, clasificado
1	1	Modificado, clasificado

RMOD = 0 y CLAR = 0 es ilegal porque el programa asegura que el archivo de datos siempre está clasificado antes de guardarlo. Cuando se ejecuta el programa, RMOD se establece en 0 (línea 1220) para indicar que no se ha efectuado ninguna modificación, y CLAR se establece en 1 (línea 1230) para indicar que el archivo está clasificado.

Toda operación que modifique un registro (como *INCREG*, *BORREG* o *MODREG*) establece RMOD en 1 y esta bandera no cambia con ninguna operación subsiguiente. A CLAR, que inicialmente se establece en 1, la restaura a 0 cualquier actividad que pudiera significar que los datos se hayan desordenado (como en *MODREG* si se altera el campo del nombre). Cualquier actividad que necesite dar por sentado que los datos están clasificados (como *ENCREG*) siempre verifica CLAR y llama a la rutina de clasificación si CLAR = 0. Utilizando estas dos banderas en lugar de RMOD únicamente, el programa puede concluir sin haber guardado un archivo de datos que durante la pertinente ejecución del programa no se hubiera modificado. Y no concluirá sin haberlo antes guardado, en caso de que, después de la modificación de un registro, se haya producido una clasificación.

La otra variable que hasta el momento no ha sido utilizada es CURS. Esta variable se emplea para guardar la posición "corriente" en la matriz de un registro después de que la rutina de búsqueda ha localizado uno. CURS no se restaura después de que se le ha asignado un valor; se la utiliza para llevar información acerca del registro objetivo a otras rutinas del programa. En las líneas 3320 y 3330 se ha modificado el final de la rutina *ENCREG* (búsqueda) para establecer el valor de CURS: o 0 si la búsqueda no da con el registro objetivo; y en MED si la búsqueda tuvo éxito.

La línea 13340 se bifurca a la subrutina *NINREG* si CURS es 0. Con ella lanzamos un mensaje para advertir que no se ha encontrado el registro y visualizamos la clave de búsqueda, NOMCAM$ (TAMA). *NINREG* nos devuelve el menú principal después de que se pulsa la barra espaciadora. *NINREG* se puede modificar con bastante facilidad para darle al usuario la oportunidad de:

```
PULSE RETURN PARA VOLVER A INTENTARLO O
BIEN BARRA ESPACIADORA PARA CONTINUAR
```

Podría parecer que la forma más sencilla de conseguir esto sería llamar *ENCREG* otra vez si se pulsara RETURN. Sin embargo, llamar a una subrutina desde dentro de sí misma, si bien no es "ilegal" en BASIC, sí "confunde" a la dirección de retorno y hará que la subrutina se repita aun cuando usted no lo desee. Se puede obviar el problema, ¡pero la programación empieza a ponerse muy intrincada!

Una forma más sencilla sería utilizar una bandera (como NREG para "ningún registro") y restaurarla en *WINREG*, permitiendo que la subrutina retorne de la manera normal, y forzando un salto hacia atrás, a *EJECUT*, en el programa principal; por ejemplo: 95 IF NREG = 0 THEN 80. Este enfoque se probó y resultó bien. Pero la codificación empezaba a tener un aspecto desaliñado. De acuerdo con nuestro principio de evitar las sentencias GOTO, decidimos mantener las cosas simples y limitarnos a retornar al menú principal si *ENCREG* no encuentra un registro.

Se debe hacer notar una pequeña adición a la línea 10490 en *MODNOM*. La variable numérica S también se debe restaurar (LET S = 0). No hacerlo puede, bajo ciertas condiciones inusuales, hacer que *MODNOM* funcione mal.

La otra rutina ejecutada en esta versión final del programa es *MODREG*. Esta rutina localiza primero el registro a modificar llamando a *ENCREG* (línea 14120). Esta línea llama a la línea 13030, no a la 13000, con el fin de suprimir la sentencia de *ENCREG* para limpiar la pantalla. Si no se puede localizar el registro, el programa retorna al menú principal de forma normal (en la línea 14130). Si se localiza el registro, se deja visualizado en la pantalla el registro objetivo y se les indica a los usuarios:

¿MODIFICAR NOMBRE?
PULSE RETURN PARA ENTRAR EL NUEVO NOMBRE
O BARRA ESPACIADORA PARA SIGUIENTE CAMPO

La rutina que averigua cuál de las dos opciones se requiere se puede hallar desde la línea 14190 hasta la 14280.

De la línea 14190 a la 14220 constituyen un bucle sencillo que sólo termina si se pulsa la barra espaciadora o RETURN. Si A$ no es CHR$(13) (el valor ASCII para un "retorno de carro") y no es un espacio (usted también podría emplear CHR$(32) en lugar de " "), I se restaurará y se repetirá el bucle. Si la tecla pulsada fue RETURN (es decir, que se ha de cambiar el campo del nombre), las líneas siguientes llenarán NOMCAM (CURS) con el nuevo nombre, establecerán RMOD, restaurarán CLAR, llamarán a *MODNOM* y llenarán MODCAM$(CURS) con el nombre estandarizado creado por *MODNOM* y localizado en MODCAM$(TAMA).

El resto de *MODNOM* funciona exactamente de la misma manera. Observe, no obstante, que modificar los otros campos establece RMOD pero no restaura a CLAR (véase línea 14490, p. ej.). La razón de ello es que sólo cambiar el campo del nombre implica que el archivo de datos puede estar desordenado, ya que está ordenado por nombre. Cambiar cualquier otro campo sólo indica que se ha cambiado un registro (RMOD = 1) y que cuando el programa termine se debe guardar el archivo.

La otra rutina ejecutada es *BORREG*, para eliminar un registro. Ésta es muy directa. Primero limpia la pantalla (línea 15020) y visualiza un mensaje explicando lo que está sucediendo. Luego llama a *ENCREG* para localizar el registro a eliminar. Entonces se ofrece una elección: pulsar RETURN para eliminar el registro o la BARRA ESPACIADORA para retornar al menú principal. También se visualiza un mensaje de aviso (línea 15160). Un enfoque aún mejor podría ser responder con un mensaje ¿ESTA SEGURO? si se pulsara RETURN y luego sólo borrar el registro si se pulsara la tecla S (es decir, IF INKEY$ = "S" THEN...).

BORREG no restaura la bandera CLAR. Dado que el archivo ya está en orden alfabético de nombres, borrar un registro completo no alteraría este orden. Sin embargo, sí significa que el archivo se ha modificado y, por consiguiente, RMOD se restaura en la línea 15340 y TAMA se reduce en uno en la línea 13550, para dejar constancia del hecho de que ahora el archivo tiene un registro válido menos. Todos los registros se desplazan "un lugar abajo" desde la línea 15260 hasta la 15320.

Es posible que también haya notado que *ENCREG* incluye una llamada condicionada a una subrutina denominada *LSTCUR* para imprimir el registro en curso localizado por *ENCREG*. Si usted no posee una impresora, simplemente reemplace la línea 13540 por una REM para futura ejecución y omita desde la línea 13600 a la 13690.

Con esto se completa el programa de la agenda de direcciones. Hemos tratado las opciones más importantes presentadas en el menú principal: hallar un registro, agregar un registro, modificar un registro, borrar un registro y salir de! programa. La agenda de direcciones informatizada nos ha servido para ilustrar cómo un programador debe especificar, diseñar y ejecutar un programa. Una modificación esencial que habría de introducir quien tenga la intención de convertir el programa en software de aplicación, sería comprobar (y evitar) el problema que surgiría si TAMA fuera igual a 51. Esto sucedería nada más tener 50 registros en el archivo.

En el próximo capítulo analizaremos los estilos de programación y abordaremos algunos de los aspectos más avanzados de este lenguaje.

Complementos al BASIC

LPRINT

No disponen de esta orden el Commodore 64, Vic-20, BBC Micro y el Dragon 32

En el BBC Micro con una impresora en paralelo, inserte las siguientes líneas:

```
13605 VDU 2
13680 VDU 3
```

Éstas habilitan o inhabilitan, respectivamente, la impresora. Desde la línea 13610 a la 13670 sustituya PRINT por LPRINT. Para más información, consulte el manual del usuario.

En los Commodore, inserte las siguientes líneas:

```
13605 OPEN 4,4:CMD 4
13680 PRINT #4:CLOSE 4
```

Éstas habilitan e inhabilitan, respectivamente, la impresora. Desde la línea 13610 a la 13670, sustituya PRINT por LPRINT.

En el Dragon 32, inserte estas líneas:

```
13605 OPEN"0",-2
13680 CLOSE -2
```

Éstas habilitan e inhabilitan, respectivamente, la impresora. Desde la línea 13610 a la 13670, sustituya PRINT -2, (la coma forma parte de la orden) por LPRINT.

SPECTRUM

Para el Spectrum, el programa de la agenda de direcciones se publicará completo en el próximo capítulo del curso de programación BASIC.

Programa de la agenda de direcciones

```
10 REM 'PROGRAMA PRINCIPAL'
20 REM #INICIL#
30 GOSUB 1000
40 REM #PRESEN#
50 GOSUB 3000
60 REM #ELECCN#
70 GOSUB 3500
80 REM #EJECUT#
90 GOSUB 4000
100 IF OPCNK>9 THEN 60
110 END
1000 REM SUBRUTINA #INICIL#
1010 GOSUB 1100: REM SUBRUTINA #CREMAT#(CREAR MATRICES)
1020 GOSUB 1400: REM SUBRUTINA #LEARCH#(LEER ARCHIVOS)
1030 GOSUB 1600: REM SUBRUTINA #ESBAND#(ESTABLECER BANDERAS)
1050 REM
1060 REM
1070 REM
1080 REM
1090 RETURN
1100 REM SUBRUTINA #CREMAT#(CREAR MATRICES)
1110 DIM NOMCAM$(50)
1120 DIM MODCAM$(50)
1130 DIM CLLCAM$(50)
1140 DIM CIUCAM$(50)
1150 DIM PROCAM$(50)
1160 DIM TELCAM$(50)
1170 DIM INDCAM$(50)
1180 REM
1190 REM
1200 REM
1210 LET TAMA = 0
1220 LET RMOD = 0
1230 LET CLAR = 1
1240 LET CURS = 0
1250 REM
1260 REM
1270 REM
1280 REM
1290 REM
1300 RETURN
1400 REM SUBRUTINA #LEARCH#
1410 OPEN "I",#1,"DAT.AGCO"
1420 INPUT #1,TEST$
1430 IF TEST$ = "@VACIO" THEN GOTO 1530: REM CERRAR Y RETORNAR
1440 LET NOMCAM$(1) = TEST$
1450 INPUT #1,MODCAM$(1),CLLCAM$(1),CIUCAM$(1),PROCAM$(1),TELCAM$(1)
1460 INPUT #1,INDCAM$(1)
1470 LET TAMA = 2
1480 FOR L = 2 TO 50
1490 INPUT #1,NOMCAM$(L),MODCAM$(L),CLLCAM$(L),CIUCAM$(L),
     PROCAM$(L)
1500 INPUT #1,TELCAM$(L),INDCAM$(L)
1510 LET TAMA = TAMA + 1
1520 IF EOF(1) = -1 THEN LET L = 50
1530 NEXT L
1540 CLOSE #1
1550 RETURN
1600 REM SUBRUTINA #ESBAND#
1610 REM ESTABLECE BANDERAS DESPUES DE #LEARCH#
1620 REM
1630 REM
1640 IF TEST$ = "@VACIO" THEN LET TAMA = 1
1650 REM
1660 REM
1670 REM
1680 REM
1690 RETURN
3000 REM SUBRUTINA #PRESEN#
3010 PRINT CHR$(12): REM LIMPIAR PANTALLA
3020 PRINT
3030 PRINT
3040 PRINT
3050 PRINT
3060 PRINT TAB(11);"#BIEN VENIDO A LA#"
3070 PRINT TAB(9);"#AGENDA COMPUTERIZADA#"
3080 PRINT TAB(12);"#DE MI COMPUTER#"
3090 PRINT
3100 PRINT TAB(0);"(PULSE BARRA ESPACIADORA PARA CONTINUAR)"
3120 IF INKEY$<> " " THEN L = 0
3130 NEXT L
3140 PRINT CHR$(12)
3150 RETURN
3500 REM SUBRUTINA #ELECCN#
3510 REM
3520 IF TEST$ = "@VACIO" THEN GOSUB 3860: REM SUBRUTINA #PRIMERA#
3530 IF TEST$ = "@VACIO" THEN RETURN
3540 REM 'IMMENU'
3550 PRINT CHR$(12)
3560 PRINT "SELECCIONE UNO DE LOS SIGUIENTES"
3570 PRINT
3580 PRINT
3590 PRINT
3600 PRINT "1. HALLAR REGISTRO (DE NOMBRE)"
3610 PRINT "2. HALLAR NOMBRES (DE NOMBRE INCOMPLETO)"
3620 PRINT "3. HALLAR REGISTRO (DE CIUDAD)"
3630 PRINT "4. HALLAR REGISTRO (DE INICIAL)"
3640 PRINT "5. LISTAR TODOS LOS REGISTROS"
3650 PRINT "6. AGREGAR REGISTRO NUEVO"
3660 PRINT "7. MODIFICAR REGISTRO"
3670 PRINT "8. BORRAR REGISTRO"
3680 PRINT "9. SALIR Y GUARDAR"
3690 PRINT
3700 PRINT
3710 REM 'ASOPCN'
3720 REM
3730 LET L = 0
3740 LET I = 0
3750 FOR L = 1 TO 1
3760 PRINT "DE ENTRADA A OPCION (1 - 9)"
3770 FOR I = 1 TO 1
3780 LET A$ = INKEY$
3790 IF A$ = "" THEN I = 0
3800 NEXT I
3810 LET OPCNK1 THEN L = 0
3810 LET OPCN = VAL(A$)
3820 IF OPCN < 1 THEN L = 0
3830 IF OPCN > 9 THEN L = 0
3840 NEXT L
3850 RETURN
3860 REM SUBRUTINA #PRIMERA#(VISUALIZAR MENSAJE)
3870 LET OPCN = 6
3880 PRINT CHR$(12): REM LIMPIAR PANTALLA
3890 PRINT
3900 PRINT TAB(10);"NO HAY REGISTROS EN"
3910 PRINT TAB(7);"EL ARCHIVO. DEBERA EMPEZAR"
3920 PRINT TAB(8);"POR AGREGAR UN REGISTRO"
3930 PRINT
3940 PRINT TAB(0);"(PULSE BARRA ESPACIADORA PARA CONTINUAR)"
3960 IF INKEY$ <> " " THEN B = 0
3970 NEXT B
3980 PRINT CHR$(12): REM LIMPIAR PANTALLA
3990 RETURN
4000 REM SUBRUTINA #EJECUT#
4010 REM
4020 REM
4030 REM
4040 IF OPCN = 1 THEN GOSUB 13000: REM #INCREG#
4050 REM 2 ES #ENCNOM#
4060 REM 3 ES #ENCCIU#
4070 REM 4 ES #ENCINI#
4080 REM 5 ES #LISREG#
4090 IF OPCN = 6 THEN GOSUB 10000: REM #INCREG#
4100 IF OPCN = 7 THEN GOSUB 14000: REM #MODREG#
4110 IF OPCN = 8 THEN GOSUB 15000: REM #BORREG#
4120 IF OPCN = 9 THEN GOSUB 11000: REM #SAPROG#
4130 REM
4140 RETURN
10000 REM SUBRUTINA #INCREG#
10010 PRINT CHR$(12): REM LIMPIAR PANTALLA
10020 INPUT "DE ENTRADA AL NOMBRE";NOMCAM$(TAMA)
10030 INPUT "DE ENTRADA A LA CALLE";CLLCAM$(TAMA)
10040 INPUT "DE ENTRADA A LA CIUDAD";CIUCAM$(TAMA)
10050 INPUT "DE ENTRADA A LA PROVINCIA";PROCAM$(TAMA)
10060 INPUT "DE ENTRADA AL NUMERO DE TELEFONO";TELCAM$(TAMA)
10070 LET RMOD=1: LET CLAR=0: REM MODIFICADO Y NO CLASIFICADO
10080 LET INDCAM$(TAMA) = STR$(TAMA)
10090 LET TEST$ = ""
10100 GOSUB 10200: REM #MODNOM#
10110 LET OPCN = 0
10120 LET TAMA = TAMA + 1
10130 REM
10140 REM
10150 RETURN
10200 REM RUTINA #MODNOM#
10210 REM CONVIERTE CONTENIDO DE NOMCAM$ A MAYUSCULAS,
10220 REM ELIMINA CARACTERES EXTRAÑOS Y ALMACENA EN EL ORDEN:
10230 REM APELLIDO+ESPACIO+NOMBRE DE PILA EN MODCAM$
10240 REM
10250 LET N$ = NOMCAM$(TAMA)
10260 FOR L = 1 TO LEN(N$)
10270 LET TEMP$ = MID$(N$,L,1)
10280 LET T = ASC(TEMP$)
10290 IF T > = 97 THEN T = T - 32
10300 LET TEMP$ = CHR$(T)
10310 LET P$ = P$ + TEMP$
10320 NEXT L
10330 LET N$ = P$
10340 REM LOCALIZAR ULTIMO ESPACIO
10350 FOR L = 1 TO LEN(N$)
10360 IF MID$(N$,L,1) = " " THEN S = L
10370 NEXT L
10380 REM ELIMINAR CARACTERES EXTRAÑOS Y ALMACENAR
10390 REM NOMBRE DE PILA EN CNOM$
10400 FOR L = 1 TO S - 1
10410 IF ASC(MID$(N$,L,1)) > 64 THEN CNOM$ = CNOM$+MID$(N$,L,1)
10420 NEXT L
10430 REM ELIMINAR CARACTERES EXTRAÑOS Y ALMACENAR
10440 REM APELLIDO EN SNOM$
10450 FOR L = S + 1 TO LEN(N$)
10460 IF ASC(MID$(N$,L,1)) > 64 THEN SNOM$ = SNOM$ + MID$
      (N$,L,1)
10470 NEXT L
10480 LET MODCAM$(TAMA) = SNOM$ + " " + CNOM$
10490 LET P$ = " ": LET N$ = " ": LET SNOM$ = " ": LET
      CNOM$ = " ": LET S = 0
10500 RETURN
11000 REM SUBRUTINA #SAPROG#
11010 REM CLASIFICA Y GUARDA ARCHIVO
11020 REM SI ALGUN REGISTRO HA SIDO
11030 REM MODIFICADO (RMOD = 1)
11040 REM O NO HA SIDO CLASIFICADO (CLAR = 0)
11050 REM RMOD = 0 Y CLAR = 0 ES ILEGAL
11060 REM
11070 IF RMOD = 0 AND CLAR = 1 THEN RETURN
11080 IF RMOD = 1 AND CLAR = 0 THEN GOSUB 11200: REM #CLSREG#
11090 GOSUB 12000: REM #GRDREG#
11100 RETURN
11200 REM SUBRUTINA #CLSREG#
11210 REM CLASIFICA TODOS LOS REGISTROS EN ORDEN ALFABETICO
11220 REM MEDIANTE MODCAM$ Y ACTUALIZA INDCAM$
11230 REM
11240 REM
11250 LET S = 0
11260 FOR L = 1 TO TAMA - 2
11270 IF MODCAM$(L)>MODCAM$(L + 1) THEN GOSUB 11350
11280 NEXT L
11290 IF S = 1 THEN 11250
11300 REM
11310 REM
11320 LET CLAR = 1: REM ESTABLECE BANDERA 'ARCHIVO CLASIFICADO'
11330 REM
11340 RETURN
11350 REM SUBRUTINA #INTERCALAR#
11360 LET TNOMC$ = NOMCAM$(L)
11370 LET TMODC$ = MODCAM$(L)
11380 LET TCLLC$ = CLLCAM$(L)
```

```
11390 LET TCIUC$ = CIUCAM$(L)
11400 LET TPROC$ = PROCAM$(L)
11410 LET TTELC$ = TELCAM$(L)
11420 REM
11430 LET NOMCAM$(L) = NOMCAM$(L + 1)
11440 LET MODCAM$(L) = MODCAM$(L + 1)
11450 LET CLLCAM$(L) = CLLCAM$(L + 1)
11460 LET CIUCAM$(L) = CIUCAM$(L + 1)
11470 LET PROCAM$(L) = PROCAM$(L + 1)
11480 LET TELCAM$(L) = TELCAM$(L + 1)
11490 LET INDCAM$(L) = STR$(L)
11500 REM
11510 LET NOMCAM$(L + 1) = TNOMC$
11520 LET MODCAM$(L + 1) = TMODC$
11530 LET CLLCAM$(L + 1) = TCLLC$
11540 LET CIUCAM$(L + 1) = TCIUC$
11550 LET PROCAM$(L + 1) = TPROC$
11560 LET TELCAM$(L + 1) = TTELC$
11570 LET INDCAM$(L + 1) = STR$(L + 1)
11580 LET S = 1
11590 REM
11600 RETURN
12000 REM SUBRUTINA #GRDREG#
12010 REM
12020 REM
12030 OPEN "O",#1,"DAT.AGCO"
12040 REM
12050 FOR L = 1 TO TAMA - 1
12060 PRINT #1,NOMCAM$(L);",";MODCAM$(L);",";CLLCAM$(L);",";CIUCAM$(L)
12070 PRINT #1,PROCAM$(L);",";TELCAM$(L);",";INDCAM$(L)
12080 NEXT L
12090 REM
12100 REM
12110 REM
12120 REM
12130 CLOSE #1
12140 REM
12150 RETURN
13000 REM SUBRUTINA #ENCREG# (HALLAR REGISTRO)
13010 PRINT CHR$(12): REM LIMPIAR PANTALLA
13020 REM
13030 IF CLAR = 0 THEN GOSUB 11200: REM #CLSREG#
13040 PRINT
13050 PRINT
13060 PRINT TAB(10);"BUSCANDO UN REGISTRO"
13070 PRINT TAB(13);"POR EL NOMBRE"
13080 PRINT
13090 PRINT TAB(7);"DIGITE EL NOMBRE COMPLETO"
13100 PRINT TAB(3);"POR ORDEN NOMBRE DE PILA APELLIDO"
13110 PRINT
13120 PRINT
13130 REM
13140 INPUT "EL NOMBRE ES ";NOMCAM$(TAMA)
13150 GOSUB 10200: REM SUBRUTINA #MODNOM#
13160 LET SCHKEY$ = MODCAM$(TAMA)
13170 REM
13180 REM
13190 REM
13200 REM
13210 REM
13220 LET INF = 1
13230 LET SUP = TAMA - 1
13240 FOR L = 1 TO 1
13250 LET MED = INT((INF+SUP)/2)
13260 IF MODCAM$(MED) <> BUSCLV$ THEN L = 0
13270 IF MODCAM$(MED) < BUSCLV$ THEN INF = MED + 1
13280 IF MODCAM$(MED) > BUSCLV$ THEN SUP = MED - 1
13290 IF INF > SUP THEN L = 1
13300 NEXT L
13310 REM
13320 IF INF > SUP THEN LET CURS = 0
13330 IF INF <= SUP THEN LET CURS = MED
13340 IF CURS = 0 THEN GOSUB 13700: REM #NINREG#
13350 IF CURS = 0 THEN RETURN
13360 REM
13370 REM
13380 PRINT CHR$(12)
13390 PRINT
13400 PRINT TAB(11);"#HALLADO REGISTRO#"
13410 PRINT
13420 PRINT "NOMBRE:",NOMCAM$(CURS)
13430 PRINT "CALLE:",CLLCAM$(CURS)
13440 PRINT "CIUDAD:",CIUCAM$(CURS)
13450 PRINT "PROVINCIA:",PROCAM$(CURS)
13460 PRINT "TELEFONO:",TELCAM$(CURS)
13470 PRINT
13480 PRINT TAB(2);"PULSE CUALQUIER LETRA PARA IMPRIMIR"
13490 PRINT TAB(3);"O BARRA ESPACIADORA PARA CONTINUAR"
13500 FOR I = 1 TO 1
13510 LET A$ = INKEY$
13520 IF A$ = "" THEN I = 0
13530 NEXT I
13540 IF A$<> " " THEN GOSUP 13600: REM #LSTCUR#
13550 RETURN
13600 REM SUBRUTINA #LSTCUR# (LISTAR REGISTRO EN CURSO)
13610 LPRINT
13620 LPRINT "NOMBRE:",NOMCAM$(CURS)
13630 LPRINT "CALLE:",CLLCAM$(CURS)
13640 LPRINT "CIUDAD:",CIUCAM$(CURS)
13650 LPRINT "PROVINCIA:",PROCAM$(CURS)
13660 LPRINT "TELEFONO:",TELCAM$(CURS)
13670 LPRINT
13680 LPRINT
13690 RETURN
13700 REM SUBRUTINA #NINREG# (NO HALLADO REGISTRO)
13710 PRINT CHR$(12): REM LIMPIAR PANTALLA
13720 PRINT TAB(9);"#NO HALLADO REGISTRO#"
13730 PRINT TAB(4);"#EN LA FORMA: ";NOMCAM$(TAMA);" #"
13740 PRINT
13750 PRINT TAB(0);"(PULSE BARRA ESPACIADORA PARA CONTINUAR)"
13760 FOR I = 1 TO 1
13770 IF INKEY$<>" " THEN I = 0
13780 NEXT I
13790 RETURN
14000 REM SUBRUTINA #MODREG# (MODIFICAR REGISTRO)
14010 REM
14020 PRINT CHR$(12): REM LIMPIAR PANTALLA
14030 PRINT
14040 PRINT
14050 PRINT
14060 PRINT
14070 PRINT TAB(6);"#PARA MODIFICAR UN REGISTRO#"
14080 PRINT TAB(1);"#LOCALICE PRIMERO EL REGISTRO DESEADO#"
14090 REM
14100 REM
14110 REM
14120 GOSUB 13030: REM SUBRUTINA #ENCREG# SIN CLS
14130 IF CURS = 0 THEN RETURN: REM NO HALLADO REGISTRO
14140 PRINT
14150 PRINT TAB(11);"¿MODIFICAR NOMBRE?"
14160 PRINT
14170 PRINT TAB(0);"PULSE RETURN PARA ENTRAR EL NUEVO NOMBRE"
14180 PRINT TAB(0);"O BARRA ESPACIADORA PARA SIGUIENTE CAMPO"
14190 FOR I = 1 TO 1
14200 LET A$ = INKEY$
14210 IF A$<>CHR$(13) AND A$<>" " THEN I = 0
14220 NEXT I
14230 IF A$ = CHR$(13) THEN INPUT "NUEVO NOMBRE";NOMCAM$(CURS)
14240 IF A$ = CHR$(13) THEN RMOD = 1
14250 IF A$ = CHR$(13) THEN CLAR = 0
14260 IF A$ = CHR$(13) THEN NOMCAM$(TAMA) = NOMCAM$(CURS)
14270 IF A$ = CHR$(13) THEN GOSUB 10200: REM SUBRUTINA #MODNOM#
14280 IF A$ = CHR$(13) THEN LET MODCAM$(CURS) = MODCAM$(TAMA)
14290 PRINT
14300 PRINT TAB(11);"¿MODIFICAR CALLE?"
14310 PRINT
14320 PRINT TAB(0);"PULSE RETURN PARA ENTRAR LA NUEVA CALLE"
14330 PRINT TAB(0);"O BARRA ESPACIADORA PARA SIGUIENTE CAMPO"
14340 FOR I = 1 TO 1
14350 LET A$ = INKEY$
14360 IF A$<> CHR$(13) AND A$<> " " THEN I = 0
14370 NEXT I
14380 IF A$ = CHR$(13) THEN RMOD = 1
14390 IF A$ = CHR$(13) THEN INPUT "NUEVA CALLE";CLLCAM$(CURS)
14400 PRINT
14410 PRINT TAB(11);"¿MODIFICAR CIUDAD?"
14420 PRINT
14430 PRINT TAB(0);"PULSE RETURN PARA ENTRAR LA NUEVA CIUDAD"
14440 PRINT TAB(0);"O BARRA ESPACIADORA PARA SIGUIENTE CAMPO"
14450 FOR I = 1 TO 1
14460 LET A$ = INKEY$
14470 IF A$<> CHR$(13) AND A$<>" " THEN I = 0
14480 NEXT I
14490 IF A$ = CHR$(13) THEN RMOD = 1
14500 IF A$ = CHR$(13) THEN INPUT "NUEVA CIUDAD";CIUCAM$(CURS)
14510 PRINT
14520 PRINT TAB(9);"¿MODIFICAR PROVINCIA?"
14530 PRINT
14540 PRINT TAB(0);"PULSE RETURN PARA ENTRAR LA NUEVA PROVIN"
14550 PRINT TAB(0);"O BARRA ESPACIADORA PARA SIGUIENTE CAMPO"
14560 FOR I = 1 TO 1
14570 LET A$ = INKEY$
14580 IF A$<> CHR$(13) AND A$<>" " THEN I = 0
14590 NEXT I
14600 IF A$ = CHR$(13) THEN RMOD = 1
14610 IF A$ = CHR$(13) THEN INPUT "NUEVA PROVINCIA";PROCAM$
      (CURS)
14620 PRINT
14630 PRINT TAB(5);"¿MODIFICAR NUMERO DE TELEFONO?"
14640 PRINT
14650 PRINT TAB(0);"PULSE RETURN PARA ENTRAR NUEVO TELEFONO"
14660 PRINT TAB(3);"O BARRA ESPACIADORA PARA CONTINUAR"
14670
14680 LET A$ = INKEY$
14690 IF A$<>CHR$(13) AND A$<>" " THEN I = 0
14700 NEXT I
14710 IF A$ = CHR$(13) THEN RMOD = 1
14720 IF A$ = CHR$(13) THEN INPUT "NUEVO TELEF";TELCAM$ (CURS)
14730 REM
14740 REM
14750 RETURN
15000 REM SUBRUTINA #BORREG# (BORRAR REGISTRO)
15010 REM
15020 PRINT CHR$(12): REM LIMPIAR PANTALLA
15030 PRINT
15040 PRINT
15050 PRINT
15060 PRINT
15070 PRINT TAB(7);"#PARA BORRAR UN REGISTRO#"
15080 PRINT TAB(1);"#LOCALICE PRIMERO EL REGISTRO DESEADO#"
15090 REM
15100 REM
15110 REM
15120 GOSUB 13030: REM SUBRUTINA #ENCREG# SIN CLS
15130 IF CURS = 0 THEN RETURN: REM NO HALLADO REGISTRO
15140 PRINT
15150 PRINT TAB(6);"¿DESEA BORRAR ESTE REGISTRO?"
15160 PRINT TAB(1);"#AVISO#-- NO HAY SEGUNDA OPORTUNIDAD"
15170 PRINT
15180 PRINT TAB(8);"PULSE RETURN PARA BORRAR"
15190 PRINT TAB(3);"O BARRA ESPACIADORA PARA CONTINUAR"
15200 FOR I = 1 TO 1
15210 LET A$ = INKEY$
15220 IF A$<>CHR$(13) AND A$<>" " THEN I = 0
15230 NEXT I
15240 IF A$ = " " THEN RETURN
15250 FOR L = CURS TO TAMA - 2
15260 LET NOMCAM$(L) = NOMCAM$(L + 1)
15270 LET MODCAM$(L) = MODCAM$(L + 1)
15280 LET CLLCAM$(L) = CLLCAM$(L + 1)
15290 LET CIUCAM$(L) = CIUCAM$(L + 1)
15300 LET PROCAM$(L) = PROCAM$(L + 1)
15310 LET TELCAM$(L) = TELCAM$(L + 1)
15320 LET INDCAM$(L) = STR$(L)
15330 NEXT L
15340 LET RMOD = 1
15350 LET TAMA = TAMA - 1
15360 REM
15370 REM
15380 REM
15390 RETURN
```

Grace Hopper

Grace Hopper fue una gran impulsora de los lenguajes de alto nivel, y a ella se debe la identificación del primer "bug"

Cortesía de Sperry Ltd.

Suele pensarse que la ciencia de la informática por lo general es un club "sólo para hombres". Pero las mujeres van ocupando cada vez más posiciones junto a los hombres, en pie de igualdad, en el desarrollo y la aplicación de los ordenadores. Una de las pioneras de la informática fue Grace Hopper, cuya aportación más trascendente revolucionó el campo del software: fue la autora del primer compilador y colaboró de manera destacada en la elaboración y puesta a punto del lenguaje COBOL. También fue la primera persona que aisló un *bug* (véase p. 433) en un ordenador y logró "depurarlo".

Después de hacer un trabajo de posgrado en Yale, Grace Hopper regresó a su universidad de origen, Vassar, como miembro de la facultad de matemáticas. Allí permaneció hasta la edad de 39 años, cuando fue llamada para el servicio de guerra en el Naval Ordinance Computation Project. En 1945 se le ordenó que fuera a la Universidad de Harvard para ayudar al físico Howard Aiken en la construcción de un ordenador. En 1937 Aiken había propuesto a la IBM la idea de construir un ordenador utilizando equipos de tabulación adaptados. Su primer ordenador, aunque de diseño mecánico, tuvo el éxito suficiente como para animar a IBM a invertir en un modelo mejorado que pudiera emplear relés electromecánicos. Así se construyó una máquina que se denominó Harvard Mark II.

En aquellos primeros días las máquinas habían de programarse volviendo a tender sus cables para cada nueva tarea. De modo que, en el caluroso verano de 1945, Grace Hopper se encontró literalmente atrapada entre los cables del ordenador. Las necesidades de la guerra exigían con toda urgencia el procesamiento de datos balísticos, y el comandante Aiken solía irrumpir en la sala para preguntarle: "¿Por qué no está usted haciendo números, Hopper?" Una avería del ordenador lo impedía. Cuando finalmente se descubrió que el fallo se debía a una enorme polilla que había penetrado por las ventanas abiertas y que había quedado atrapada en el interruptor de un relé, Grace le replicó sucintamente: "¡Estamos desinsectando la máquina!" Este primer *bug* del que se tiene constancia se extrajo con sumo cuidado del relé con un par de pinzas y se conserva en el Museo Naval de Virginia en el cuaderno de bitácora del Harvard Mark II. Está pegado con cola junto a la entrada de las 15.45 del 9 de septiembre de 1945.

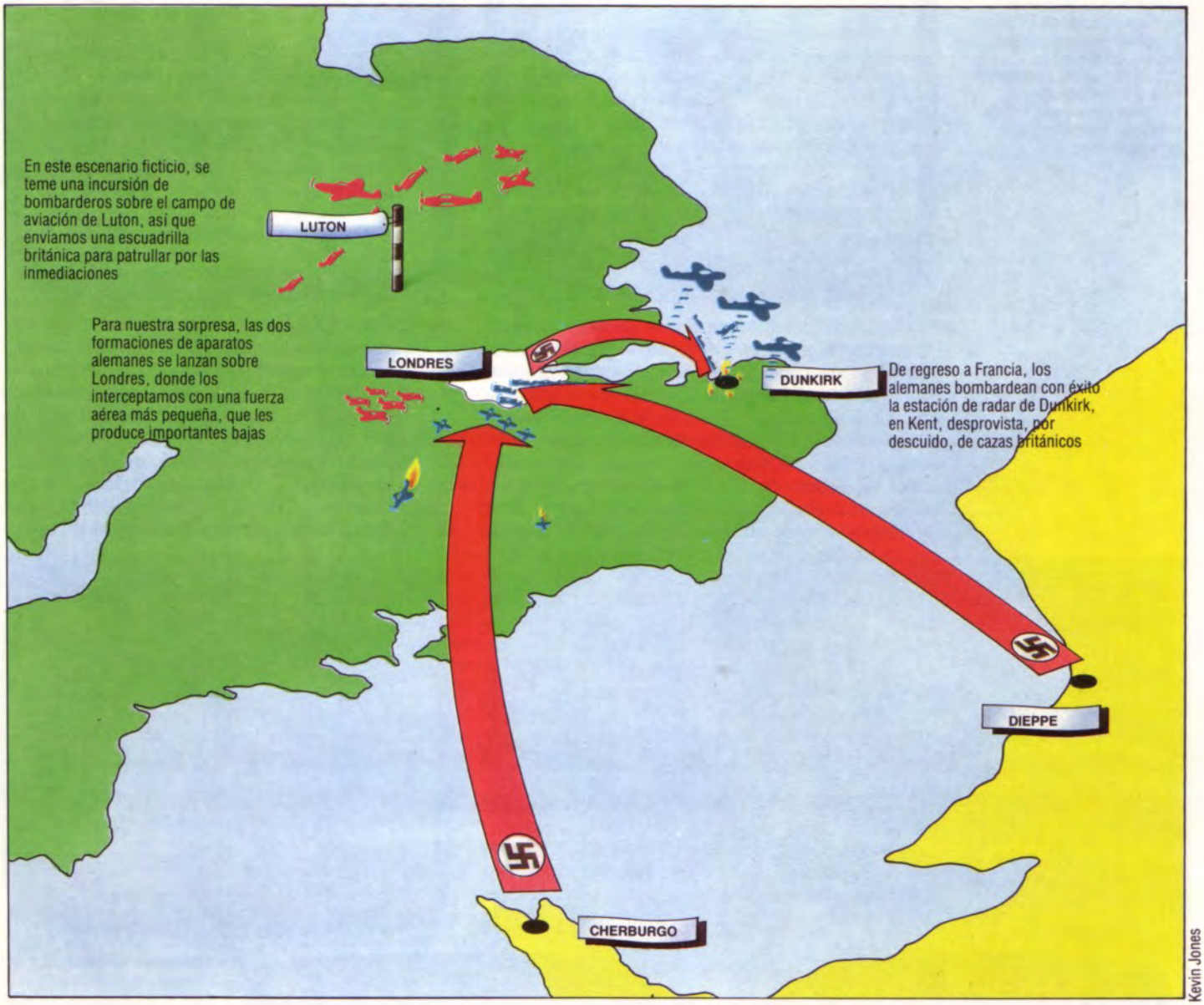
Aquel mismo año dos ingenieros, John Mauchly y Presper Eckert, estaban construyendo otro ordenador, el ENIAC (véase p. 46). Finalizada la guerra, los dos hombres crearon su propia empresa para fabricar una versión comercial de la máquina, e invitaron a Grace a unirse a su equipo. La ayuda más valiosa que aportó al desarrollo de este ordenador, denominado UNIVAC (UNIVersal ACcounting Machine) la constituyó la creación de software para el mismo. Y fue mientras intentaba construir programas de aplicaciones empresariales en el UNIVAC cuando Grace buscó por primera vez un atajo para ahorrarse la necesidad de volver a escribir ciertas subrutinas que se repetían una y otra vez. Valiéndose de lo que entonces se tuvo como brillante estratagema, la de que un ordenador podía escribir sus *propios* programas, Grace creó el primer lenguaje de programación, junto con el compilador necesario para traducirlo a código de lenguaje máquina. Se le dio el nombre de "A-0". Cuando este compilador fue presentado en público por primera vez suscitó la incredulidad entre los profesionales de la informática, quienes pensaban que las máquinas sólo podían realizar operaciones aritméticas o manipular símbolos. Se quedaron atónitos viendo cómo un ordenador saltaba a una subrutina de su biblioteca al encontrarse con un verbo en imperativo al comienzo de lo que era casi una sencilla locución en inglés.

En mayo de 1959 la capitana Hopper fue invitada al Pentágono para formar parte de una comisión de trabajo que estaba intentando crear y estandarizar un lenguaje para ordenadores de uso comercial. En menos de un año la comisión dio a la luz la primera versión del COmmon Business Oriented Language (COBOL). Grace colaboró valiosamente en los esfuerzos de la comisión por aunar los mejores logros de cada uno de los lenguajes existentes y, por consiguiente, crear un lenguaje óptimo para las empresas en virtud de su calidad. Prueba del éxito que consiguió la comisión es que el COBOL continúa siendo hoy en día uno de los lenguajes más ampliamente utilizados.

COBOL

El COBOL fue uno de los primeros lenguajes de programación que se escribieron con la intención de hacerlos fácilmente accesibles a los no matemáticos. El lenguaje favorece la utilización de procedimientos generalizados escritos en un estilo narrativo del inglés, en vez de rutinas codificadas sólo válidas para un problema determinado.
Un programa en COBOL consta de cuatro unidades. El nombre del programa, su autor y otra información de referencia constituyen la división de identificación (*Id-division*). Aunque se supone que los programas en COBOL se pueden ejecutar en diferentes máquinas, todos los detalles relativos al ordenador para el cual se escribió originalmente el programa se incluyen en la división Entorno (*Environment division*). Y dado que en muchas partes del mismo programa se podrían emplear los mismos datos, el COBOL posee una división de Datos (*Data division*) aparte. Por último, los procedimientos que manipularán los datos se codifican en la división Procedimientos (*Procedure division*)

Guerra y paz



Kevin Jones

Muchos juegos ponen a prueba la aptitud como estratega militar del usuario, simulando batallas históricas o hipotéticas

En la actualidad, los generales modernos conceden una gran importancia a los juegos de guerra donde planificar respuestas adecuadas a un supuesto ataque o conspiración. Para jugar a estos sofisticados juegos se han ideado complicados sistemas de hardware y software combinando todos los aspectos conocidos de un conflicto potencial, como el despliegue inicial de las fuerzas aliadas y enemigas, las condiciones de aprovisionamiento, la retaguardia, etc. El sistema incluye hasta las condiciones climatológicas adversas, los cambios de táctica del enemigo, los efectos de las actividades de la quinta columna, y cualquier otra variable que tenga que ver con el éxito de una operación militar. Una de las principales tareas encomendadas al sistema

Ian McKinnell

La Batalla de Inglaterra
Fighter Command (ideado por Strategic Simulations Inc. para el Apple) es uno de los típicos juegos de estrategia bélica para microordenadores. Antes de comenzar, el jugador debe elegir entre una gama de opciones muy amplia, inclusive el tipo de avión a utilizar y las condiciones meteorológicas. La acción se visualiza en forma de símbolos que se mueven sobre el mapa, junto con una información adicional en forma de texto. El paquete y la documentación incluyen un mapa impreso con piezas de cartón para una referencia visual suplementaria

Tactical Armour Command
El TAC (de Avalon Hill) es utilizable por los ordenadores Atari, Apple, IBM PC y Commodore 64. Simula un combate de acorazados durante la segunda guerra mundial. El TAC es para uno o dos jugadores, quienes seleccionan entre cinco escenarios diferentes y manipulan fuerzas británicas, norteamericanas, rusas y alemanas

Ian McKinnell

NORAD de defensa por radar de las montañas Cheyenne (Wyoming) —descrito en la película *Juegos de guerra (War games)*— es la de determinar, actualizar y calibrar de forma continua la potencia relativa de Estados Unidos y de la Unión Soviética y asesorar la preparación de una respuesta ante cualquier eventualidad.

Por supuesto, los juegos de guerra para generales amateurs son menos sutiles. Con el fin de crear el adecuado nivel de complejidad, el jugador de guerra ha debido remitirse a hojas con tablas, voluminosos libros de reglas y un gran número de dados. La indecible fatiga que supone la práctica de juegos de guerra los ha ido restringiendo a un grupo de entusiastas no tan reducido. Sin embargo, con el advenimiento del ordenador personal y la disponibilidad de programas para juegos de guerra, aquel tedioso "trabajo de Estado Mayor" ha desaparecido y ha dejado en su lugar un absorbente juego que ofrece tanto incitación como desafío, al igual que cualquier otro tipo de juegos para ordenador de los que se encuentran habitualmente en el mercado.

Hay una riquísima variedad de juegos. Se puede recrear o simular prácticamente cualquier tipo de operación militar desde los tiempos de la Grecia antigua hasta una teórica confrontación entre la OTAN y el Pacto de Varsovia allá por el día de sanjamás. Se pueden librar batallas aéreas, navales, guerras de galaxias e incluso guerras entre míticos imperios. El panorama es ilimitado.

Los juegos históricos ofrecen la posibilidad de descubrir por qué Napoleón se equivocó en Waterloo o de intentar ser más listos que Hitler desbaratando su intento de invadir Rusia en 1941. Los juegos cósmicos desafían aún más nuestra inventiva. No sólo podemos maniobrar una escuadra de naves interestelares a través de las galaxias, sino también especificar la clase de naves que deseamos. Por supuesto, bajo determinados compromisos. A cambio de mayor velocidad puede que debamos sacrificar armamento, y un refuerzo antibalas de la pantalla quizá se consiga a cambio de menor aprovisionamiento de combustible. El juego consiste en optar por el compromiso que mejor se adapte al estilo con que libramos una batalla.

A diferencia de sus equivalentes convencionales, los juegos de estrategia por ordenador no exigen ninguna pericia ni conocimientos previos, y la

Cortesía de Soft

Ian McKinnell

Legionnaire (Legionario)
Esta simulación del arte de la guerra entre las fuerzas de César (usted) y los bárbaros (el ordenador: un Apple o un Atari) se juega en tiempo real. La infantería, la caballería y las otras fuerzas se representan por medio de símbolos, que se pueden seleccionar y desplazar mediante el cursor (cuadrado blanco) controlado por una palanca de mando. El juego es una producción de Avalon Hill

mayoría de ellos viene con breves notas y pistas para los principiantes. No obstante, conviene saber que algunos juegos se clasifican como de iniciación, intermedios o avanzados. Si usted se dispone a practicar juegos de estrategia, lo mejor que puede hacer es comenzar por el nivel de iniciación, y asimilar los conceptos básicos de los juegos de guerra y las simulaciones estratégicas antes de probar con los juegos de niveles más avanzados.

El formato de los juegos varía según los diferentes tipos de operaciones militares representadas. Por lo general se practican sobre grandes mapas. Y si son tan extensos que no se pueden visualizar en una sola pantalla, entonces ésta suele actuar como una ventana móvil (dirigida por una palanca de mando) a través del mapa. En las figuraciones históricas, los diseñadores de juegos intentan reproducir, con la mayor fidelidad posible, el terreno sobre el cual se libró la batalla original. En *Computer Bismarck*, de SSI, la acción se desarrolla en el Atlántico Norte, lo que no supuso demasiados problemas en cuanto al diseño de los gráficos. Por el contrario, para otro juego de SSI, *Battle for Normandy* (La batalla de Normandía), el trabajo de los diseñadores fue muchísimo más difícil. No sólo el terreno general debía ser el apropiado, sino también los lugares específicos, como las playas y la costa, las ciudades, los pueblos y los ríos. En los juegos no históricos, el diseñador tiene amplio margen para hacer que el jugador se centre sobre todo en las fuerzas de que dispone, pero aun en este caso el diseñador ha de tener mucho cuidado de incluir los obstáculos y las compensaciones suficientes para evitar que el juego sea demasiado sencillo por uno u otro bando.

El mapa tiene también una cuadrícula superpuesta. Esta cuadrícula trocea el mapa al modo como se recuadra un tablero de ajedrez, si bien para los mapas para juegos de guerra suelen preferirse retículas hexagonales antes que cuadradas. A cada cuadrado o hexágono se le otorga un valor según el tipo de terreno que encierre. Este valor representa el grado de dificultad que tendría una unidad para alcanzar esa zona o atravesarla. El esfuerzo invertido en tales desplazamientos se traduce en una reducción de la capacidad de movimiento según el valor acordado. Cuando el margen de movimiento de la unidad equivale a cero, o es inferior al valor de la zona en la que se propone entrar, cabe la posibilidad de que en esa jugada ya no pueda avanzar más.

Por lo general el juego se divide en una cantidad de "jugadas" que representan el tiempo transcurrido, y cada jugador se asigna unos objetivos que debe alcanzar en el tiempo disponible si quiere ganar. En la mayoría de los casos no es necesario, o ni siquiera posible, alcanzar todos los objetivos fijados. De modo que la primera decisión que ha de tomar el jugador es juzgar cuáles son sus posibilidades y, en función de ellas, determinar sus prioridades estratégicas. En tales circunstancias el papel del contrincante consiste en evitar que el atacante consiga los objetivos fijados. Tampoco a él le será posible protegerlo todo, de modo que habrá de decidir cuándo abandonar defensas desesperadas, durante cuánto tiempo resistir en las plazas fuertes, y si correr o no el riesgo de lanzar contraataques para recuperar posiciones perdidas o desbaratar los preparativos de su contrario para un nuevo ataque.

El jugador se comunica con el programa a través de la representación gráfica y textual de las fuerzas bajo su mando que están en el mapa. La visualización gráfica representa la situación de una unidad determinada en el campo de batalla y la textual proporciona información relativa a la eficacia en combate de la unidad y al margen de movimiento. El jugador desplaza sus unidades señalándolas mediante un cursor o haciendo que el ordenador se las presente en rotación. Una vez escogida la unidad, se da la orden de desplazamiento. En el caso de un mapa con divisiones hexagonales, 1 enviaría la unidad hacia el norte, 2 la enviaría hacia el noreste, y así sucesivamente a los puntos de la brújula. Es cada vez mayor el número de estos juegos que funcionan a base de palancas de mando o mandos de bola, los cuales "toman" una unidad y la mueven simplemente. Para dar por terminado el desplazamiento se suele utilizar la orden FINISH o F. Aun entonces, algunos juegos ofrecen la variante de permitir que el jugador vuelva a señalar la unidad y la mueva otra vez, a menos que haya agotado todas sus posibilidades y ya no le quede margen de movimiento. Cuando se han completado todos los movimientos, el jugador se lo indica al ordenador mediante la orden EXECUTE o E. El ordenador iniciará entonces la fase de combate.

Durante la fase de combate el ordenador indicará qué unidades aliadas están en condiciones de enfrentarse con el enemigo e informará acerca de la potencia relativa de las unidades. Sobre la base de esta información, el jugador puede aceptar o rechazar las sugerencias para el combate a medida que éstas se le van ofreciendo. Una vez llevado a cabo el combate y después de calculados y visualizados todos sus efectos, comienza la vez del segundo jugador.

Cortesía de Soft

Eastern Front (Frente del Este)
En este juego el jugador asume el rol del ejército alemán intentando alcanzar Moscú en 1941, mientras que el ordenador actúa defendiendo las fuerzas rusas. Escrito por Chris Crawford y distribuido por Atari, incorpora configuraciones nuevas, como *fine scrolling*. Una de las configuraciones más atractivas de *Eastern Front*, desde el punto de vista visual, es la forma en que se modifica el mapa a medida que transcurre el año. En otoño los árboles pierden las hojas y luego, en invierno, los ríos se hielan y el suelo se cubre de nieve. En este juego, llegar a Moscú se considera algo muy difícil

Ian McKinnell

Para muchas personas la fascinación que ejercen los juegos estratégicos nace del hecho de que no existe ninguna solución "correcta" a los problemas que plantea el juego. El placer del jugador deriva de la superación de los problemas físicos y logísticos del terreno en el que está operando, así como del reto intelectual que supone utilizar los recursos disponibles para derrotar al enemigo. Naturalmente, a todos los estrategas les gustaría ganar con los esquemas más atrevidos y las trampas más cuidadosamente urdidas; pero, sobre todo, ¡lo que les interesa es ganar!

Herramientas

Los equipos de herramientas son paquetes de software que mejoran una versión limitada de BASIC y ofrecen al programador utensilios para corregir errores

Los primeros ordenadores personales, como el Apple II y el Commodore PET, tenían capacidades limitadas y se diseñaron fundamentalmente para manipular números y texto. Del BASIC con que se dotaba a estas máquinas sólo se exigía que proporcionara órdenes y rutinas con estos fines. Como resultado de ello, se escribieron muchos programas de "utilidades" o de "equipos de herramientas", por lo general en código de lenguaje máquina, que operaban desde fuera del área de programación en BASIC. Éstos proporcionaban medios auxiliares para programación en forma de órdenes directas adicionales que podían ser de ayuda en la construcción y depuración de programas.

Desde entonces los ingenieros han creado una multitud de capacidades para gráficos y sonido, como consecuencia del creciente interés en los juegos recreativos para ordenadores personales. Cada nuevo modelo introduce configuraciones más amplias que enseguida se incorporan al software escrito profesionalmente. Sin embargo, salvo una o dos excepciones, el BASIC incorporado añade pocas mejoras, cuando no ninguna, respecto a las primeras versiones. Por ende, el usuario ha de elaborar rutinas, a menudo empleando órdenes PEEK y POKE repetidas, para incorporar estas nuevas configuraciones a la gama de órdenes disponibles. Consecuencia de todo esto es que ahora existen muchas utilidades, equipos de herramientas y ampliaciones para la mayoría de las máquinas populares. Se ha conseguido con ello un acceso más sencillo a las facilidades existentes (p. ej., a los editores de sprites o de sonido), una ampliación del software (p. ej., creadores de sprites) o simplemente ayudas para la programación en BASIC.

Cajas de herramientas
He aquí algunos de los equipos de herramientas y paquetes de ampliación del BASIC disponibles para algunos de los ordenadores personales más populares. Los paquetes que crean una facilidad de sprites en los ordenadores que carecen de esta configuración están adquiriendo una creciente popularidad

Equipos de herramientas	
SUPER TOOL KIT	De Nectarine, disponible para los Spectrum de 16 K y 48 K
SPECTRUM EXTENDED BASIC	De CP Software, para el Spectrum de 48 K
SPECTRUM KEYDEFINE	De Scientific Software, para el Spectrum de 48 K
PROGRAMMER'S AID	De Commodore, para el Vic-20
BUTI	De Audiogenic, para el Vic-20
TOOL BOX	De BBC Software, para los BBC Modelos A y B
SPRITE MAGIC	De Merlin Micro Systems, disponible para el Dragon 32
SPRITE GRAPHICS	De B Sides Software, para el Spectrum de 48 K
SPRITE MASTER	De Micro Dealer UK, para el BBC Modelo B

Las ampliaciones de esta clase se pueden localizar en RAM, en ROM interna o en cartucho de ROM. Es preferible una ampliación de ROM a una cargada en RAM, ya que no ocupa la memoria del usuario y está protegida contra un borrado por descuido. Por lo general, un programa escrito con la ayuda de un equipo de herramientas sólo funcionará en otro ordenador que esté similarmente equipado. No obstante, existen utilidades generadoras de programas autónomos, que se pueden ejecutar en una versión no ampliada del ordenador. Ésta es la base de la mayoría de los editores de gráficos y de sprites, así como de algunos editores de sonido.

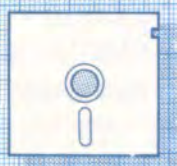
Entre las configuraciones útiles que cabe hallar en las ampliaciones al BASIC están las órdenes especiales para gráficos (como PAINT, DRAW, PLOT, CIRCLE, etc.) y las órdenes para sonido (como SOUND, PLAY, MUSIC, ENVELOPE, etc., o palabras que describen un efecto sonoro, como BANG o ZAP). Otras facilidades útiles son las órdenes para programación estructurada, como REPEAT...UNTIL e IF... THEN...ELSE. Las sentencias de este tipo le permiten al usuario escribir programas que vayan avanzando en secuencia lógica, y evitan el código desaliñado y difícil de comprender que resulta de la utilización indiscriminada de GOTO.

El BASIC de Simon

Actualmente, la ampliación más completa para el lenguaje BASIC es el BASIC de Simon, disponible para el Commodore 64 en forma de un cartucho de ROM. El BASIC Commodore estándar, incorporado en el modelo 64, está bastante anticuado en el sentido de que apenas si proporciona órdenes especializadas y ninguna para la programación estructurada. Aunque sí posee algunas configuraciones de hardware avanzadas, como un amplio sintetizador de sonido, gráficos en alta resolución y gráficos sprite, el control del BASIC sobre estas funciones se efectúa a través de PEEK y POKE. El Simon le proporciona al Commodore una considerable ampliación, en virtud de las siguientes facilidades extras:

Herramientas para máquinas

Estas órdenes son representativas de los servicios que debe ofrecer toda buena ampliación de BASIC



Incluirá varias órdenes de utilidades DISK para dar formato a discos nuevos, copiar y eliminar archivos individuales y hacer copias completas de un disco

DISK

Además de órdenes para dibujar líneas rectas en gráficos de alta resolución, puede ofrecer una facilidad para dibujar (DRAW) de punto a punto, que es una forma más sencilla de construir imágenes

DRAW

PAINT proporciona un medio de rellenar una zona definida en la pantalla con cualquier color elegido. En algunas máquinas el cursor se posiciona en el medio de la zona y enseguida el ordenador la rellena pintando desde el centro hacia afuera hasta que se topa con una línea sólida

PAINT

La finalidad de las órdenes para gráficos como CIRCLE (circunferencia) es obvia; algunas además permitirán dibujar circunferencias parciales (arcos) y elipses

CIRCLE

Aunque las funciones para sonido varían mucho de una máquina a otra, una orden MUSIC normalmente significa la capacidad de tocar una secuencia de notas que previamente se ha definido en una variable

MUSIC

El directorio de un disco lleva una lista actualizada de los nombres, tipos y dimensiones de todos los archivos. Una orden DIR le servirá dicho directorio en pantalla, sin perder el programa que está en la RAM

DIR

En muchos ordenadores, acceder a la palanca de mando es una tarea difícil que implica la utilización de PEEK y POKE. Las órdenes JOY (abreviatura de *Joystick*: palanca de mando) colocarán la posición de la palanca de mando directamente en variables BASIC

JOY

Kevin Jones

1) Un extenso acervo de recursos auxiliares para programación, incluyendo funciones que aseguran un control adicional sobre el listado y la depuración del programa y las ayudas de seguridad (protección del programa contra copias no autorizadas).

2) Órdenes adicionales para el manejo de variables y la manipulación de textos.

3) Operadores aritméticos extras y órdenes para conversión numérica.

4) Órdenes simplificadas para manipulación de discos.

5) Órdenes para gráficos en alta resolución que permiten mezclar texto con el trazado de puntos y dibujos. Se incluye también una facilidad para colorear croquis.

6) Órdenes para gráficos en baja resolución y manipulación de pantalla que pueden duplicar las áreas asignadas para gráficos y manipular con gran facilidad el área de pantalla. También permiten guardar en disco, cinta o impresora el contenido de cualquier pantalla.

7) Generador y editor de sprites fácil de usar.

8) Programación estructurada empleando órdenes para procedimientos como PROC, CALL y EXEC; más rutinas para bucles y comprobación de condiciones, como REPEAT...UNTIL, LOOP...EXIT, IF...END LOOP e IF...THEN...ELSE, que generalmente eliminan la necesidad de sentencias GOTO y GOSUB.

9) Rutinas para la creación de sonidos que permiten acceder a la gama completa de facilidades de sonido del 64 utilizando órdenes simples para dar forma a sonidos o tocarlos.

10) Órdenes simples para lápiz óptico, palancas de mando y mandos de raqueta.

Es muy poco frecuente que una ampliación incluya una gama de rutinas adicionales tan completa. La mayoría de los paquetes proporcionan utilidades y órdenes para un área específica de la programación. Por ejemplo, el cartucho Super Expander de Commodore, para el Vic-20, sólo suministra una gama simple de órdenes para música y gráficos en alta resolución. Las ampliaciones más populares son las ayudas para la construcción de programas. Éstas generalmente proporcionan órdenes de entrada de teclas únicas y diversas rutinas automáticas que simplifican la numeración de sentencias, la edición y la depuración en la modalidad directa.

Es un hecho normal que las utilidades y ampliaciones aporten más facilidad de acceso a las capacidades incorporadas del ordenador. Pero las rutinas que mejoran estas capacidades son más difíciles de encontrar, aunque están ya saliendo al mercado paquetes muy ingeniosos. Por ejemplo, las muchas ventajas que ofrecen los gráficos sprite para los juegos recreativos de acción rápida han sido la fuente de inspiración de algunas empresas más dinámicas para componer utilidades generadoras de sprites para aquellos ordenadores que no disponen de esta configuración.

Las utilidades, los equipos de herramientas y las ampliaciones para el BASIC que hemos reseñado aquí constituyen una pequeña fracción de las mejoras y las ayudas existentes. Aunque la tendencia actual de los fabricantes se orienta a proporcionar versiones de BASIC amplias y avanzadas, siempre existirá la necesidad de ayudas de software para contribuir a que la programación no sea una pesada carga sino un placer creativo.

La voz inconfundible

Los sistemas de reconocimiento de voz se están usando cada vez más en aplicaciones comerciales y de seguridad. No obstante, su poder se ve limitado por la capacidad de memoria del ordenador

Para que un ordenador sirva para algo ha de disponer de algún medio viable que permita alimentar en él las órdenes y la información. La interface que normalmente empleamos para comunicarnos con un ordenador personal es un teclado (aunque los "ratones" y las palancas de mando constituyen posibles alternativas). No obstante, utilizando un teclado nos encontramos con que nos vemos obligados a comunicarnos con el sistema mediante un lenguaje artificial. Las órdenes como CLS, DIRECTORY, RUN, LOAD y SAVE pueden tener algún significado para el sistema operativo, pero no son "naturales".

El sistema de comunicación natural entre los humanos es el habla, y no digitar los mensajes en teclados y contemplar las respuestas en aparatos de televisión. Si se pudiera lograr que un ordenador comprendiera las órdenes orales (aunque estuvieran expresadas en la misma forma que las consignadas a través de un teclado), éste sería mucho más fácil de utilizar, especialmente por aquellas personas con alguna incapacidad física. Para que un sistema de ordenador pueda "comprender" palabras habladas, primero debe procesar la entrada de sonido: las señales analógicas se deben analizar y convertir en una forma digital que pueda manejar el ordenador. Aunque parece ser algo muy sencillo de generar electrónicamente, la voz es una combinación de sonidos extraordinariamente compleja.

Los sueños del reconocimiento completo e instantáneo de voz (tal como tipifica el ordenador HAL en *2001: una odisea del espacio*) es muy poco probable que lleguen a ser realidad durante muchos años, si es que se llegan a conseguir. La máquina de escribir para entrada de voz está igualmente lejana; y, a pesar de todo, la tecnología tanto para esta máquina como para el ordenador "comprensivo" ya existe. Pero ninguna de las dos se puede conseguir por un precio reducido, porque existe una enorme dificultad en cuanto a crear sistemas de reconocimiento de voz: las palabras pueden sonar iguales pero tener significados diferentes, según el contexto en el que aparezcan. El potencial de procesamiento necesario para solventar este problema simplemente no se logra a un precio razonable.

A pesar de que los investigadores han creado sistemas que se aproximan a este objetivo, han descubierto que al aumentar el número de hablantes que el ordenador puede reconocer se reduce el número de palabras que se pueden reconocer por vez. Por lo general, un sistema de reconocimiento de hablantes múltiples permitirá reconocer entre 20 y 30 palabras al mismo tiempo, con un margen de acierto de un 85 o 90 % aproximadamente.

Las aplicaciones potenciales de los sistemas de reconocimiento de voz son considerables. La Oficina de Correos alemana utiliza uno de ellos para ayudar a clasificar la correspondencia; y en la actualidad existen muchas aplicaciones en el ámbito aeroespacial, tanto militar como civil, donde los pilotos, pese a utilizar manos y pies, se ven desbordados en su esfuerzo por controlar los aparatos. En todas estas situaciones el número de palabras que se pueden reconocer cada vez se limita a alrededor de 20. Sin embargo, esto no significa que el sistema global sea limitado. El usuario selecciona una palabra de entre las 20 de un menú, y cada orden reconocida produce un nuevo menú de palabras entre las que escoger. El ordenador sólo emprenderá alguna acción después de que consiga reconocer la

Las partes de la oración

Una de las técnicas para el reconocimiento de la voz consiste en digitalizar la señal y realizar un análisis de "reconocimiento de patrón". Un método más eficaz es usar el preprocesamiento en hardware, en virtud del cual una cantidad de circuitos miden la señal en términos de sonidos sonoros (p. ej., las vocales), fricativos (*s, f, t* inglesa, etc.) y breves períodos de silencio (p. ej., entre las sílabas). La salida producida por cada uno de estos dispositivos de filtro es una serie de unos o ceros, que el ordenador compara con una biblioteca de ejemplos almacenados, seleccionando la pareja más aproximada como la palabra que ha reconocido

Kevin Jones

secuencia completa. En el caso de la oficina de clasificación, el primer nivel de clasificación sería por país y, una vez seleccionado el país correcto, la siguiente clasificación sería por provincia, luego pueblo, etc. El envío sólo se remitiría hacia su destino en el nivel inferior, asegurando de esta manera la máxima fiabilidad de la operación.

Análisis de voz

El reconocimiento de voz se suele abordar de dos maneras. El método "rápido" consiste simplemente en alimentar la voz a través de un convertidor analógico-digital, y utilizar el potencial del ordenador para llevar a cabo todo el análisis. Lamentablemente, este procedimiento posee algunos inconvenientes, siendo el principal el tiempo que ocupa realizar el análisis. Los sistemas que emplean este método pueden tardar dos o tres segundos en reconocer la entrada. Para que el reconocimiento de voz sea verdaderamente útil, el ordenador debe "comprender" el habla con la misma rapidez que un ser humano, y el enfoque de procesamiento de números raramente lo consigue.

El otro método consiste en emplear el preprocesamiento. En vez de analizar la señal de voz matemáticamente, se puede hacer gran parte del trabajo mediante la electrónica estándar. Lo que se le proporciona entonces al ordenador es información acerca de la entrada hablada: de la frecuencia, contenido, tono, energía, etc. Las frecuencias se pueden medir filtrando la señal y detectando el nivel en cada banda de frecuencia, algo así como utilizar los controles de tono en un equipo de alta fidelidad para "producir" el bombo. Dado que todo este procesamiento electrónico se realiza al mismo tiempo que se alimenta a los circuitos la señal de habla original, el análisis es casi instantáneo. Efectuar una operación similar en los datos de un convertidor analógico-digital requeriría que varios ordenadores trabajaran con los números al mismo tiempo. El método de preprocesamiento está todavía en fase de investigación pero parece ser el que presenta mayores posibilidades de éxito.

Una vez que se ha extraído de la señal original (independientemente del método empleado) la información relativa a frecuencia de contenido, tono, energía, etc., se lleva a cabo el verdadero reconocimiento, comparando el conjunto de las cifras obtenidas con una cantidad de modelos almacenados en la memoria del ordenador. Estos modelos se crean "entrenando" al sistema de reconocimiento. Las palabras que se van a reconocer se le dictan de viva voz al sistema de a una por vez y la información resultante se almacena en una "biblioteca" digital de ejemplos. Entonces se vuelve a dictar el conjunto completo de palabras y el ordenador compara la entrada con el modelo que tiene de ella. Si coinciden, al primer juego de información se agrega otro, para formar una versión más completa del modelo.

Para reconocer una palabra hablada, el ordenador debe emparejar el patrón de información de la entrada con uno o varios de los modelos almacenados en la biblioteca habitual. En muchos casos se hallarán varias posibles parejas que forman parte de otras palabras que concuerdan con el patrón de entrada. Las dos primeras sílabas de "internacional", por ejemplo, son las mismas que las de "inter-

Ian McKinnell

Control del entorno
Las aplicaciones del reconocimiento de voz más recientes son de naturaleza educativa. Una de éstas es la denominada *entorno limitado*, en la que intervienen un ordenador, un brazo-robot y un número de objetos sencillos que el brazo pueda manipular. Hablando por un micrófono, el usuario puede indicarle al brazo que «COLOQUE EL HUEVO EN LA HUEVERA». El ordenador tendrá que interpretar las órdenes y buscar en su memoria las posiciones de los objetos

vento". Al final de la búsqueda, debería haber una palabra que resaltara como la pareja más perfecta en relación a todas las otras posibilidades, y es ésta la que el ordenador interpretará como la entrada.

Con toda seguridad, las facilidades para reconocimiento de voz hallarán en el futuro muchas aplicaciones, pero es probable que se utilicen más directamente en paquetes complejos de software, como bases de datos, donde las órdenes se seleccionan de un menú que aparece en la pantalla. Este tipo de aplicación eliminará el mayor obstáculo con el que se encuentran quienes no son expertos: el teclado. Los sistemas de videotex han reducido el dispositivo de entrada a un sencillo teclado numérico adicional, pero éste limita sustancialmente el grado de interacción que puede conseguir el usuario. Una interface activada mediante la voz que pudiera reconocer un juego estándar de órdenes de interrogación para bases de datos, así como los símbolos numéricos y las letras del alfabeto, proporcionaría una eficaz configuración.

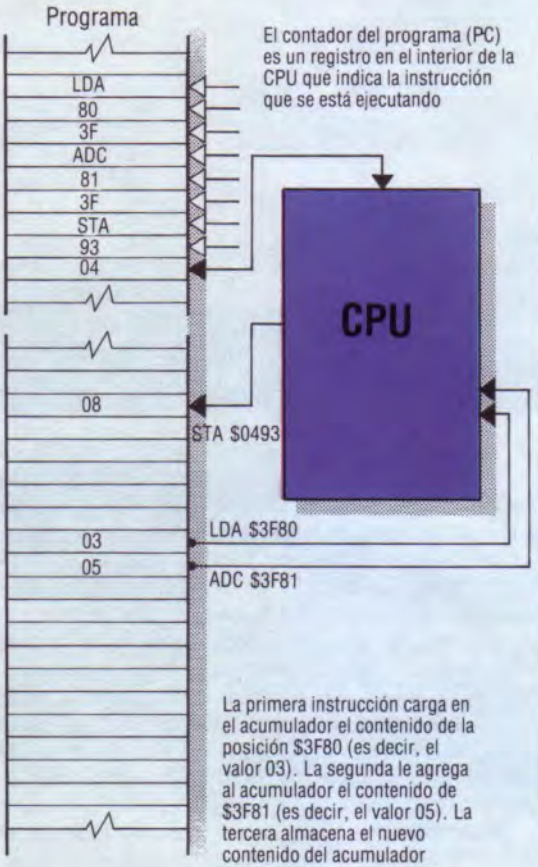
Existen en la actualidad unidades de reconocimiento disponibles a nivel comercial que se pueden enchufar en ordenadores personales, pero se trata de dispositivos muy poco sofisticados. Sistemas como Big Ears y Speech Lab, de Heuristic Inc., emplean gran potencial de procesamiento para reconocer apenas unas pocas palabras pronunciadas por una persona. Lo que se necesita para que el reconocimiento de voz sea verdaderamente útil es una capacidad para reconocer las palabras pronunciadas por *cualquier* persona, independientemente del dialecto o el acento. En esta etapa, el factor restrictivo es la cantidad de memoria disponible para retener los modelos. Una posibilidad interesante consiste en utilizar un videodisco para retener un juego estándar de modelos: éste apenas si emplearía memoria interna y prácticamente no se notaría ninguna reducción de velocidad.

Lenguaje máquina

El aprendizaje del código de lenguaje máquina ofrece más dificultades que el del BASIC, pero proporciona un gran aumento de velocidad y eficacia

Paso a paso

El programa en código de lenguaje máquina se almacena en una zona de la memoria, si bien los datos sobre los que operará pueden estar en cualquier otro lugar de la memoria. Observe que los operandos (p. ej., $3F80) se almacenan como dos bytes, con el byte inferior ($80) antes del byte superior ($3F)



Todos los programas en código de lenguaje máquina se construyen a partir de operaciones simples que transfieren bytes de memoria a los registros internos de la CPU, los procesan y los devuelven a una posición de la memoria. Este diagrama muestra el programa que se requiere para sumar los contenidos de dos posiciones y almacenar el resultado en una tercera

La primera instrucción carga en el acumulador el contenido de la posición $3F80 (es decir, el valor 03). La segunda le agrega al acumulador el contenido de $3F81 (es decir, el valor 05). La tercera almacena el nuevo contenido del acumulador (ahora 08) en la posición de memoria $0493

Kevin Jones

Hasta este momento en *Mi Computer* la programación se ha centrado en el lenguaje BASIC, porque es a la vez versátil y fácil de utilizar. Sin embargo, a medida que su experiencia aumente y que los proyectos de programación que emprenda se vayan haciendo más ambiciosos, no tardará en descubrir las limitaciones de este lenguaje.

Por el contrario, la programación en código de lenguaje máquina supone muy pocas restricciones en cuanto a lo que usted puede hacer y, en comparación con el BASIC, da la sensación de una velocidad casi infinita. Así y todo, muy pocos usuarios de ordenadores personales se atreven a dar el salto del BASIC al lenguaje máquina, en parte porque la utilización del código de lenguaje máquina es un proceso de programación que exige un trabajo mucho más minucioso, y también porque conceptualmente se diferencia mucho del BASIC y de todos los otros lenguajes de alto nivel. A pesar de ello, vale la pena tener idea del código de lenguaje máquina; y en este capítulo, al que seguirá otro más, analizaremos los procedimientos fundamentales.

Como ya explicamos, el lenguaje máquina es el único que comprende el microprocesador (la CPU), auténtico cerebro del ordenador. Este microprocesador sólo sabe realizar funciones muy sencillas (p. ej., puede sumar dos dígitos de un número, pero no sabe multiplicarlos). Sin embargo, realiza estas funciones a velocidades asombrosas. Cada operación de un microprocesador se caracteriza por el número de *ciclos de reloj* que necesita. Si la CPU de su ordenador funciona a 1 MHz, entonces un ciclo de reloj equivale a un microsegundo, y una operación cuya ejecución requiera cuatro "ciclos de reloj" se realizará en cuatro millonésimas de segundo.

Por consiguiente, todo programa escrito en lenguaje máquina constará de un gran número de instrucciones y todas las funciones se han de construir "a mano" a partir de operaciones simples. Toda programación en código de lenguaje máquina consiste en la manipulación de bits y bytes individuales de la memoria, utilizando funciones lógicas simples como AND, OR y NOT, y una elemental aritmética en sistema binario.

Ésta es una de las razones por las cuales escribir en código de lenguaje máquina es una tarea tan lenta; la otra es que el programador tiene que saber cuál es el lugar de la memoria en que se conserva todo. En BASIC, cada vez que se encuentra una sentencia como LET A = 5, es tarea del intérprete encontrar un espacio de la memoria donde almacenar esa variable. Y cada vez que más adelante en el programa se haga referencia a A, recordará dónde buscar los datos necesarios. Cuando se comienza a programar en código de lenguaje máquina, uno descubre que debe especificar una dirección (una posición de memoria) para cada dato que necesite almacenar, y que ha de asegurar que encima no se escriban por descuido otros datos diferentes.

Analicemos en qué consiste el lenguaje máquina. (A propósito, todos nuestros ejemplos se refieren a unidades CPU de ocho bits, como el Z80 y el 6502; los dispositivos de 16 bits trabajan de modo similar pero a cada operación procesan el doble de bits.) El microprocesador está conectado a la memoria del ordenador mediante dos buses (un bus no es más que un grupo de cables o líneas): el bus de direcciones y el bus de datos (véase p. 144). También existe el llamado bus de control, pero éste sólo proporciona a la CPU señales de sincronización y el programador no lo utiliza.

El bus de direcciones tiene una anchura de 16 bits y, colocando en este bus un patrón de bits, la CPU puede seleccionar cualquiera de los 65 536 bytes de su "mapa de memoria" (véase p. 329). En un ordenador personal típico, algunas de estas posiciones de memoria son RAM, otras ROM, otras chips especiales de entrada-salida y hay algunas más sin emplear. Si la CPU desea leer una posición de memoria (una de las líneas del bus de control indica si se ha de efectuar una lectura o una escritura), entonces el byte seleccionado colocará su contenido en el bus de datos, en forma de un patrón de ocho bits. Del mismo modo, la CPU puede escribir un patrón de ocho bits en cualquier posición selec-

cionada. La CPU no sabe qué partes de la memoria son ROM y RAM, y dar con las direcciones correctas es otro de los aspectos cruciales que se dejan al programador.

Dentro del microprocesador existen quizá media docena de "registros", que son como posiciones individuales de memoria y que se utilizan para almacenar resultados temporales y realizar las funciones de lógica y de aritmética binaria. La mayoría de estos registros equivalen a un byte de memoria, aunque algunos son de 16 bits de anchura. Uno de estos últimos se denomina registro *contador de programa* (PC) y éste contiene la dirección en la memoria de la instrucción en código de lenguaje máquina que se está realizando en esos momentos. Podemos imaginárnoslo como la línea en un programa escrito en BASIC.

Otro registro importante (pero en esta ocasión de sólo ocho bits de anchura) es el *acumulador*. Como su nombre indica, este registro puede acumular totales (es decir, se le pueden sumar o restar bytes) y en general es el único registro que puede efectuar toda clase de operaciones aritméticas. Por tanto, un programa muy sencillo en código de lenguaje máquina se podría especificar así:

1) Cargar en el acumulador el contenido de la posición de memoria $3F80. En código de lenguaje máquina las direcciones se suelen escribir en hexadecimal (véase p. 179). Los números hexadecimales se indican anteponiéndoles un signo especial, por lo general $.

2) Sumarle al acumulador el contenido de la posición de memoria $3F81, dejando abierta la posibilidad de que el resultado pueda ser mayor de lo que se puede almacenar en un solo byte, en cuyo caso también habrá un "bit de arrastre" (denominado en inglés *carry bit*).

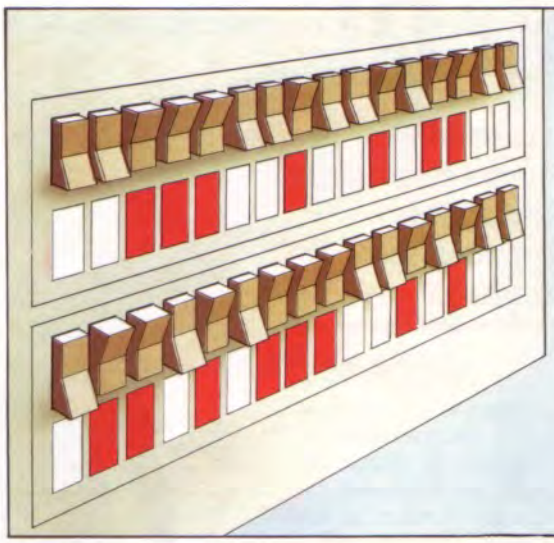
3) Almacenar el nuevo contenido del acumulador (es decir, el resultado) en la posición de memoria $0493.

Cada uno de estos puntos constituye una instrucción en código de lenguaje máquina, y normalmente el programa se escribirá de la manera siguiente:

LDA $3F80	(LoaD Accumulator) (cargar acumulador)
ADC $3F81	(ADd with Carry) (sumar con posible arrastre)
STA $0493	(STore Accumulator) (almacenar acumulador)

Las observaciones entre paréntesis, al igual que las sentencias REM en BASIC, no poseen ningún efecto. La primera entrada de cada línea se denomina *opcode* (código de la operación) e indica la naturaleza de la operación a realizar. La segunda columna contiene el "operando": los detalles, o el paradero, del dato sobre el cual se ha de operar. Un microprocesador normalmente admitirá varias docenas de posibles *opcodes* (es decir, puede realizar varias docenas de tipos de operaciones simples) y, cuando se le haya dado entrada en la máquina, cada *opcode* ocupará tan sólo un byte de la memoria.

Por consiguiente, un *opcode* se puede especificar como un número dentro de la escala 0-255 (o, mejor dicho, dentro de la escala hexadecimal entre $00 y $FF). Sin embargo, mientras se está desarrollando un programa, para que el listado resulte más legible se utilizan tres letras mnemotécnicas, como LDA, ADC y STA.



Luces intermitentes
La idea de los inmensos paneles de luces que suelen caracterizar los ordenadores en las películas proviene del "panel frontal" que poseen muchos miniordenadores. Este panel frontal era una línea de luces e interruptores que representaban los buses de direcciones y de datos de la CPU. Antes de que los teclados se conectaran mediante interfaces, se había de dar entrada a todos los programas en código de lenguaje máquina de esta forma, en sistema binario

Kevin Jones

Cada uno de los tres operandos mencionados se compone de un número hexadecimal comprendido en la escala $0000-$FFFF, y emplean hasta dos bytes del espacio de memoria para el programa. No obstante, algunos operandos sólo tienen un byte de longitud, y algunos *opcodes* no poseen ningún operando. El breve programa que hemos visto ocuparía, por tanto, un total de sólo nueve bytes, sin incluir las tres posiciones de memoria ($3F80, $3F81 y $0493) sobre las que operará el programa. Para este ejercicio trivial, el siguiente programa en BASIC alcanzaría exactamente los mismos objetivos, pero ocuparía alrededor de 50 bytes y efectuaría la operación cien veces más lentamente, en virtud de todo el tiempo que invertiría el intérprete en traducirlo:

```
10 A = PEEK (16256)
20 A = A + PEEK (16257)
30 POKE 1171,A
```

Nota: Quizá las posiciones utilizadas para este programa no sean adecuadas para su máquina.

En el próximo artículo analizaremos cómo se da entrada al código de lenguaje máquina en un ordenador personal y cómo se ejecuta.

Opcodes
He aquí algunos de los opcodes (códigos de operación o instrucciones) que puede ejecutar un microprocesador normal

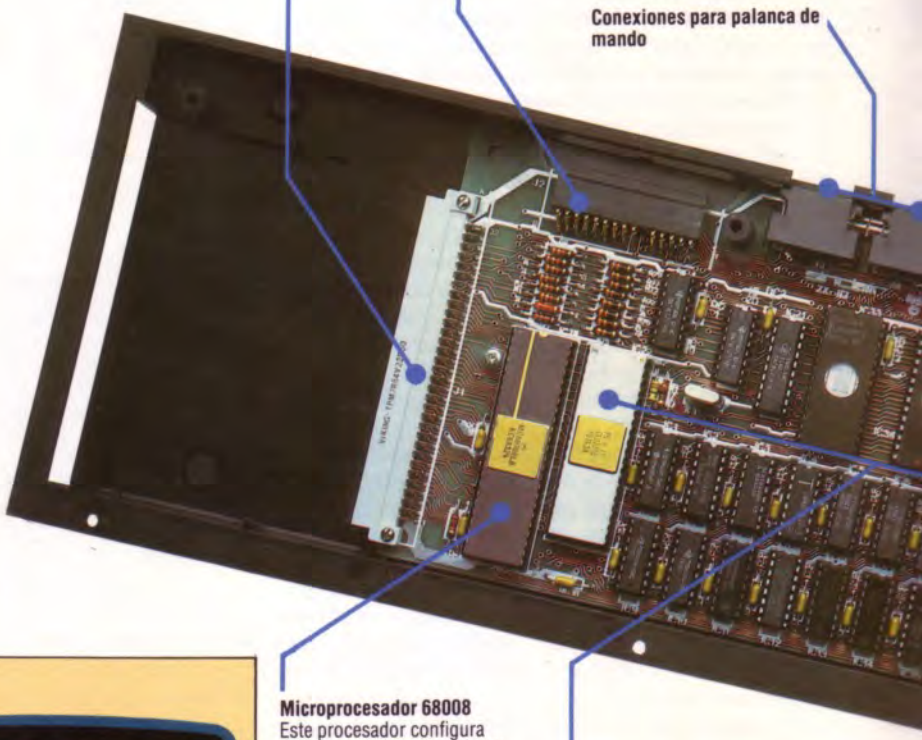
Sinclair QL

El Quantum Leap ofrece el microprocesador más avanzado de todos los ordenadores personales, con un potencial para medio megabyte

Todas las innovaciones de sir Clive Sinclair en el campo de los ordenadores personales han representado saltos cuantitativos tanto en términos de tecnología como de precio, sólo que su último microordenador es la primera de sus máquinas que declara esta verdad ya en su propio nombre: el Sinclair Quantum Leap (QL), que literalmente significa "salto cuantitativo". Por su precio bastante moderado, está pensado para un número creciente de usuarios que son o bien grandes entusiastas de los ordenadores o que pretenden alguna aplicación tanto de gestión como personal. Como tal, representa una competencia muy seria para máquinas como el Commodore 64 y el BBC Modelo B, aunque por su especificación técnica es a todas luces superior.

Resulta bastante evidente que el QL nació como un resumen de todos los componentes y configuraciones más novedosas en cuanto a ordenadores. Abriendo una brecha en la opción normal entre el Z80 o el 6502, la CPU es un miembro de la familia Motorola 68000, que actualmente es el microprocesador más exquisito que puede hallarse en un microordenador, y que utilizan máquinas como el Lisa de Apple (véase p. 261). No obstante, la CPU es un 68008, lo que significa que si bien sus registros internos son de 16 bits (y puede realizar muchas funciones a través de 32 bits completos), su bus de datos externo sólo tiene una anchura de ocho bits. Esto retardará la operación de la CPU muy ligeramente, porque la carga y el almacenamiento de los registros se habrá de hacer por mitades. Pero esto también significa que se mantiene reducido el costo de los chips de memoria y, a la hora de elegir los componentes, la economía suele ser una de las consideraciones básicas de Sinclair.

El QL viene con 128 Kbytes de RAM como estándar, pero con futuros accesorios se podrán ampliar a 512 Kbytes ("medio mega", como se suele decir). Esta vasta memoria es particularmente útil para aplicaciones de gestión empresarial, ya que reduce la frecuencia con la cual el programa se debe remitir al almacenamiento fuera de línea. Este al-



Interface para ampliación
Capaz para acoplar periféricos y hasta 0,5 Megabytes de RAM

Ranura para cartucho ROM
Aquí se pueden enchufar hasta 32 K de ROM adicional

Conexiones para palanca de mando

Microprocesador 68008
Este procesador configura registros internos de 16 y 32 bits, con un bus de datos externo de 8 bits

Chips a la medida
Es cada vez mayor el número de ordenadores nuevos que configuran un chip diseñado a medida. El QL posee dos, para manipular la visualización y diversas interfaces

El software del QL

QL Quill es un paquete para tratamiento de textos que visualiza el texto en la pantalla en el mismo formato en que se imprimirá

QL Abacus es una hoja electrónica en la que se pueden señalar las celdas por su nombre en lugar de mediante las coordenadas

QL Archive es un paquete para base de datos. El usuario puede diseñar trazados para informes, valiéndose de la ayuda de un editor de pantalla.

QL Easel, utilidad para gráficos y diagramas, maneja aspectos del diseño como la reducción y ampliación a escala

Ian McKinnell

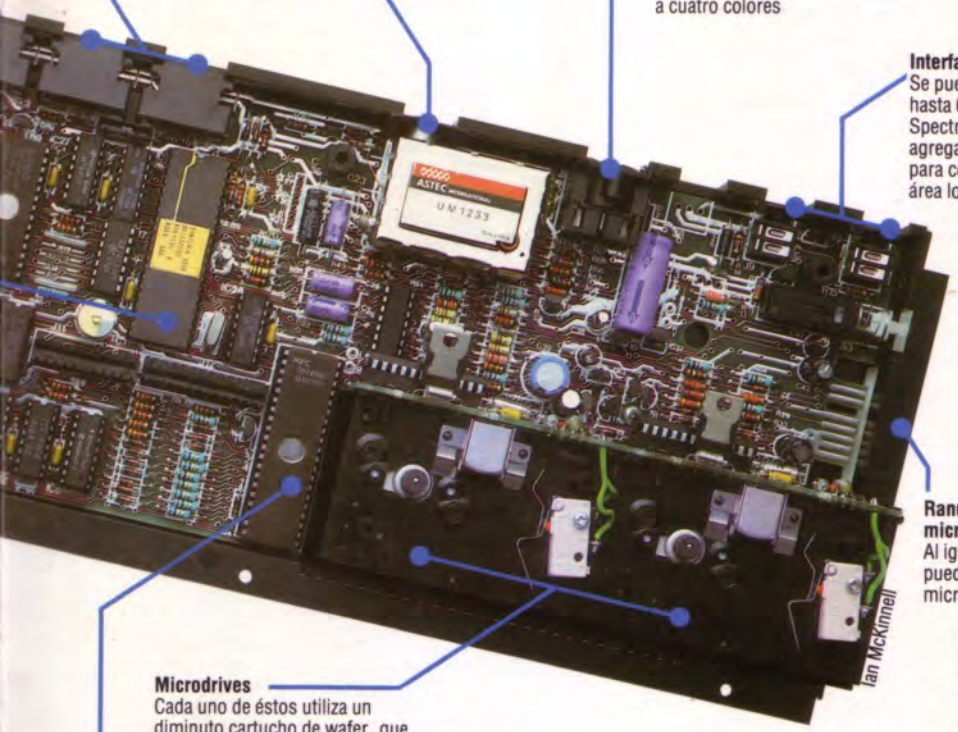
Teclado
El teclado se basa en una construcción de tipo membrana (que lo protege contra accidentes como derramamiento de café, salpicaduras, etc.), pero contiene 65 teclas de recorrido total y la "sensación táctil" es tan buena como la de algunas de las máquinas de gestión más caras. Hay cuatro teclas para control del cursor y cinco teclas de función programable. También incluye el símbolo de copyright

Puertas en serie
Incorpora dos puertas RS232, aptas para activar una impresora y un modem. La interface para impresora más común (Centronics) se debe adquirir como accesorio

Conector para TV
El QL funciona con un aparato de televisión, pero normalmente sólo visualizará 40 o 60 columnas, mientras que con un monitor el número de columnas es de 85

Conexión monitor
A diferencia del Spectrum, el QL puede activar directamente un monitor RGB, indispensable para aprovechar la máxima resolución de 512 × 256 pixels a cuatro colores

Interface para red
Se pueden conectar entre sí hasta 64 ordenadores QL y Spectrum (estos últimos agregando la interface 1) para conformar una red de área local



Ranura para ampliación de microdrive
Al igual que el Spectrum, el QL puede manejar hasta ocho microdrives

Microdrives
Cada uno de éstos utiliza un diminuto cartucho de wafer, que contiene un bucle continuo de cinta para almacenar hasta 100 K cada uno

Segundo microprocesador
Este Intel 8049 controla el teclado, el sonido y las puertas en serie, dejando libre al 68008 para la ejecución de los programas del usuario

SINCLAIR QL

DIMENSIONES
472 x 138 x 46 mm

CPU
Motorola 68008

VELOCIDAD DEL RELOJ
7,5 MHz

MEMORIA
128 K de RAM, ampliables a 512 K
32 K de ROM, ampliables a 64 K

VISUALIZACION EN VIDEO
25 líneas de 85 caracteres (con monitor), gráficos de alta resolución: 512 x 256 pixels (4 colores), 256 x 256 (8 colores)

INTERFACES
RS232 en serie (2), palancas de mando (2), microdrives, LAN, TV, monitor RGB

LENGUAJE SUMINISTRADO
BASIC

OTROS LENGUAJES DISPONIBLES
Existen planes para varios, entre los que destaca el lenguaje "C"

VIENE CON
Manual de instrucciones, cuatro programas de aplicaciones

DOCUMENTACION
El manual provisional es de un nivel elevado, viene en carpeta de anillas e incluye manuales para el software estándar

macenamiento se compone de dos microdrives incorporados en la carcasa, que ofrecen alrededor de 100 Kbytes cada una. Aunque esto hace que el QL sea un sistema de gestión autocontenido, los microdrives se deben considerar más bien como un punto débil en comparación con el procesador, que es notablemente eficaz. Localizar un dato en el microdrive cuesta unos 3,5 segundos, en contraste con el medio segundo que puede costar en la nueva generación de miniunidades de disco flexible.

Sinclair ha prometido producir una interface para una unidad de disco rígido (Winchester), pero no existe ningún plan para discos flexibles, aunque no cabe ninguna duda de que algunos fabricantes independientes los ofrecerán. Será una pena, pues sin disco el QL no podrá ejecutar el sistema operativo Unix, que se suele considerar como una de las principales razones para optar por una CPU Motorola 68000, llamada a suplir al CP/M como sistema operativo estándar para el software de gestión.

El QL viene con cuatro paquetes de gestión tipo, todos ellos elaborados por la empresa de software Psion. *Quill* es un procesador de textos; *Abacus*, un paquete de hoja electrónica; *Archive*, una base de datos, y *Easel*, un paquete para gráficos. Todos funcionan bajo el sistema operativo residente, que Sinclair ha apodado QDOS. La popularidad que probablemente conseguirá esta máquina implica que se desarrollará muchísimo software para ella, aunque a las firmas de software no les será fácil transferir al QL los paquetes ya existentes. Así y todo, se podría aducir que si Sinclair adoptara los estándares de la industria, sus productos no ocuparían el primer lugar en el mercado, como ahora.

El BASIC residente se ha mejorado respecto a la versión del Spectrum y, como si el propio nombre Quantum Leap no fuera ya de por sí suficientemente descriptivo, Sinclair lo ha denominado SuperBASIC. Incluye facilidades para manipular procedimientos (favoreciendo, por tanto, la programación estructurada) y para acceder al sistema operativo desde un programa en BASIC. Tanto éste como el QDOS están contenidos en los 32 Kbytes de la ROM estándar.

El Sinclair QL es indudablemente una máquina impresionante y, tal vez lo más importante de todo, posee las suficientes posibilidades de ampliación como para protegerse contra la obsolescencia.

PROTOTIPO DEL TABLERO
En la fotografía vemos el tablero de circuito impreso de un QL prefabricado. Algunos extremos pueden variar en los modelos fabricados

Principios sonoros

Las funciones para sonido de los modelos Atari disponen de cuatro voces independientes

Las facilidades de sonido del Atari son buenas (tal como se desprende de muchos de los juegos en cartucho), aunque los medios para su control no son tan comprensibles. Dispone de cuatro osciladores de onda cuadrada independientes, cada uno de ellos con una escala de tres octavas. Además, la salida del oscilador se puede distorsionar de siete maneras para dar mayor variedad al sonido. A estas facilidades se puede acceder cómodamente desde el BASIC a través de la orden SOUND, pero ésta no hace un uso cabal de las configuraciones extras del chip para sonido POKEY de Atari, que puede modificar aún más el sonido producido con filtros "paso-altos" y modalidades especiales de operación. Por consiguiente, la gama completa de control de sonido sólo se puede explotar totalmente utilizando complicadas órdenes POKE o código de lenguaje máquina, que cae fuera de este nivel del curso. La salida se efectúa solamente a través del altavoz del televisor.

SOUND

Ésta es una orden muy sencilla, con el siguiente formato:

```
SOUND O,P,D,V

O = oscilador (0-3)
P = tono (0-255)
D = distorsión (1-15)
V = volumen (1-15)
```

Cada orden SOUND puede seleccionar sólo un oscilador, de modo que es imposible comenzar más de un oscilador a la vez. Esto no es un gran problema, pero si se programa música utilizando todos los osciladores para armonías de cuatro partes, la demora es perceptible.

El tono se calcula de forma un tanto extraña y, en consecuencia, algunas frecuencias son inexactas. La frecuencia disminuye a medida que aumenta el número de tono, dando una escala efectiva desde *do* a 29 (1046,5 Hz) hasta *do* a 243 (130,81 Hz). La lista da los números de tono para algunas notas musicales. En el manual de referencias del BASIC de Atari se ofrece la lista completa.

Octava-1	Octava-3
(Centr.) do – 121	do – 29
si – 128	si – 31
la – 144	la – 35
sol – 162	sol – 40
fa – 182	fa – 45
mi – 193	mi – 47
re – 217	re – 53
do – 243	do – 60

El parámetro de distorsión "P" equivale al canal de ruido que poseen la mayoría de los ordenadores, pero es muchísimo más versátil. Cada número par hace que se mezcle con la entrada estándar del osci-

Redondeos

Un somero análisis de los gráficos del Oric revela su semejanza con el Spectrum

El ordenador personal Oric-1 salió al mercado a mediados de 1983 y es obvio que se diseñó para que compitiera con el ZX Spectrum de Sinclair. El Oric ofrece cuatro modalidades de visualización. Sin embargo, sólo una modalidad permite la utilización de gráficos de alta resolución. Hay ocho colores disponibles; los colores de fondo y primer plano se establecen respectivamente mediante las órdenes INK y PAPER. El BASIC del Oric posee varias órdenes especiales de alta resolución para ayudar al programador de gráficos.

La pantalla se compone de 28 líneas, cada una de las cuales contiene 40 espacios de caracteres. Los caracteres del Oric no se diseñan utilizando la cuadrícula normal de ocho pixels por ocho, sino que se construyen sobre una de ocho por seis. En la modalidad de alta resolución, la resolución de la pantalla es de 240 × 200 pixels, y se reservan las tres líneas inferiores para información como mensajes de error, etc. No existe ninguna orden similar a PAINT, pero con poco esfuerzo se puede conseguir dicha función utilizando la orden FILL. Al igual que en el Spectrum (véase p. 392), se pueden mezclar gráficos de alta resolución con texto en la misma pantalla, pero el Oric permite colorear individualmente cada una de las líneas del cuadrado de un carácter, mientras que el Spectrum únicamente admite un color dentro del cuadrado de un carácter determinado.

Analicemos ahora con mayor detalle las modalidades en baja resolución que ofrece el Oric-1. El Oric posee tres modalidades en baja resolución: TEXT, LORES0 y LORES1. La única diferencia entre LORES0 y LORES1 es que utilizan juegos de caracteres diferentes. En la modalidad TEXT, las letras se

lador un arreglo diferente de impulsos al azar. Nótese que 10 da una señal libre de distorsión, y no 0, como cabría esperar. Un experimentado uso de sonidos distorsionados puede proporcionar timbres interesantes, y es particularmente útil para efectos especiales.

El volumen "V" se puede establecer entre 1 y 15, y un nivel intermedio razonable sería 7 u 8. Observe que no hay un modo adecuado de sincronizar la duración de las notas o los silencios entre ellas. En estas circunstancias el método normal consiste en utilizar bucles FOR...TO...NEXT sincronizados con sumo cuidado.

Para ilustrar el empleo de SOUND, las siguientes órdenes darán un *sol* no distorsionado en la octava 3 por el oscilador 1 a un volumen 8 para 50 pasos de bucles FOR...TO...NEXT:

```
10 SOUND 1,40,10,8
20 FOR N = 1TO50:NEXT N
30 END
```

El END de la línea 30 apaga todos los osciladores. Por otra parte, una nueva orden SOUND para el mismo oscilador interrumpe la nota antigua y da inmediatamente la nota nueva. He aquí un programa para tocar una melodía sencilla:

```
10 REM *DIXIE*
20 FOR I = 1TO7
30 READ N:REM *NOTA*
40 SOUND 3,N,10,7:REM *TOCAR NOTA*
50 FOR P = 1TO400:NEXT P:REM *PAUSA*
60 NEXT I
70 DATA 217,162,128,144:REM *RE SOL SI LA*
80 DATA 162,193,162:REM *SOL MI SOL*
90 END
```

A las capacidades de sonido del chip POKEY del Atari podemos llegar en BASIC colocando (POKE) números en las posiciones de memoria desde la 53760 a la 53763. Con este método, las rutinas de sonido se ejecutan con mayor rapidez y se pueden iniciar todos los osciladores a una. La información necesaria para conseguir esto, junto con algunas ampliaciones de conocimientos, incluyendo técnicas más atrevidas en código de lenguaje máquina, las encontrará en *De Re Atari*, que ofrece el Atari Program Exchange (APX), así como en el excelente *Atari Sound and Graphics*, publicado por John Wiley & Son.

pueden posicionar horizontalmente mediante la orden TAB. Sin embargo, en las otras dos modalidades LORES esta posibilidad se mejora pues permite especificar las posiciones verticales y las horizontales, utilizando la orden PLOTx,y,A$ donde *x* e *y* son las coordenadas de una posición de carácter determinada y A$ es la palabra o la frase a imprimir. El breve programa que sigue ilustra cómo se usan para escribir un nombre verticalmente:

```
10 REM LETRAS EN VERTICAL
20 CLS
30 LORES0
40 A$ = "RAMON"
50 FOR X = 1TO5
60 B$ = MID$(A$,X,1)
70 PLOT16,11 + X,B$
80 NEXT X
90 END
```

La orden HIRES le permite al usuario dar entrada a la modalidad en alta resolución del Oric. En la modalidad HIRES la pantalla tiene su punto de origen en la esquina superior izquierda.

En el BASIC del Oric existen varias órdenes que atañen a los gráficos: CURSETx,y,k posiciona el cursor sobre el punto dado por las coordenadas (x,y). El tercer número, "k", permite emplear con CURSET distintas funciones.

Valor de k	Función
0	traza el pixel en el color del fondo
1	traza el pixel en el color del primer plano
2	invierte los colores
3	no hace nada

CURMOVx,y,k es similar a CURSET, con la excepción de que el movimiento del cursor está en función de su posición anterior. DRAWx,y,k dibuja una línea recta desde la posición de partida del cursor hasta un punto "x" unidades a través e "y" unidades hacia arriba. CIRCLEr,k es una orden que traza en la pantalla un círculo de radio "r". PATTERN es una orden interesante y poco usual: trocea los círculos y las líneas dibujados en una serie de puntos o rayas. El patrón exacto se define mediante el número "n", cuya escala va de 0 a 255. El Oric toma este número y se vale del patrón de bits de su equivalente binario para producir un patrón repetitivo de puntos, rayas o espacios. He aquí dos ejemplos que ilustran su utilización:

Valor de n	Equivalente binario	Patrón producido
170	10101010	- - - - - - - -
15	00001111	— —

Por último, está la orden FILLa,b,n. Cada fila en los espacios de caracteres de la pantalla del Oric posee un número relacionado con ella que alude a los colores del fondo y del primer plano, al carácter presente y si el carácter está intermitente o no. Se dice que este número es el *atributo* de esa fila. FILLa,b,n rellena las celdas del carácter "b" por "a" filas con los atributos representados por el número "n".

```
10 REM CONO
20 HIRES
30 CURSET120,0,3
40 PAPER3:INK4
50 FOR R = 1TO65
60 PATTERN 200-R
70 CURMOV0,2,3
80 CIRCLE R,1
90 NEXT R
100 END
```

PATTERN de un cono
Este programa demuestra alguna de las capacidades del Oric-1 en alta resolución. Se dibuja una forma cónica empleando un conjunto de círculos de radio creciente. Observe, asimismo, la utilización de la orden PATTERN para fragmentar las circunferencias a medida que se van dibujando

Agdpyp w bcqagdpyp

La criptografía fue una de las primeras aplicaciones de los ordenadores. En la actualidad, elaborar y descifrar un código sencillo está al alcance de un programador de BASIC

Toda nuestra comunicación con los demás está codificada. Tanto el habla como el lenguaje escrito son inteligibles sólo si la persona que recibe el mensaje conoce el código del comunicante. Lo mismo sucede con nuestras conversaciones con los ordenadores. La mayoría de los ordenadorès personales se comunican por medio de una versión del BASIC para ser accesibles a la mayoría de la gente, pero nosotros sabemos que la propia máquina no emplea este lenguaje para realizar sus funciones: ella debe primero interpretar las sentencias en BASIC en una forma puramente numérica que después utiliza para establecer las secuencias de conmutación definidas en el programa y, de este modo, producir los resultados deseados. Los códigos de este tipo (lenguajes humanos y lenguajes de programación) son de fácil acceso en nuestra vida cotidiana. Con un poco de esfuerzo y voluntad, cualquiera puede aprender francés, alemán, BASIC o FORTRAN.

Compresión de datos

Los usuarios de ordenadores que necesitan almacenar grandes cantidades de archivos de texto están a la búsqueda constante de formas de comprimir los datos en aquellos archivos. Una forma de conseguirlo es la distintivación. De modo muy similar a la serie de microordenadores ZX de Sinclair que producen una palabra completa reservada en BASIC al digitar una sola tecla, un distintivo se puede sustituir por una palabra o una frase. Además, las técnicas de codificación también se utilizan para comprimir aún más los datos. Se considera que Compact, una utilidad Unix, puede comprimir archivos de textos en un 38 %, y Clip, que funciona bajo CP/M, consigue regularmente resultados aún mejores. Compactor, que se ejecuta en el Commodore 64, realiza la misma función para programas en BASIC mediante la eliminación de sentencias REM, espacios innecesarios, etc.

Pero existe otro tipo de codificación (o *criptografía*, para usar la palabra adecuada) que tiene por objetivo exactamente lo contrario de la comunicación: su finalidad consiste en evitar que lo comprendan todos, a excepción de un reducido grupo al cual está destinado el mensaje. Hasta la segunda mitad del siglo XX, la transmisión de información en una forma ininteligible para el público general era privativa de los gobiernos y de algún que otro asunto industrial importante. Pero más recientemente, la criptografía se ha convertido en algo cotidiano.

Las claves y los códigos oscilan desde los muy simples (la suma o la resta de un valor determinado a cada byte, o la sustitución según algún formato de un carácter por otro cada vez que éste aparece) hasta las claves en extremo complejas por las que se encaminan los más recientes avances de la teoría de los números. Estas claves no contienen ningún elemento de repetición y, por consiguiente, no son descifrables con los tradicionales métodos de decodificación por análisis de frecuencia.

Quizás la más sencilla de todas las técnicas significativas de criptografía sea la *clave del César* (que probablemente se utilizó por primera vez en la época del Imperio romano). Para descifrar la clave del César sólo se necesita el mensaje y un conocimiento de la clave, de modo que no hay que consultar voluminosos libros de códigos ni documento alguno, ni se necesitan unas máquinas especiales. He aquí un breve mensaje escrito en clave del César:

AYJYZNXNQ W BPYENLCQ

Podemos aventurar algunas suposiciones acerca de estas palabras crípticas, a tenor de la forma en que se separan los grupos cifrados (aunque, por supuesto, ¡esto se podría haber hecho para crear mayor confusión!). Lo más obvio que destaca a primera vista es que el mensaje consta de tres palabras: la primera posee 9 letras, la segunda posee 1 y la última, 8. También es claro que la primera y la tercera palabras terminan con la misma letra. Aquí la última letra final común (Q) es, asimismo, una de las tres letras del mensaje que se repiten con mayor frecuencia (las otras dos son la Y y la N). Para el criptoanalista esta observación tiene un valor considerable (al menos, cuando sabe en qué idioma está trabajando). En castellano, las letras que se presentan con mayor frecuencia son la A y la E entre las vocales, y la S entre las consonantes; esta última, dado que con ella se forma el plural, suele hallarse al final de las palabras.

Con una muestra tan reducida como la que tenemos aquí (un total de sólo 18 letras, cifra que todo estadístico consideraría muy insuficiente para basar en ella cualquier análisis), es probable que nuestros resultados sean falibles. Pero, aun así, vamos a probar con la sustitución de frecuencia y veremos si los resultados obtenidos tienen algún significado. En primer lugar vamos a reemplazar la Q por la S, por ser aquélla la última letra de las dos palabras más largas.

AYJYZNXNs W BPYENLCs

El mensaje aún no tiene significado, pero existen otras pistas. ¿Qué hay de la relación entre la letra original y la letra por la cual la hemos sustituido? En el alfabeto, la Q está dos lugares antes que la S. ¿Qué sucedería si sometiéramos el resto del mensaje a la misma transformación? Dos lugares después de la Y (la otra de nuestras letras que aparece con mayor frecuencia) está la A (si consideramos el alfabeto como una cadena ininterrumpida), así que vamos a intentar agregar esta información:

AaJaZNXNs W BPaENLCs

En la primera palabra tenemos ahora dos vocales intercaladas, que en castellano es una construcción válida. Además, la letra final es una S, hecho que en esta lengua es de común ocurrencia, de modo que tal vez estemos en el camino correcto. Vamos a

someter el resto del mensaje a la misma transformación. Dos lugares tras de la A se encuentra la C; dos lugares después de la J está la N; aplicando la misnia correspondencia, la Z se convierte en B, la N en O y la X en Z... De esta manera llegamos a la solución: *Calabozos y dragones (Dungeons and dragons)*, un atractivo juego de aventuras.

La clave del César es, pues, un código de sustitución que se basa en "deslizar" el alfabeto hacia atrás o hacia adelante según un número que es secreto y que nos dará el nuevo valor de cada carácter. Sin utilizar las frecuencias, el mensaje puede simplemente construirse según una serie clave de

La clave del César

Este programa (escrito en BASIC Commodore) codificará textos en clave del César utilizando una serie de claves múltiples de cinco elementos. El mensaje aparece en texto llano mientras se le va dando entrada, y cuando se pulsa RETURN se imprime la versión cifrada. Al mensaje se le debe dar entrada sin espacios ni signos de puntuación

```
10 INPUT "DE ENTRADA A UNA CLAVE DE CINCO
   CIFRAS";K$
20 INPUT "DE ENTRADA AL MENSAJE";M$
30 FOR I = 1 TO LEN(M$)
40 LET J = I — INT (I/5)*5 + 1
50 REM***ROTA A TRAVES DE LA CLAVE
60 LET M = ASC(MID$(M$,1,1)) — VAL(MID$
   (K$,J,1))
70 IF M < 65 THEN LET M = M + 26
80 PRINT CHR$(M);
90 NEXT I
```

Para el Spectrum, se debe reemplazar la línea 60 por:

```
60 LET M = CODE(M$(I)) — VAL(K$(J))
```

transformaciones: 24225, por ejemplo. En este caso la primera letra se desplazaría dos lugares, la segunda cuatro, la tercera dos y así sucesivamente. Cuando se llega al último número de la serie clave la volvemos a iniciar. Con esta serie clave, el mensaje de muestra *Calabozos y dragones* quedaría:

AWJYWNVNQ T BÑYEKLAQ

En este caso, el análisis de frecuencia sería inútil, pues no hay uniformidad en la sustitución: una letra tendrá distintas sustitutas según su posición en el mensaje global. Otra clave sencilla autocontenida da el mismo mensaje en estos términos:

CBSDOAAOO RGNSLZYAE

Si observamos atentamente, podemos ver que esta serie de caracteres es en realidad un anagrama de *Calabozos y dragones*, completo y con los dos espacios entre las palabras. Aquí simplemente estamos tratando de determinar el algoritmo criptográfico, a partir de ejemplos tanto de texto llano como de texto cifrado, un procedimiento sorprendentemente común. Si la clave ha de ser comprensible al destinatario del mensaje, entonces la combinación de las letras debe ser predecible de alguna forma. Esta clave particular, conocida como *bar fence*, también requiere que el decodificador la conozca; en este caso es 3. Vamos a tomar los cinco primeros caracteres y escribirlos con tres espacios de por medio:

C***B***S * D***O

¿Reconoce algo? Entonces probemos con esto: escriba el mensaje de texto llano en tres líneas, yendo hacia arriba y hacia abajo entre ellas, de la siguiente manera:

```
C   B   S   D   O
 A A O O * * R G N S
  L   Z   Y   A   E
```

Los asteriscos representan los espacios entre las palabras; el método criptográfico es llano.

Los ejemplos que hemos citado hasta el momento han sido todos cifras, definidas como un método de escritura secreta que utiliza la sustitución o transformación de letras de acuerdo a una clave. Los códigos son algo diferentes, en el sentido de que tienden a reemplazar bloques enteros por otros bloques, normalmente más pequeños (permitiendo, de este modo, al mismo tiempo, comprimir los datos). Su inconveniente es que exigen que ambas partes posean un libro de código para que se puedan comunicar mensajes. Un ejemplo de esta técnica emplea una novela, un periódico u otro texto que se pueda conseguir con facilidad, e indica las palabras del mensaje dando el número de secuencia en que se producen. Un texto como:

> "Maigret leyó con atención el aviso: 'Por favor, al objeto de favorecer el tráfico, permanezca al volante, no baje del coche'. Sonrió. Ahora comprendía por qué en su momento el cadáver de Smith no fue identificado."

podría ser la clave para el código 4,6,7,17,2. Quizá usted pueda descifrar el alarmante mensaje...

Un ordenador de cualquier clase puede ser de enorme valor al intentar cifrar o descifrar mensajes criptográficos. Un requisito primordial de la clave del César, por ejemplo, es la capacidad de desplazamiento a través de una serie alfanumérica, sumando o restando una variable al valor ASCII de cada carácter, que se puede entonces imprimir. Dicha constante ha de ser susceptible de modificarse cuando se ejecuta el programa, y debe hacer que el alfabeto se "enrolle" (es decir, una A, cuando la clave fuera uno, debería dar Z). Así que:

AOPK AO PKZK WIECK

Criptoanálisis
Una de las primeras utilizaciones del ordenador fue la de romper los códigos de sustitución de claves múltiples, muy complicados, que empleaban ambos bandos durante la segunda guerra mundial. Los alemanes habían preparado una máquina, denominada ENIGMA, que generaba sus propias claves. Los criptogramas resultantes, sumamente complicados, obligaban a los aliados a dedicar muchísimos esfuerzos a interpretarlos. El grupo del Colossus, que trabajaba en Bletchley Park y del que Alan Turing era miembro destacado, obtuvo finalmente el éxito

Cortesía Science Museum

Cuestión de estilo

Conocidas las reglas fundamentales del BASIC, analicemos importantes aspectos del estilo de programación y algunas órdenes nuevas para perfeccionar la técnica de programación

El programa que hemos venido construyendo en los últimos capítulos del curso para la agenda de direcciones computerizada recurre a muchas de las configuraciones más importantes del lenguaje BASIC, pero no a todas. En los próximos capítulos veremos lo que el BASIC le ofrecerá si es que desea convertirse en un programador de nivel avanzado.

Programas en lenguaje máquina

La mayoría de las versiones de BASIC permiten incluir como parte del programa rutinas escritas en lenguaje máquina. En líneas generales, existen dos maneras de hacerlo. La más sencilla consiste en utilizar PEEK y POKE. La sentencia PEEK se emplea para examinar direcciones de memoria específicas. Por ejemplo, LET X = PEEK(1000) obtendrá el valor almacenado en la posición de memoria 1000 y se lo asignará a la variable X. Al ejecutar PRINT X se imprimirá, entonces, el valor que estaba (y que sigue estando) en la posición 1000. He aquí un breve programa para "mirar" (PEEK) qué contienen 16 posiciones de memoria e imprimirlas en la pantalla:

```
10 INPUT "DEME DIRECCION INICIAL DE 'PEEK'";S
20 PRINT
30 FOR L = 1 TO 16
40 LET A = PEEK(S)
50 PRINT "LA POSICION ";S;" CONTIENE: ";A
60 LET S = S + 1
70 NEXT L
80 PRINT "PULSE BARRA ESPACIADORA PARA
   EXAMINAR LAS 16 POSICIONES SIGUIENTES"
90 PRINT "O RETURN PARA TERMINAR"
100 FOR I = 1 TO 1
110 LET C$ = INKEY$
120 IF C$ <> CHR$(13) AND C$<> "" THEN
    I = 0
130 NEXT I
140 IF C$ = CHR$(13) THEN GOTO 160
150 GOTO 30
160 END
```

El bucle desde la línea 100 a la 130 verifica la entrada desde el teclado y después va, o bien al final del programa, si la tecla pulsada fue un RETURN (13, en ASCII), o bien al principio, saltándose INPUT.

Si se desea, también es posible imprimir el carácter ASCII situado en una posición de memoria: basta usar PRINT CHR$(A). Pero cuidado con las sorpresas, porque los valores ASCII inferiores a 32 (que es el "espacio" en ASCII) no están definidos uniformemente. Todos los valores ASCII desde 0 a 31 representan caracteres no susceptibles de impresión o funciones especiales, como los movimientos del cursor. Los diversos fabricantes de ordenadores prácticamente sólo coinciden en que ASCII 13 es por lo general retorno del carro y que ASCII 7 hace sonar el altavoz interno o produce un pitido.

POKE es lo inverso de PEEK. Permite escribir cualquier valor entre 0 y 255 en cualquier posición de memoria RAM. Sin embargo, esta facilidad se debe emplear con extrema cautela, porque escribir en una parte de la memoria que ya esté siendo utilizada por el programa puede producir resultados inesperados y hasta ruinosos. Las rutinas escritas en código de lenguaje máquina se pueden "colocar" (POKE) en las direcciones apropiadas y cuando se ejecuta el programa se las puede invocar mediante la sentencia CALL. Cómo escribir programas en código de lenguaje máquina es un tema que está fuera de los límites de este curso. Baste decir que el código de lenguaje máquina se ejecuta muchísimo más rápido que las mejores versiones de BASIC. En aquellas situaciones en las que la velocidad de ejecución es esencial, o en las que se requiere gran precisión, el código de lenguaje máquina es, con mucho, la mejor alternativa.

Mover el cursor

Ahora muchos ordenadores personales permiten dirigirse hacia un punto de la pantalla de inmediato, pero aun cuando su máquina no lo admita, se puede mover el cursor a izquierda, derecha, arriba y abajo de la pantalla con relativa facilidad. Primero necesita usted saber qué códigos ASCII se utilizan para representar las teclas de control del cursor. El breve programa que ofrecemos a continuación, tras pulsar una tecla le informará acerca del valor ASCII correspondiente a dicha tecla:

```
1 REM HALLA LOS CODIGOS ASCII PARA LAS
  TECLAS DEL CURSOR
10 PRINT "PULSE UNA TECLA";
20 FOR I = 1 TO 1
30 LET K$ = INKEY$
40 IF K$ = "" THEN I = 0
50 NEXT I
60 PRINT ASC(K$)
70 GOTO 10
80 END
```

Esta rutina le permitirá también hallar el código para la tecla RETURN (por lo general, 13), ESC (comúnmente, 27) y la tecla para espacio (por lo general, 32), además de los códigos para las teclas de control del cursor. El ordenador Sord M23, en el cual se probaron todos los programas para este curso de programación BASIC, utiliza los valores 8 para cursor a la izquierda, 28 para cursor a la derecha, 29 para cursor arriba y 30 para cursor abajo. Es probable que su ordenador exhiba valores diferentes. Sustituyendo los valores que ha hallado

para los códigos de control del cursor de su ordenador en el programa anterior, pruebe con éste:

```
10 PRINT CHR$(12): REM USAR CLS O CODIGO
   ADECUADO
20 FOR L = 1 TO 39
30 PRINT "*";
40 NEXT L
50 FOR L = 1 TO 22
60 PRINT CHR$(8);:REM USAR CODIGO 'CURSOR
   IZQUIERDA'
70 NEXT L
80 FOR L = 1 TO 4
90 PRINT "@";
100 NEXT L
110 END
```

Con esto debería imprimirse en la pantalla una línea parecida a ésta:

********** @@@@**********

Las sentencias de la 20 a la 40 simplemente habrán impreso una línea de 39 estrellas, y las sentencias 50 a 70 "imprimieron" el "carácter" del cursor izquierda 22 veces, de modo que el cursor se desplazó 22 lugares hacia atrás a lo largo de la línea. De la 80 a la 100 imprimieron luego @ cuatro veces y entonces el programa terminó. Las técnicas de programación como ésta permiten que el programador desplace el cursor alrededor de la pantalla para imprimir nuevos caracteres en nuevas posiciones que podrían no conocerse hasta que se calcularan los valores en el programa. Esta técnica ofrece la ventaja de posibilitar la utilización de caracteres de pantalla normales para trazar gráficos sencillos, sin necesidad de recurrir a las configuraciones especiales para gráficos del ordenador (si posee alguna).

Para ver cómo emplear este tipo de control del cursor para producir gráficos como una salida de sus programas, pruebe con este programa:

```
10 PRINT "ESTE PROGRAMA IMPRIME UN GRAFICO
   DE BARRAS DE 3 VARIABLES"
20 INPUT "DE ENTRADA A LOS TRES
   VALORES";X,Y,Z
30 PRINT
40 FOR L = 1 TO 2
50 FOR A = 1 TO X
60 PRINT "*";
70 NEXT A
80 PRINT CHR$(13)
90 NEXT L
100 FOR L = 1 TO 2
110 FOR A = 1 TO Y
120 PRINT "+";
130 NEXT A
140 PRINT CHR$(13)
150 NEXT L
160 FOR L = 1 TO 2
170 FOR A = 1 TO Z
180 PRINT "#";
190 NEXT A
200 PRINT CHR$(13)
210 NEXT L
220 PRINT
230 END
```

El programa imprime un diagrama de barras horizontales de las tres variables. Las barras se imprimen de izquierda a derecha siguiendo el movimiento "natural" del cursor. Observe que en las sentencias 80, 140 y 200 se necesita una PRINT CHR$(13). Ello se debe a que los puntos y coma al final de las sentencias PRINT suprimen los retornos del carro (13 es el código ASCII para <CR>).

En torno a las variables

Hasta ahora hemos tratado las variables como si sólo fueran de dos tipos (numéricas y alfanuméricas). En realidad, el BASIC conoce diversos tipos de variables numéricas, y un buen programador siempre especificará el tipo correcto para economizar memoria y asegurar la validez.

Cuando se enuncia una variable en un lenguaje de programación, automáticamente se reserva cierta cantidad de memoria para almacenar dicha variable. Si el programa sabe que la variable será siempre un entero (p. ej., LET NO = VOTOS — (SI + NULOS) se necesita reservar menos memoria para la variable. Si tenemos una variable que puede asumir un número indefinido de valores distintos (p. ej., LET AREA = PI * RADIO * RADIO), la cantidad de espacio de memoria a destinar será mayor.

Mientras elaborábamos nuestra agenda de direcciones computerizada, nos familiarizamos con la regla convencional de especificar las variables alfanuméricas utilizando el signo $ después del nombre de la variable (p. ej., LET BUSCLV$ = MODCAM$(TAMA)). Se suponía que las variables que no terminaran en un signo "dólar" eran variables numéricas ordinarias. Sin embargo, se pueden utilizar acuerdos similares después de los nombres de las variables para especificar el tipo de variable numérica. Se da por sentado que un nombre de variable que carezca de especificador es una variable numérica real de precisión sencilla. Entre los signos que reconocen la mayoría de las versiones de BASIC se incluyen: % para especificar una variable de números enteros, ! para especificar una variable de precisión sencilla, y # para especificar una variable de precisión doble (es decir, la variable puede almacenar el doble de dígitos significativos). A continuación ofrecemos el fragmento de un programa hipotético que recurre a estos signos:

```
70 LET JUGADOR $ = "LUIS": REM VARIABLE
   ALFANUMERICA
80 LET PUNTUACION % = 0: REM VARIABLE DE
   ENTEROS
90 LET PI! = 3,1416:REM VARIABLE DE
   PRECISION SENCILLA
100 LET AREA # = PI*R*R: REM VARIABLE DE
    PRECISION DOBLE
110 LET INTENTO = 6: REM SE SUPONE QUE ES
    REAL DE PRECISION SENCILLA
```

Es necesario señalar que no todas las versiones de BASIC admiten todos estos tipos de variables. El Spectrum, por ejemplo, no posee variables de enteros. Los enteros se almacenan simplemente como números reales de precisión sencilla. Tampoco admite números de precisión doble. No obstante, en el BASIC Spectrum los números de precisión sencilla se calculan con nueve guarismos significativos, contra los sólo siete guarismos significativos del BASIC Microsoft. El BBC Micro admite variables del tipo entero y los reales de precisión sencilla se calculan en nueve guarismos significativos. El BASIC Microsoft admite variables de precisión doble con 16 lugares significativos.

Los ordenadores que aceptan variables de enteros normalmente destinan dos bytes para almacenar un número, que puede estar dentro del intervalo [−32 768 , 32 767]. Por lo general este intervalo es perfectamente adecuado para variables tales como puntuaciones, número de empleados, contadores de bucles FOR...NEXT y otras cantidades que sólo admiten enteros. Dado que para almacenar el número sólo se emplean dos bytes, la utilización de variables de enteros, si se dispone de ellas, supondrá un ahorro de memoria, si bien en muchas versiones de BASIC ello sólo es válido para las matrices de enteros y no para las variables individuales.

En el próximo capítulo de este curso de programación sopesaremos las ventajas y las desventajas del BASIC como lenguaje de programación.

Complementos al BASIC

INKEY$

Para el Lynx, en el primer programa sustituya las sentencias 110, 120 y 140 por:

```
110 C = KEYN
120 IF (C <> 13) AND (C <> 32)
    THEN I = 0
140 IF C = 13 THEN GOTO 160
```

CURSOR

En el Dragon, el BBC y el Lynx, el tercer programa se puede ejecutar (RUN), pero no producirá el resultado deseado

PEEK

Para el BBC Micro, sustituya PEEK(S) por ?S

La agenda de direcciones para el Spectrum

Ésta es la versión completa para el Spectrum del programa para la agenda de direcciones. En el próximo capítulo incluiremos los "Complementos al BASIC" correspondientes al Lynx, Dragon 32, BBC Micro, Commodore 64 y Vic-20, que harán referencia a este listado

```
1 REM *CREAR ARCHIVO DE DATOS*
2 DIM N$(50,30)
3 LET N$(1)="@VACIO"
4 INPUT"INSERTE CINTA DE DATOS, PULSE PLAY Y PULSE
  'ENTER'";A$
5 SAVE "NCAM" DATA N$()
6 INPUT"REBOBINE CINTA, PULSE PLAY, PULSE 'ENTER'";A$
7 VERIFY "NCAM" DATA N$()
8 STOP
```

Éste es el programa de inicialización que crea la matriz en cinta por primera vez. Después de ejecutar este programa, rebobine la cinta de datos, cargue (LOAD) el programa principal (listado inferior) y ejecútelo (RUN). Usted no volverá a necesitar el programa de inicialización otra vez, a menos que desee crear un nuevo archivo para la agenda de direcciones.

```
10 REM 'PROPRI'
20 REM *INICIL*
30 GOSUB 1000
40 REM *PRESEN*
50 GOSUB 3000
60 FOR M=1 TO 1
70 LET M=0
80 REM *ELECCN*
90 GOSUB 3500
100 REM *EJECUT*
110 GOSUB 4000
120 IF OPCN=9 THEN LET M=1
130 NEXT M
140 STOP

1000 REM S/R *INICIL*
1010 GOSUB 1100
1020 GOSUB 1400
1030 GOSUB 1600
1090 RETURN

1100 REM S/R *CREMAT*
1110 DIM N$(50,30)
1120 DIM M$(50,30)
1130 DIM C$(50,30)
1140 DIM D$(50,15)
1150 DIM P$(50,15)
1160 DIM T$(50,15)
1170 DIM X$(50,30)
1180 DIM B$(30) :DIM Z$(30)
1190 DIM U$(30):DIM W$(15)
1210 LET TAMA=0
1220 LET RMOD=0
1230 LET CLAR=1
1240 LET CURS=0
1250 LET Z$="@VACIO"
1260 LET Q$=B$
1300 RETURN

1400 REM S/R *LEARCH*
1405 INPUT "INSERTE CINTA DE DATOS, PULSE PLAY Y PULSE
     'ENTER'";A$
1410 LOAD "NCAM" DATA N$()
1420 IF N$(1)=Z$ THEN LET Q$=Z$:RETURN
1430 LOAD "MCAM" DATA M$()
1440 LOAD "CCAM" DATA C$()
1450 LOAD "DCAM" DATA D$()
1460 LOAD "PCAM" DATA P$()
1470 LOAD "TELCAM" DATA T$()
1480 LOAD "INDCAM" DATA X$()
1485 INPUT"PARE LA CINTA Y PULSE 'ENTER'";A$
1490 REM 'TAMAND'
1500 LET TAMA=51
1510 FOR L=1 TO 50
1520 IF N$(L)=B$ THEN LET TAMA=L:LET L=50
1530 NEXT L
1540 RETURN

1600 REM S/R *ESBAND*
1640 IF Q$=Z$ THEN LET TAMA=1
1690 RETURN

3000 REM *PRESEN*
3010 CLS
3020 PRINT:PRINT:PRINT:PRINT
3060 PRINT TAB(7);"*BIEN VENIDO A LA*"
3070 PRINT TAB(5);"*AGENDA COMPUTERIZADA*"
3080 PRINT TAB(8);"*DE MI COMPUTER*"
3090 PRINT
3100 PRINT "(PULSE BARRA ESPAC. PARA SEGUIR)"
3110 FOR L=1 TO 1
3120 IF INKEY$<>" " THEN LET L=0
3130 NEXT L
3140 CLS
3150 RETURN

3500 REM S/R *ELECCN*
3520 IF Q$=Z$ THEN GOSUB 3860:RETURN
3540 REM 'IMMENU'
3550 CLS
3560 PRINT "SELECCIONE UNO DE LOS SIGUIENTES"
3570 PRINT:PRINT:PRINT
3600 PRINT"1. HALLAR REGISTRO (DE NOMBRE)"
3610 PRINT"2. HALLAR NOMBRES (DE NOMBRE INCOMPLETO)"
3620 PRINT"3. HALLAR REGISTRO (DE CIUDAD)"
3630 PRINT"4. HALLAR REGISTRO (DE INICIAL)"
3640 PRINT"5. LISTAR TODOS LOS REGISTROS"
3650 PRINT"6. AGREGAR REGISTRO NUEVO"
3660 PRINT"7. MODIFICAR REGISTRO"
3670 PRINT"8. BORRAR REGISTRO"
3680 PRINT"9. SALIR Y GUARDAR"
3690 PRINT:PRINT
3710 REM 'ASOPCN'
3750 PRINT"DE ENTRADA A OPCION (1-9)"
3760 FOR L=1 TO 1
3770 FOR I=1 TO 1
3780 LET A$=INKEY$
3790 IF A$="" THEN LET I=0
3800 NEXT I
3810 LET OPCN = CODE A$-48
3820 IF (OPCN<1) OR(OPCN>9) THEN LET L=0
3840 NEXT L
3850 RETURN

3860 REM S/R *PRIMERA*
3870 LET OPCN=6
3880 CLS
3890 PRINT
3900 PRINT TAB(6);"NO HAY REGISTROS EN"
3910 PRINT TAB(3);"EL ARCHIVO. DEBERA EMPEZAR"
3920 PRINT TAB(4);"POR AGREGAR UN REGISTRO"
3930 PRINT
3940 REM *CONTINUAR*
3950 GOSUB 3100
3990 RETURN

4000 REM S/R *EJECUT*
4040 IF OPCN=1 THEN GOSUB 5700
4050 REM 2 ES *ENCNOM*
4060 REM 3 ES *ENCCIU*
4070 REM 4 ES *ENCINI*
4080 REM 5 ES *LISREG*
4090 IF OPCN=6 THEN GOSUB 4200
4100 IF OPCN=7 THEN GOSUB 6600
4110 IF OPCN=8 THEN GOSUB 7500
4120 IF OPCN=9 THEN GOSUB 5000
4140 RETURN

4200 REM S/R *INCREG*
4210 CLS
4220 INPUT "DE ENTRADA AL NOMBRE";N$(TAMA)
4230 INPUT "DE ENTRADA A LA CALLE";C$(TAMA)
4240 INPUT "DE ENTRADA A LA CIUDAD";D$(TAMA)
4250 INPUT "DE ENTRADA A LA PROVINCIA";P$(TAMA)
4260 INPUT "DE ENTRADA AL NUMERO DE TELEFONO";T$(TAMA)
4270 LET RMOD=1:LET CLAR=0
4280 LET X$(TAMA)=STR$(TAMA)
4290 LET Q$=""
4300 GOSUB 4500
4310 LET OPCN=0
4320 LET TAMA=TAMA+1
4350 RETURN

4500 REM S/R *MODNOM*
4510 REM CONVERTIR A MAYUSCULAS
4520 LET R$=N$(TAMA):LET S$=""
4530 FOR L=1 TO LEN(R$)
```

```
4540 LET A$=R$(L)
4550 LET T=CODE A$
4560 IF T>=97 THEN LET T=T-32
4570 LET A$=CHR$ T
4580 LET S$=S$+A$
4590 NEXT L
4600 LET R$=S$:LET S$=" ":LET A$=" ":LET T=LEN(R$):LET S=0
4610 REM LOCALIZAR PRIMER ESPACIO
4620 FOR L=1 TO T
4630 IF R$(L)=" " THEN LET S=L:LET L=T
4640 NEXT L
4650 REM DEPURAR Y COLOCAR NOMBRE DE PILA EN S$
4660 FOR L=1 TO S-1
4670 IF CODE(R$(L))>64 THEN LET S$=S$+R$(L)
4680 NEXT L
4690 REM DEPURAR Y COLOCAR APELLIDO EN A$
4700 FOR L=S+1 TO LEN(R$)
4710 IF CODE(R$(L))>64 THEN LET A$=A$+R$(L)
4720 NEXT L
4730 LET M$(TAMA)=A$+" "+S$
4740 LET S$="":LET A$="":LET S=0
4750 RETURN

5000 REM S/R #SAPROG#
5010 IF (RMOD=0) AND (CLAR=1) THEN RETURN
5020 IF (RMOD=1) AND (CLAR=0) THEN GOSUB 5200
5030 GOSUB 5600
5040 RETURN

5200 REM S/R #CLSREG#
5210 FOR K=1 TO 1
5220 LET S=0
5230 FOR L=1 TO TAMA-2
5240 LET T=L+1
5250 IF M$(L)>M$(T) THEN GOSUB 5400
5260 NEXT L
5270 IF S=1 THEN LET K=0
5280 NEXT K
5290 LET CLAR=1
5300 RETURN

5400 REM #INTERCALAR#
5410 LET U$=N$(L):LET N$(L)=N$(T):LET N$(T)=U$
5420 LET U$=M$(L):LET M$(L)=M$(T):LET M$(T)=U$
5430 LET U$=C$(L):LET C$(L)=C$(T):LET C$(T)=U$
5440 LET W$=D$(L):LET D$(L)=D$(T):LET D$(T)=W$
5450 LET W$=P$(L):LET P$(L)=P$(T):LET P$(T)=W$
5460 LET W$=T$(L):LET T$(L)=T$(T):LET T$(T)=W$
5470 LET X$(L)=STR$(L)
5480 LET X$(T)=STR$(T)
5490 LET S=1
5500 RETURN

5600 REM S/R #GRDREG#
5605 INPUT"INSERTE CINTA DE GRABACION Y PULSE
     'ENTER'";A$
5610 SAVE "NCAM" DATA N$()
5620 SAVE "MCAM" DATA M$()
5630 SAVE "CCAM" DATA C$()
5640 SAVE "DCAM" DATA D$()
5650 SAVE "PCAM" DATA P$()
5660 SAVE "TELCAM" DATA T$()
5670 SAVE "INDCAM" DATA X$()
5680 INPUT"DETENGA LA CINTA Y PULSE 'ENTER'";A$
5690 RETURN

5700 REM S/R #ENCREG#
5710 CLS
5720 IF CLAR=0 THEN GOSUB 5200
5730 PRINT:PRINT
5740 PRINT TAB(6);"BUSCANDO UN REGISTRO"
5750 PRINT TAB(9);"POR EL NOMBRE"
5760 PRINT
5770 PRINT TAB(3);"DIGITE EL NOMBRE COMPLETO"
5780 PRINT TAB(1);"POR ORDEN NOMBRE APELLIDO"
5790 PRINT:PRINT
5800 INPUT"EL NOMBRE ES";N$(TAMA)
5810 GOSUB 4500
5820 LET U$=M$(TAMA)
5830 LET INF=1
5840 LET SUP=TAMA-1
5850 FOR X=1 TO 1
5860 LET MED=INT((INF+SUP)/2)
5870 IF M$(MED)<>U$ THEN LET X=0
5880 IF M$(MED)< U$ THEN LET INF=MED+1
5890 IF M$(MED)> U$ THEN LET SUP=MED-1
5900 IF INF>SUP THEN LET X=1
5910 NEXT X
5920 IF INF>SUP THEN LET CURS=0
5930 IF INF<=SUP THEN LET CURS=MED
5940 IF CURS=0 THEN GOSUB 6400:RETURN
5950 CLS
5960 PRINT
5970 PRINT TAB(7);"#HALLADO REGISTRO#"
5980 PRINT
5990 PRINT"NOMBRE:",N$(CURS)
6000 PRINT"CALLE:",C$(CURS)
6010 PRINT"CIUDAD:",D$(CURS)
6020 PRINT"PROVINCIA:",P$(CURS)
6030 PRINT"TELEFONO:",T$(CURS)
6040 PRINT
6050 PRINT"PULSE UNA LETRA PARA IMPRIMIR"
6060 PRINT"O BARRA ESPAC. PARA CONTINUAR"
6070 FOR I=1 TO 1
6080 LET A$=INKEY$
6090 IF A$="" THEN LET I=0
6100 NEXT I
6110 IF A$<>" " THEN GOSUB 6200
6120 RETURN

6200 REM S/R #LSTCUR#
6210 LPRINT
6220 LPRINT"NOMBRE",N$(CURS)
6230 LPRINT"CALLE",C$(CURS)
6240 LPRINT"CIUDAD",D$(CURS)
6250 LPRINT"PROVINCIA",P$(CURS)
6260 LPRINT"TELEFONO",T$(CURS)
6270 LPRINT:LPRINT
6280 RETURN

6400 REM S/R #NINREG#
6410 CLS
6420 PRINT TAB(5);"#NO HALLADO REGISTRO#"
6430 PRINT TAB(4);"#EN FORMA ";N$(TAMA);" #"
6440 PRINT
6450 REM 'CONTINUAR'
6460 GOSUB 3100
6470 RETURN

6600 REM S/R #MODREG#
6610 CLS
6620 PRINT:PRINT:PRINT
6630 LET E$=CHR$ 13
6640 PRINT "#PARA MODIFICAR UN REGISTRO#"
6650 PRINT "#PRIMERO LOCALICE ESE REGISTRO#"
6660 GOSUB 5720
6670 IF CURS=0 THEN RETURN
6680 PRINT
6690 PRINT TAB(7);"MODIFICAR NOMBRE ?"
6700 PRINT
6710 PRINT "PULSE 'ENTER' PARA NUEVO NOMBRE"
6720 PRINT "O BARRA ESPAC. PARA SIGU. CAMPO"
6730 FOR I=1 TO 1
6740 LET A$=INKEY$
6750 IF (A$<>E$) AND (A$<>" ") THEN LET I=0
6760 NEXT I
6770 IF A$=E$ THEN INPUT"NUEVO NOMBRE";N$(CURS)
6780 IF A$=E$ THEN LET RMOD=1
6790 IF A$=E$ THEN LET CLAR=0
6800 IF A$=E$ THEN LET N$(TAMA)=N$(CURS)
6810 IF A$=E$ THEN GOSUB 4500
6820 IF A$=E$ THEN LET M$(CURS)=M$(TAMA)
6830 PRINT
6840 PRINT TAB(7);"MODIFICAR CALLE ?"
6850 PRINT
6860 PRINT "PULSE 'ENTER' PARA NUEVA CALLE"
6870 PRINT "O BARRA ESPAC. PARA SIGU. CAMPO"
6880 FOR I=1 TO 1
6890 LET A$=INKEY$
6900 IF (A$<>E$) AND (A$<>" ") THEN LET I=0
6910 NEXT I
6920 IF A$=E$ THEN LET RMOD=1
6930 IF A$=E$ THEN INPUT"NUEVA CALLE";C$(CURS)
6940 PRINT
6950 PRINT TAB(7);"MODIFICAR CIUDAD ?"
6960 PRINT
6970 PRINT "PULSE 'ENTER' PARA NUEVA CIUDAD"
6980 PRINT "O BARRA ESPAC. PARA SIGU. CAMPO"
6990 FOR I=1 TO 1
7010 LET A$=INKEY$
7020 IF (A$<>E$) AND (A$<>" ") THEN LET I=0
7030 NEXT I
7040 IF A$=E$ THEN LET RMOD=1
7050 IF A$=E$ THEN INPUT"NUEVA CIUDAD";D$(CURS)
7060 PRINT
7070 PRINT TAB(5);"MODIFICAR PROVINCIA ?"
7080 PRINT
7090 PRINT "PULSE 'ENTER' PARA NUEVA PROV."
7100 PRINT "O BARRA ESPAC. PARA SIGU. CAMPO"
7110 FOR I=1 TO 1
7120 LET A$=INKEY$
7130 IF (A$<>E$) AND (A$<>" ") THEN LET I=0
7140 NEXT I
7150 IF A$=E$ THEN LET RMOD=1
7160 IF A$=E$ THEN INPUT"NUEVA PROVINCIA";P$(CURS)
7170 PRINT
7180 PRINT "MODIFICAR NUMERO DE TELEFONO ?"
7190 PRINT
7200 PRINT "PULSE 'ENTER' PARA NUEVO TELEF."
7210 PRINT "O BARRA ESPACIADORA PARA SIGU. CAMPO"
7220 FOR I=1 TO 1
7230 LET A$=INKEY$
7240 IF (A$<>E$) AND (A$<>" ") THEN LET I=0
7250 NEXT I
7260 IF A$=E$ THEN LET RMOD=1
7270 IF A$=E$ THEN INPUT"NUEVO NUMERO";T$(CURS)
7280 RETURN

7500 REM S/R #BORREG#
7510 CLS
7520 PRINT:PRINT:PRINT:PRINT
7530 LET E$=CHR$ 13
7540 PRINT TAB(3);"#PARA BORRAR UN REGISTRO#"
7550 PRINT TAB(1);"#LOCALICE PRIMERO REG. DESEADO#"
7560 GOSUB 5720
7570 IF CURS=0 THEN RETURN
7580 PRINT
7590 PRINT "DESEA BORRAR ESTE REGISTRO ?"
7600 PRINT "#AVISO# - NO HAY SEGUNDA OPORTUNIDAD !"
7610 PRINT
7620 PRINT TAB(3);"PULSE 'ENTER' PARA BORRAR"
7630 PRINT TAB(1);"O BARRA ESPAC. PARA CONTINUAR"
7640 FOR I=1 TO 1
7650 LET A$=INKEY$
7660 IF (A$<>E$) AND (A$<>" ") THEN LET I=0
7670 NEXT I
7680 IF A$=" " THEN RETURN
7690 FOR L=CURS TO TAMA-2
7700 LET T=L+1
7710 LET N$(L)=N$(T)
7720 LET M$(L)=M$(T)
7730 LET C$(L)=C$(T)
7740 LET D$(L)=D$(T)
7750 LET P$(L)=P$(T)
7760 LET T$(L)=T$(T)
7770 LET X$(L)=X$(T)
7780 NEXT L
7790 LET RMOD=1
7800 LET TAMA=TAMA-1
7810 RETURN
```

Un reto universitario

El primer ordenador programable del mundo se ideó en la Universidad de Manchester

Acabada la segunda guerra mundial, la Universidad de Manchester nombró a dos nuevos profesores. Max Newman fue nombrado profesor de matemáticas después del trabajo de descifrar códigos que realizó en Bletchley Park con el Colossus, el primer ordenador electromecánico del mundo, y el ingeniero de radares, F. C. Williams, obtuvo el puesto de ingeniería eléctrica. Williams se llevó consigo a un joven ayudante, Tom Kilburn, que se había familiarizado con los problemas de dispositivos de impulsos con memoria electrónica mientras trabajaba con los radares durante la guerra. Posteriormente, Kilburn se convertiría en el primer profesor de la recién creada cátedra de estudios informáticos en esa Universidad.

Durante un viaje realizado en 1946 a Estados Unidos de visita a las instalaciones de radares, Williams había visto el prototipo del ordenador de válvulas, ENIAC (véase p. 46) y, de regreso a Gran Bretaña, consiguió que la Royal Society invirtiera 35 000 libras esterlinas en un "laboratorio de máquinas de calcular" en la Universidad de Manchester. Éste no era el único centro docente empeñado en construir un ordenador de programas almacenados. La Universidad de Pennsylvania estaba construyendo el EDVAC, en la Universidad de Cambridge se estaba trabajando en el EDSAC y en el National Physical Laboratory se seguía trabajando en la elaboración del ACE (véase p. 88). No obstante, en todos estos proyectos se estaba utilizando un almacenamiento de memoria construido a partir de tubos en línea de demora de mercurio. El equipo de Manchester iba a hacer su máquina con un dispositivo de memoria, invención del propio Williams, que utilizaba un tubo de rayos catódicos. Para el otoño de 1947, Williams había conseguido retener 2 048 bits durante varias horas.

Gracias al "tubo Williams", en junio de 1948 el ordenador Manchester Mark I ejecutó con éxito un programa, convirtiéndose en el primer ordenador del mundo con programa almacenado. El Mark I podía ejecutar una instrucción en 1,2 milésimas de segundo. Un tubo de rayos catódicos permitía almacenar la información, la memoria tenía la ventaja de ser de acceso directo y se podía representar visualmente el contenido del almacenamiento principal o del registro de control.

Una vez demostrada la viabilidad del uso de un "tubo Williams" para el almacenamiento de memoria, se construyó una versión mejorada del Mark I, que se aplicó en problemas del diseño de óptica y en la generación de números primos. El consejero científico del Estado, sir Ben Lockspeiser, quedó tan impresionado con el rendimiento del ordenador, que encargó la construcción de una versión comercial del Mark I a una empresa local de Manchester. El Ferranti Mark I salió al mercado en febrero de 1951, anticipándose en cinco meses al UNIVAC; fue, pues, el primer ordenador disponible comercialmente.

Una importante innovación del Ferranti Mark I era su capacidad para modificar las instrucciones durante el proceso en virtud de otro almacenamiento denominado tubo "B". En el momento requerido, éste podía agregar su contenido al registro de control y, por consiguiente, modificar el código de la instrucción original. Este principio aceleraba la ejecución de los programas. En sus primeros ordenadores la IBM utilizó algunas de las patentes del Manchester, y en una visita a la oficina central de la sociedad, en Nueva York, blasonada por doquier con el lema THINK (PIENSA) de la empresa, se le preguntó a Williams cómo fue que el equipo de Manchester había logrado construir un ordenador cuando todos los esfuerzos de la IBM habían fracasado. Williams respondió sin titubear: "¡Es que nosotros no nos paramos a pensarlo tanto!"

La llegada de Alan Turing (véase p. 200) a Manchester, en 1948, fue un gran estímulo para las actividades de programación. En 1950, Turing sacó a la luz el primer manual de programación de Manchester. Dos años después, al equipo de Manchester ya le rondaba la idea de construir un ordenador más compacto y económico. Sus planes se aceleraron con la invención del transistor, y en noviembre de 1953 entraba en funcionamiento en Manchester el primer ordenador de transistores del mundo.

A fines de la década de los cincuenta, Estados Unidos se estaba adelantando en cuanto a la tecnología de ordenadores, a resultas de lo cual el gobierno británico decidió invertir en un proyecto que ayudaría a que Gran Bretaña recuperara su liderazgo. En diciembre de 1962 se encargó el ordenador Atlas, que se construyó bajo la dirección de Tom Kilburn. Empleaba una palabra de 48 bits con formato de dirección sencillo, un almacenamiento principal de 16 Kbytes y una memoria en tambor de lectura solamente de ocho Kbytes. Se vendieron unidades al Atomic Energy Research Establishment, en Harwell, y a la British Petroleum.

El Manchester Mark I
Por haber logrado ejecutar un programa con pleno éxito en junio de 1948, se puede decir que el Manchester Mark I fue el primer ordenador del mundo con programa almacenado. Ferranti, entonces una empresa local, recibió el encargo de fabricar una versión comercial del ordenador, que salió al mercado a principios de 1951

Cortesía de la Universidad de Manchester

Apuestas por ordenador

Rex Features Ltd.

Un día en las carreras
Las carreras de caballos son un campo interesante para la aplicación de ordenadores personales, pero sólo para aquellas personas que saben lo que están haciendo. Así, por lo menos un criador utiliza un microordenador Apple II para llevar una base de datos con la raza de todos sus caballos, y valerse de ellos para intentar que salgan campeones

Los ordenadores ya tienen muchas aplicaciones en el mundo de las apuestas. Existen programas de quinielas incluso para ordenadores personales

Las apuestas se basan en la probabilidad, aunque a muchos jugadores les gustaría que se basaran en la certeza de ganar. Este deseo, por supuesto, carece de fundamento, dado que la inmensa mayoría de los jugadores pierden las más de las veces y, con frecuencia, grandes cantidades. Ello se debe a que las probabilidades están en contra de ellos y a favor del casino, del corredor de apuestas y del Patronato de Apuestas Mutuas. Para ver si los ordenadores pueden contribuir a compensar el equilibrio, primero debemos considerar estas probabilidades.

Despojados de sus adornos exteriores, todos los juegos de azar se reducen a apostar sobre un suceso aleatorio. Éste suele generarse mediante algún dispositivo, como una bola que gira dentro de la rueda de una ruleta, o una carta extraída de un mazo de naipes cuidadosamente barajado. Si se conocen los parámetros (la cantidad de naipes, p. ej.), la teoría de la probabilidad permite efectuar ciertas predicciones acerca de la verosimilitud de que el hecho fortuito se produzca. Por ejemplo, la ruleta que se utiliza en los casinos posee 37 hendiduras numeradas del 0 al 36. Existen, por lo tanto, 18 hendiduras de número impar y 18 hendiduras de número par donde puede caer la bola, más el cero. Las probabilidades de que la bola se deposite en una hendidura de número impar se pueden expresar como 18/37, o 0,4864864, o de un poco más del 48,6 %. Este porcentaje es ligeramente inferior a la probabilidad de que una moneda, después de lanzarla al aire, caiga del lado cara; la diferencia, determinada por la presencia de la hendidura del cero, representa la "ventaja" de la casa o su margen de beneficios.

Es esta "ventaja" lo que hace que los juegos de azar, y la mayoría de todas las otras formas de apuesta, sean tan poco gratificantes. A pesar de las argumentaciones que ocasionalmente aducen los proveedores de sistemas de apuestas, los ordenadores no pueden hacer nada por mejorar las probabilidades básicas de un juego determinado.

Tales objeciones teóricas no han conseguido desalentar a los inventores entusiastas y en la actualidad a los propietarios de ordenadores personales se les ofrece una variedad de sistemas de apuestas supuestamente a prueba de fallos, algunos de los cuales hasta parecen funcionar y todo. En el caso de los sistemas de apuestas para casino, éstos demuestran casi invariablemente ser variantes de la *duplicación*, un procedimiento que adolece del inconveniente de que para alcanzar el éxito hay que aportar una cantidad infinita de dinero al juego. Existe, asimismo, otro problema: aunque técnicamente no es ilegal, la administración de los casinos no permite usar ningún tipo de ordenador. Presumiblemente los ordenadores son de más ayuda cuando están implicadas la pericia y la estrategia.

Lamentablemente, la pericia exigida por la mayoría de los juegos de apuestas más populares es

mínima. Un caso concreto es el de las quinielas futbolísticas. En Gran Bretaña, el programa para predicción de quinielas más completo es el célebre *F4 Football Forecast*, del profesor Frank George, que está a la venta en versiones aptas para la mayoría de los ordenadores personales. Analizando las estadísticas de los últimos diez años, el programa le atribuye un valor al rendimiento promedio de cada equipo. Ajustando una línea para sopesar su situación a largo plazo (inferido a partir de las posiciones en las tablas de liga), su situación a corto plazo y el resultado del último partido, una comparación de estas tres cifras de rendimiento le permite al programa predecir el resultado probable de un partido determinado.

Si el líder juega en casa contra el último de la clasificación, el resultado puede ser incuestionable, pero el punto fuerte del programa está en predecir el desenlace de un encuentro entre equipos más nivelados. Ello no equivale a afirmar que este sistema acierte con seguridad. El análisis estadístico sugiere que con el programa del profesor George aproximadamente se triplican las probabilidades de éxito: "Admito que los casos no favorables de acierto siguen siendo enormes, pero ¿es o no es mejor apostar con el mayor conocimiento de causa posible?", pregunta el profesor.

La noticia ha saltado a las calles de algunas ciudades españolas, respondiendo a esta pregunta. Varios acertantes de quinielas millonarias lo han sido gracias —según sus propias declaraciones— al ordenador. Parece, pues, que la informática comienza a domesticar el azar.

Las carreras de caballos parecen ofrecer, en todo caso, una mayor esfera de acción para el programador. Un colegial de Darlington ha creado un programa para ordenadores personales que predice los ganadores. Escrito originalmente para el Sinclair ZX-81, y ahora adaptado para el Spectrum, el programa de David Stewart ha dado en el clavo en cierto número de ocasiones.

Quizá sea significativo que los profesionales de las carreras se hayan negado mayoritariamente a utilizar ordenadores. El Jockey Club todavía sigue efectuando manualmente el handicap oficial (aunque los datos se almacenan en ordenador). *Tuneform*, la biblia del aficionado a las carreras, también compila de forma manual la mayoría de sus datos. "Nosotros sólo empleamos el ordenador para calcular las cifras de tiempo promedio para cada caballo, teniendo en cuenta la dirección del viento", explica el director gerente de la publicación, Reginald Griffin. "No existe algo así como un auténtico sistema de handicap informatizado en ningún sitio. El problema es, simplemente, que los ordenadores no pueden dar cuenta de los extraordinarios resultados que suceden a diario."

David W. Hamilton

La rueda de la fortuna
En la ruleta, es el número cero el que le proporciona al casino el margen de beneficios. No se puede hacer nada por aumentar las probabilidades del jugador, así que los programas que se desarrollen para este juego se deben concentrar en los sistemas de apuesta

La informática en las quinielas
Existen varios paquetes para ordenadores personales de los que se afirma que mejoran las probabilidades de ganar en las quinielas, y muchísimos programadores han intentado escribir sus propios programas. Los mejores utilizan una amplia base de datos conteniendo información sobre partidos anteriores, y pueden resultar valiosos para predecir el resultado de partidos decisivos. Como toda forma de apuesta asistida por ordenador, los programas sólo pueden mejorar sus pronósticos de forma marginal, y, por tanto, los proveedores declinan toda responsabilidad

Los ordenadores se están utilizando cada vez más al otro lado del mostrador del establecimiento de apuestas, aunque no para calcular las probabilidades. El personal empleado en las grandes cadenas de corredores de apuestas está entrenado para utilizar calculadoras especiales exclusivamente para averiguar los beneficios sobre las cantidades apostadas. El aspecto crediticio del negocio también se está informatizando cada vez más. Una persona que posea una cuenta en una de las agencias de apuestas pertenecientes a la cadena puede, simplemente, hacer su apuesta comunicándola por teléfono al centro de ordenadores de la cadena. Los detalles se digitan y se adeuda en la cuenta el valor de la apuesta. Si el caballo escogido sale ganador, se calculan las ganancias y se ingresan en la cuenta del cliente.

La mayor parte de los corredores de apuestas se muestran escépticos respecto al uso del ordenador. "Nunca nadie ha propuesto uno que gane constantemente; de lo contrario, no estaríamos aquí", ha declarado el portavoz de la William Hill. Así y todo, fue esta firma la que montó una de las simulaciones por ordenador más extraordinarias y controvertidas de todos los tiempos. Los detalles relativos al estado de los antiguos ganadores del clásico Derby se incorporaron a un programa encargado especialmente. Entonces se invitó a los lectores del periódico a que predijeran los primeros seis. La controversia surgió al asignar un puesto al gran caballo italiano *Ribop*, que jamás perdió una carrera. ¡El ordenador lo colocó en cuarto lugar!

De todos los ordenadores "para apuestas", quizá el más famoso sea ERNIE (*Electronic Random Number Indicator Equipment*: equipo electrónico indicador de números al azar), la máquina que elige los números ganadores de los bonos emitidos

por las cajas de ahorros británicas. Se puede discutir si ERNIE es realmente un ordenador en toda regla; pero, aunque no es programable, sí ejecuta un programa. La máquina la desarrolló Plessey en 1973 para sustituir a la máquina original, construida en 1957. Su función consiste en generar al azar alrededor de 200 000 números y escribirlos en cinta magnética. Estos números se generan a partir de una serie que empieza por el bono premiado que posea el número de emisión más bajo, y termina por el más alto. La cinta se carga luego en un ordenador de unidad principal ICL, que compara los números con una cinta en la que están listados todos los números de aquellos bonos que ya se han reintegrado. Una vez que se han eliminado estos números no elegibles, el ordenador puede imprimir los impresos de cobro para los ganadores.

Desde que se encargaron, los dos ERNIE han generado los números para 22,2 millones de premios, que suman unos 265 millones de pesetas. ¿Le parece mucho? En realidad, no es tanto; las posibilidades de que un bono obtenga un premio en la tirada mensual son de apenas una entre 15 000.

Otro juego de azar muy conocido es el que aparece diariamente en algunos periódicos británicos. Las probabilidades de ganar uno de los premios de un millón de libras que se ofrecen en estas campañas de promoción son todavía más remotas. Los números de los cupones que se distribuyen a los lectores tienen una longitud de 12 dígitos. Una secuencia de dígitos comprendida entre 000000000000 y 999999999999 ofrece un billón de combinaciones distintas. Estadísticamente, las probabilidades en contra de que una mañana determinada aparezca un número dado serían algo así como un millón de millones contra uno. Sobre esta base, resulta sumamente dudoso que el periódico tenga que pagar *alguna vez* el premio gordo.

La situación se puede visualizar mejor si nos imaginamos un bolso que contenga dos millones y medio de bolas blancas, que representan a los participantes (el número de cupones en circulación), y un billón de bolas negras que representan el total de números distintos posibles. Huelga decir que la probabilidad de extraer una bola blanca a la primera es más bien remota. Además, la probabilidad no mejora significativamente ni siquiera después de que se hayan extraído las correspondientes a un año. Las probabilidades de que el periódico tenga que pagar un millón de libras son de una entre 667.

Un golpe de suerte
En *blackjack* (o "veintiuno") existe como juego para la mayoría de los ordenadores personales, y proporciona una de las mejores pautas para escribir programas ganadores. La capacidad de memorizar los naipes que ya se han jugado aumenta las probabilidades de éxito del jugador, aunque como los casinos no admiten ordenadores en las mesas de juego, en todas las hazañas célebres han participado ordenadores ocultos (ha habido un caso en que fue atado a la pierna del jugador, cubierto con la pernera de su pantalón) o enlaces por radio con máquinas situadas en el exterior

Cargando los dados

El ingrediente básico de las apuestas, la generación de números al azar, se puede simular fácilmente en un ordenador personal. La mayoría de las versiones de BASIC proporcionan una función generadora de números aleatorios. Sin embargo, en muchos casos los números que se generan de este modo no son auténticamente al azar, como demuestra el breve programa que ofrecemos a continuación:

```
10 LET A = RND
20 LET B = RND
30 LET C = RND
40 PRINT A,B,C
```

En cada una de las tres primeras líneas, se les asigna un número supuestamente al azar a las variables A, B y C. Luego, éstas se imprimen. Podremos obtener los resultados siguientes (aunque los suyos diferirán):

```
.014007 .964370 .457397
```

Pero si vuelve a ejecutar el programa, la mayoría de los microordenadores visualizará otra vez la misma secuencia. Y se explica: cuando solicitamos RND, el ordenador responde con el siguiente número de una secuencia fija. Por norma, ésta puede abarcar el millón de fracciones de seis dígitos entre 0,000000 y 0,999999, cada una de las cuales se selecciona una vez en el ciclo completo (pero no en secuencia).

Ciertas versiones de BASIC utilizan una sintaxis ligeramente diferente, que exige una expresión entre paréntesis denominada *argumento*. Éste asume la forma LET A = RND(X). El efecto es muy similar: tanto RND como RND(X) se pueden utilizar de la misma forma que otras variables.

Algunas versiones de BASIC también incorporan una función RANDOMIZE, que hace que la secuencia empiece en un punto impredecible. Insertando la orden RANDOMIZE en las primeras líneas de cualquier programa en el cual se utilice RND, se asegura que cada vez que se ejecute el programa se genere una secuencia de números distinta.

Para simular el lanzamiento de un dado necesitamos enteros en la escala entre 1 y 6. O sea, es necesario eliminar las fracciones. Esto se realiza utilizando la función INT (entero). PRINT INT(6.99) produce como resultado 6, al igual que PRINT INT(6.01) producirá también 6. Todo cuanto venga después de la coma decimal se descarta.

Puesto que el máximo número que puede generar RND es .999999, se requiere una pequeña multiplicación. La fórmula generalmente aceptada es:

```
LET A = INT(6*RND) + 1
```

Multiplicamos por seis porque un dado posee seis caras. El "más uno" es sencillamente para asegurar que el resultado esté comprendido entre 1 y 6, y no entre 0 y 5.

El lenguaje ensamblador

Continuando con nuestra introducción al código de lenguaje máquina, analizamos las formas de expresar los programas, desde los números binarios al lenguaje ensamblador

Una de las dificultades conceptuales que experimenta la mayoría de los recién iniciados en el código de lenguaje máquina es el hecho de que los programas puedan asumir diversas formas. Todo dato almacenado en la memoria del ordenador, en última instancia asume la forma de números binarios de ocho dígitos. Sin embargo, cuando éstos se escriben sobre papel, ocupan muchísimo espacio, son difíciles de leer y recordar y propician los errores de digitación. Por eso se recurre a los números en base dieciséis (hexa), pues ofrecen las ventajas de que el contenido de un byte se puede expresar como un número de dos dígitos y que cualquier dirección en la memoria del ordenador (de 0 a 65535 en decimal) se puede representar mediante cuatro dígitos.

Cuando escribimos un número hexa sobre el papel por lo general le ponemos delante un signo $ para distinguirlo de los números decimales, bien sabiendo que cuando se da entrada al programa el signo no se incorpora a la memoria del ordenador. Además, cuando un opcode posee un operando de dos bytes (p. ej., LDA $3F80), a los dos bytes se les da entrada en la máquina por orden inverso; es decir, el byte de la derecha seguido del byte de la izquierda. En el ejemplo anterior, por tanto, los tres bytes serían AD (la representación hexa del opcode LDA en lenguaje 6502) seguido de 80, seguido de 3F. Ello simplifica mucho las cosas para el procesador, pero puede resultarle confuso al usuario.

Normalmente, un programa en código de lenguaje máquina se imprime como un *dump* (volcado) en hexadecimal: una larga lista de valores hexadecimales de dos dígitos. Además, se dará una dirección de comienzo (ya sea en hexa o en decimal) y es en esta posición donde se debe cargar el primer valor hexa, el segundo en la posición siguiente, y así sucesivamente. La carga se puede conseguir mediante la orden POKE del BASIC. Si la dirección de comienzo es $1000 (4096, en decimal), y el *dump* hexa es:

```
AD      (173 en decimal)
80      (128 en decimal)
3F      (63 en decimal)
```

el programa se puede cargar con estas tres sentencias en BASIC:

```
POKE 4096,173
POKE 4097,128
POKE 4098,63
```

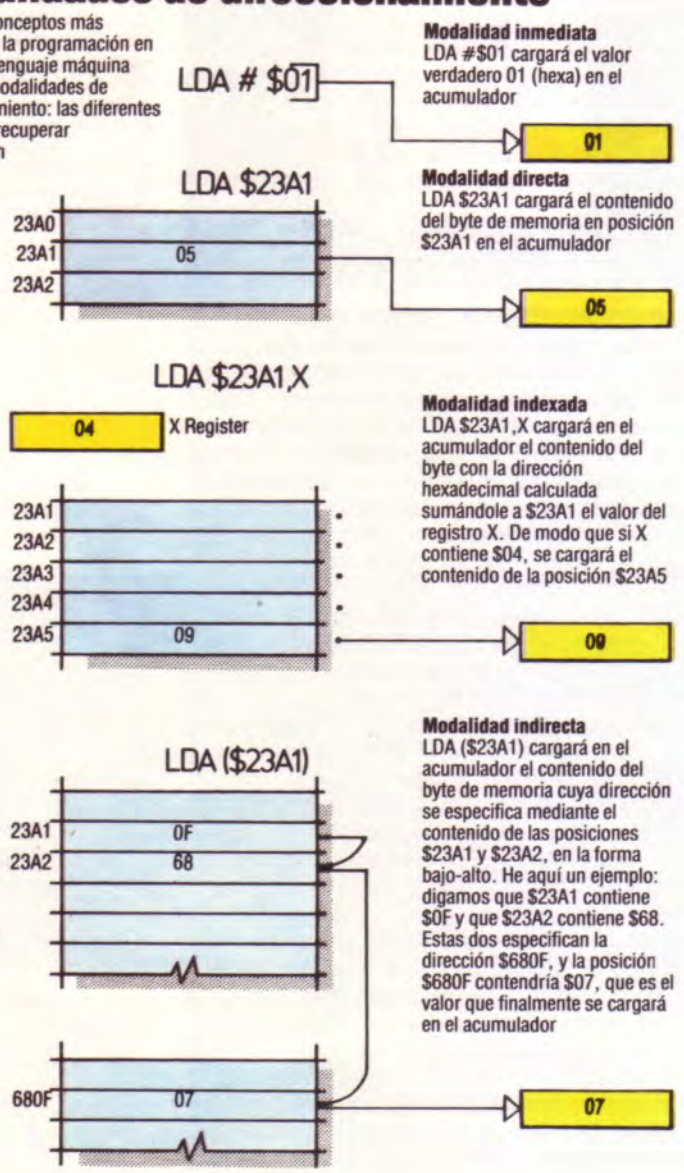
Observe cómo hemos de convertir todos los valores de hexa a decimal antes de que los podamos utilizar en la sentencia POKE; en el interior de la máquina se almacenarán en binario.

Para *dumps* hexa más largos es normal emplear un breve programa en BASIC denominado *cargador de código máquina*. Éste pregunta la dirección de comienzo y después los valores hexa. A medida que se les va dando entrada a cada uno, la breve rutina en BASIC convierte el valor hexa en decimal, y lo coloca (POKE) en la posición siguiente. Si se desea, el *dump* hexa lo puede leer (READ) el programa por medio de sentencias DATA.

Una vez que se ha cargado el código máquina, se puede prescindir del programa cargador en BASIC. Por consiguiente, es importante cargar el código máquina en algún lugar de la memoria donde el

Modalidades de direccionamiento

Entre los conceptos más eficaces de la programación en código de lenguaje máquina están las modalidades de direccionamiento: las diferentes formas de recuperar información



Versión íntegra y abreviada

Un programa en código de lenguaje máquina puede asumir varias formas diferentes. El programador lo suele escribir en forma de lenguaje ensamblador, que utiliza mnemotécnicos para los opcodes y etiquetas para los operandos, de modo que:

```
LDA PESO
ADC COMBUSTIBLE
STA PESO
```

Debemos, no obstante, especificar las direcciones de aquellas etiquetas. Por ejemplo:

```
COMBUSTIBLE = $03EE
PESO = $031F
```

Un paquete ensamblador transformaría esto en un *dump* hexa, utilizando una unidad de disco. El "lenguaje seudoensamblador", como vemos más abajo, resulta menos fácil de leer, pero a menudo se le puede dar entrada en un paquete denominado *ensamblador puntual*, que no requiere discos.

```
LDA $031F
ADC $03EE
STA $031F
```

Un *dump* hexa consta de una dirección de comienzo (a la izquierda) y la secuencia de valores hexa de dos dígitos como aparecerán en la memoria. Observe que un operando como $031F se almacena en orden inverso (1F 03) y que los opcodes se han sustituido por el valor hexa adecuado:

```
19C4 AD 1F 03 6D EE 03 8D 1F 03
```

programa en BASIC no lo "machaque", ni donde sentencias como NEW lo destruyan.

La mayoría de los ordenadores personales posee alguna orden en BASIC para indicarle a la máquina que deje de ejecutar BASIC y que empiece a ejecutar el programa en código máquina a partir de un punto determinado. Una forma de esta orden es SYS 4096 (RETURN), que significa "transfiere el control al sistema comenzando por la posición decimal 4096"; otra forma es CALL $E651, que significa "llama a la rutina en código máquina a partir de la posición hexa E651".

El programa o la subrutina en código máquina ejecutará entonces este sistema o rutina (con los resultados visibles o no, según la naturaleza del programa). Si está escrita correctamente y si incorpora el procedimiento de finalización adecuado, el control se volverá a pasar al BASIC. Entre paréntesis, esto significa que se puede llamar a subrutinas en código máquina desde varios lugares durante la operación de un programa en BASIC cuando sea necesario efectuar una función a gran velocidad.

Uno de los inconvenientes de programar en código máquina es que si usted ha cometido un error en su código, el ordenador no le asistirá con su útil y caritativo mensaje ERROR DE SINTAXIS. En cambio, es más que probable que le "reviente": la máquina no responderá a nada que le digite. Esto no es perjudicial para el ordenador, pero tendrá que restablecerlo (o apagar la máquina y volver a encenderla) y esto por norma significa que tendrá que volver a introducir de nuevo el programa partiendo de cero. Ésta es la razón por la cual uno no puede experimentar en código máquina del modo como puede experimentar en BASIC: la viabilidad de un programa debe verificarse exhaustivamente sobre el papel antes de darle entrada en el ordenador.

No obstante, un dispositivo de software que puede ser de gran ayuda para dar entrada y verificar el código máquina es el *monitor de código de lenguaje máquina* (que no tiene nada que ver con un monitor de pantalla). Algunos ordenadores, muy pocos, lo llevan incorporado en su ROM, pero por lo general se compra en forma de paquete sobre cassette o cartucho. Un monitor de código máquina es un sistema operativo sencillo que visualiza en la pantalla el contenido de cualquier sector de la memoria que se solicite. Estos valores (hexa) se pueden simplemente modificar o bien volver a escribir sobre ellos, de modo que un monitor es, con mucho, la forma más rápida de dar entrada a un *dump* hexa. Además, por lo general puede cargar y guardar los programas en código máquina directamente en cassette, sin necesidad del programa cargador en BASIC. Los más avanzados programas de utilidades en código máquina (el equivalente en código máquina a los equipos de herramientas de BASIC; véase p. 444) muestran el contenido de cada uno de los registros internos del procesador.

Los *dumps* hexa constituyen una conveniente forma de expresar el código de lenguaje máquina, pero no son fáciles de leer. A menos que recuerde el equivalente hexadecimal de todos los diversos opcodes, es casi imposible distinguirlos de los operandos. De modo que los programas se suelen escribir con un artificio mnemotécnico de tres letras que ya vimos en el capítulo anterior (véase p. 449), y éstas se traducen a hexa con una tabla de códigos que figura en el manual del microprocesador.

Sin embargo, un tipo más refinado de monitor de código máquina le permitirá digitar el programa mnemotécnico en su totalidad y obtener las conversiones de manera automática. Es el llamado *ensamblador puntual*, porque ensambla las letras mnemotécnicas en forma de números sobre la marcha.

Esto nos lleva a la forma final en la que se puede expresar el código máquina, el lenguaje ensamblador, que no sólo recurre a la mnemotecnia para los opcodes sino que puede manipular nombres (o etiquetas) en vez de los números hexas para los operandos. Por tanto, si la posición $07B2 contiene el número actualizado de misiles que se disparan en un juego, nosotros podemos cargarlo en el acumulador con la instrucción:

```
LDA MISSIL
```

Al comienzo del programa tendremos que especificar la posición MISSIL = $07B2, y asegurar que contenga inicialmente el valor $09 (nueve misiles).

Cuando hemos acabado de desarrollar este programa en lenguaje ensamblador (denominado *código fuente* del programa), ejecutamos un programa de utilidades que recibe el nombre de *ensamblador*. Éste revisa el código, sustituyendo las palabras mnemotécnicas y todas las etiquetas por sus equivalentes hexas y creando, por consiguiente, una nueva versión denominada *código objeto*. A este código se le puede dar entrada en la memoria del ordenador y se puede ejecutar. El proceso es similar al de la compilación (véase p. 84), si bien en este caso existe una correspondencia individualizada entre el código fuente y el código objeto.

El lenguaje ensamblador, al ser un lenguaje de mayor nivel que el código de lenguaje máquina, es considerablemente más fácil de escribir, sin merma de eficacia. No obstante, los paquetes ensambladores por lo general sólo funcionan con una unidad de disco, y por ello no están al alcance de todos los usuarios de ordenadores personales.

Opcodes

He aquí algunos opcodes más, que un microprocesador corriente incorporará

JSR

Jump SubRoutine
(Salto a subrutina)
Esta función equivale a la GOSUB del BASIC. JSR $354D modificará el contenido del registro contador (PC) del programa de modo que ejecute el código desde $354D hacia adelante

RTS

ReTurn from Subroutine
(Regreso de subrutina)
Al encontrarse con RTS, el procesador saltará hacia atrás hasta la posición desde la cual se llamó a la subrutina (o sea, equivale a RETURN en BASIC). RTS no tiene operando, porque la dirección de retorno se habrá almacenado automáticamente en una zona especial de la memoria denominada *stack* (pila, montón)

BMI

Branch if MInus
(Bifurcación si menos)
Ésta es una de las varias formas de bifurcación condicionada en código máquina (en BASIC, IF...THEN GOTO es una bifurcación condicionada). Si en la última operación se obtuvo como resultado un valor negativo en el acumulador, la ejecución del programa saltará a una dirección especificada. BPL hace lo mismo cuando se obtiene un signo positivo (PL*us*)

LDX

LoaD X register
(Carga en registro X)
X es un registro de un único byte dentro del microprocesador, y si bien no puede efectuar aritmética del mismo modo que el acumulador, se lo utiliza para el "direccionamiento indexado" (véase cuadro). LDX carga un valor en X, y STX (STore X: almacenar X) lo volverá a almacenar en la memoria

INX

INcrement X
(Incrementar X)
Sumando 1 al valor de X (DEX le restaría 1: DEcrement X) y utilizando el direccionamiento indexado, se puede atravesar un número de posiciones de la memoria, realizando en cada una de ellas el mismo proceso

El futuro

Los últimos cinco años han resultado cruciales para la revolución informática; pero ¿qué nos deparará el próximo lustro?

¿Qué aspecto tendrá el ordenador personal de la última década del siglo XX y cómo funcionará? Son preguntas a las que tratamos de dar respuesta en este artículo, repasando uno a uno los principales componentes y sistemas de la máquina del mañana. Muchas de estas ideas se basan en tecnologías que están comenzando a salir al mercado (quizá en otros campos distintos de la informática), mientras que otras representan lo que nosotros creemos que son desarrollos probables.

Una de las configuraciones más fundamentales de nuestro diseño hipotético es la modularidad. Habiendo adquirido la unidad base, el usuario dispondrá de una amplia gama de opciones para ampliar la máquina. En realidad, el usuario podrá virtualmente diseñar su propia máquina seleccionando este módulo para gráficos y aquella facilidad para el sonido. Algo es seguro: la velocidad de los cambios en el mercado del ordenador seguirá en continua aceleración muchos años más.

1 Visualización de teclado

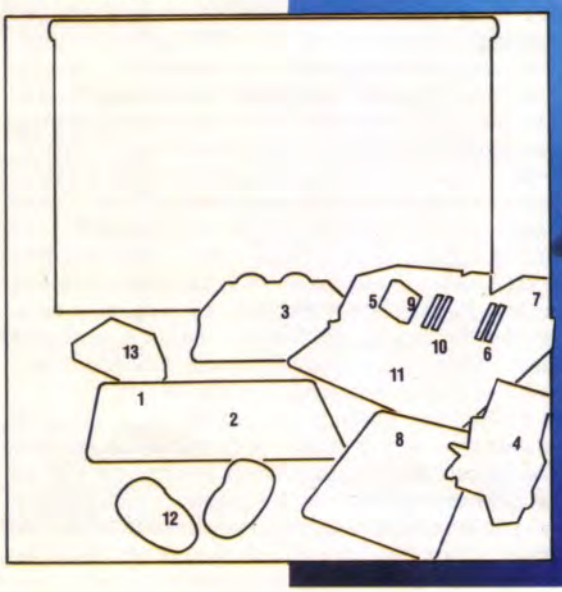
La potencia del microprocesador de 32 bits permitirá visualizar la información en varias formas simultáneamente. Por ejemplo, la pantalla principal podría mostrar la vista que se contempla desde el sillón de mando de una nave espacial, mientras una pantalla subsidiaria montada sobre la consola de mando-teclado podría visualizar la información de control desde la cabina

2 Teclado

A pesar de la innata ineficacia del teclado QWERTY, es poco probable que se efectúen intentos serios por establecer un modelo alternativo. Las teclas con resorte estilo máquina de escribir son con mucho las que gozan de mayor popularidad, aunque es probable que se extienda la utilización de las teclas de efecto Hall, que emplean imanes en vez de resortes. Los interruptores electrónicos propiamente dichos podrían reemplazarse por un sistema que se basa en que las teclas conmutan una matriz de rayos láser

3 Monitor

Los televisores de proyector existen desde comienzos de la década de los 80, pero su campo de acción se ve limitado por la potencia de emisión de luz del tubo de rayos catódicos (TRC). Es probable que la tecnología TRC ponga a nuestro alcance sistemas de proyección que abarquen todo el ancho de una habitación. Los primeros televisores de proyector necesitaban pantallas curvadas especiales, pero los modelos más recientes ya se enfocan contra una superficie plana



Tony Lodge

4 Procesadores alternativos

Además del procesador principal de 32 bits, es probable que el micro dentro de unos años albergue procesadores adicionales a manera de módulos enchufables. El procesador principal podría entonces "delegar" parte del procesamiento (p. ej., la operación de un periférico determinado o clasificar un archivo de datos) al subprocesador más idóneo. Además, los módulos enchufables baratos podrían emular a los ordenadores clásicos de los años ochenta, de modo que el software de cualquier otro ordenador se podría ejecutar sin ninguna modificación

5 Memoria de acceso directo

El procesador de 32 bits puede direccionar hasta casi 4 300 millones de posiciones de memoria, lo que representa una diferencia abismal en relación al límite de 65 536 bytes impuesto por los procesadores de ocho bits con los que han entrado los microordenadores en el hogar

6 Comunicaciones

Mientras que las antenas parabólicas para recibir señales emitidas por satélites serán algo corriente en los años noventa, y la mayoría de los canales telefónicos estarán digitalizados en vez de basarse en señales analógicas, aun entonces será necesario regular la velocidad de transmisión y recepción. Estos controladores de comunicaciones realizarán parte de las funciones de control de los moduladores-demoduladores actuales

7 Alimentación eléctrica

La creciente carga y multiplicidad de dispositivos conectados al microordenador probablemente exijan una fuente de alimentación eléctrica significativamente mayor que las que se utilizan en la actualidad. Incorporará circuitos estabilizadores y el apoyo de pilas de reserva recargables, de modo que las fluctuaciones o los cortes de potencia de la red eléctrica no ocasionen la pérdida o el deterioro de los datos

8 Pantalla portátil

La tecnología de pantalla plana (que probablemente utilice una matriz de cristal líquido quizá conectada al procesador central mediante una conexión de infrarrojos o incluso de microondas) se podría emplear para visualizar texto y material gráfico. Si este dispositivo fuera, además, sensible al tacto, podría ser a la vez tablero para selección de menús y digitalizador

9 CDROM

La *Compact Disk ROM* (ROM en disco compacto), que utiliza un rayo láser para leer información codificada ópticamente, tiene grandes posibilidades de reemplazar a los cartuchos de ROM convencionales; debido a su capacidad: una CDROM típica retendría cuatro megabytes

10 Diskettes flexibles

Para finales de la década, los diskettes flexibles habrían de alcanzar un desarrollo como para competir con los discos Winchester, tanto en velocidad como en densidades de almacenamiento de datos. Al mismo tiempo, su diámetro se habría de reducir a menos del mínimo corriente de 3 pulgadas

11 Panel frontal

Antes del advenimiento de los lenguajes de alto nivel y de los teclados, se había de dar entrada a los programas en notación binaria mediante el panel frontal: una línea de luces e interruptores que le proporcionaban al usuario el control sobre cada bit de los buses de direcciones, de datos y de control. Para los entusiastas del código de lenguaje máquina, el panel frontal aún puede ser una herramienta útil, de modo que esta idea podría resurgir en los ordenadores personales futuros

12 Ratones infrarrojos

El IBM PC-Junior ya utiliza la radiación infrarroja para transferir los datos del teclado al ordenador sin ninguna conexión por cable. Esta tecnología podría proporcionar la interconexión entre todos los periféricos, incluyendo los ratones, eliminando por consiguiente el "efecto spaghetti". Existirán, por supuesto, modelos tanto para diestros como para zurdos

13 Microprocesadores

Los primeros ordenadores personales basados en microprocesadores de 32 bits salieron al mercado en 1983, pero no tuvieron más remedio que depender de buses de datos de 16 bits, e incluso de 8, para mantener la compatibilidad con los chips de memoria y los periféricos existentes, y no dieron el potencial que prometían. Con la introducción de dispositivos como el chip Motorola 68032, que ofrece un procesamiento de 32 bits y una transferencia de datos de 32 bits, las capacidades de estos procesadores de gran capacidad se generalizarán. Muchos miniordenadores de alto precio poseen ya procesadores de 32 bits

Pugna generacional

Con la introducción de la tecnología VLSI, estamos actualmente a punto de entrar en la cuarta generación de ordenadores. Pero los japoneses anuncian ya una quinta generación

Se dice que los ancianos no hacen revoluciones, y el director del proyecto japonés para la creación de la quinta generación de ordenadores parece haberse tomado al pie de la letra estas palabras. Al seleccionar 40 científicos de las diez mayores corporaciones y de los laboratorios del gobierno para que trabajaran con él en el Instituto para la Tecnología de la Nueva Generación de Ordenadores, en Tokio, el doctor Kazuhiro Fuchi los ha escogido entre aquellos que tenían menos de 35 años de edad. El Instituto se fundó el 14 de abril de 1982 con un presupuesto de alrededor de 70 000 millones de pesetas (a gastar en diez años) y es una empresa de cooperación entre el gobierno y la industria. Empresas como Fujitsu, Sharp y Toshiba participan en este ambicioso proyecto, que tiene como objeto dar un salto respecto al estado actual de la tecnología de ordenadores.

La acuñación del término *quinta generación* centra la atención en los principales avances del pasado en el campo del diseño de ordenadores, estimulando fantásticas posibilidades para el futuro. La primera generación de ordenadores se caracterizó por la utilización de válvulas termoiónicas, pronto envejecidas con la invención del transistor. Los ordenadores de transistores de la segunda generación fueron, a su vez, superados por las máquinas que utilizan la tecnología de integración a gran escala (LSI: *Large Scale Integration*), consistente en incorporar muchos transistores en un solo chip. Ahora mismo estamos ante el final de esta tercera generación de ordenadores, pero los últimos años de la década de los ochenta deberían ver la aparición en el mercado de los chips VLSI (*Very Large Scale Integration*: integración a escala muy grande) de la cuarta generación. Estos chips tendrán hasta diez millones de transistores por chip, frente al límite actual de aproximadamente un cuarto de millón.

En la actualidad, la IBM invierte al año más de 230 billones de pesetas en investigación y desarrollo de ordenadores, cifra que convierte a nuestros ojos la inversión japonesa en algo insignificante. Pero el desembolso del capital japonés no obedece sólo a razones estrictamente lucrativas.

La ciencia ha desplazado durante los últimos cien años su foco de interés desde el aprovechamiento de la energía en bruto (amplio abanico que va de la electricidad al motor de combustión interna) hasta el estudio de la forma de riqueza más intangible: la información. La tierra, el trabajo, el capital y la industria pueden haber sido en el pasado las fuentes de poder, pero el futuro favorecerá a aquéllas relativas al control de la información. El conocimiento y el procesamiento de la información serán las claves de la sociedad postindustrial. De modo que lo que se necesita para esta nueva sociedad es un ingenio, una máquina con razonamiento automático que se pueda aplicar a cualquier problema concreto o campo del esfuerzo humano con la precisión matemática y la certeza típicas de un ordenador. El ingenio que están construyendo actualmente los japoneses se denomina KIPS (*Knowledge and Information Processing System*: sistema de procesamiento de la información y el conocimiento).

Los seres humanos son muy buenos en cuanto a convertir señales sensoriales en formas cognoscitivas (la situación en una partida de ajedrez se puede colegir con una mirada), pero cuando se trata de tomar decisiones que dependen de grandes cantidades de datos, nuestras limitaciones quedan enseguida de manifiesto. Las reglas del ajedrez se pueden explicar en unos pocos minutos, y, sin embargo, el juego es tan complicado que los grandes maestros sólo son conscientes de una docena de posibles movimientos. No obstante, en principio todo problema al cual se le aplica el raciocinio se puede dividir en una serie de pasos simples, cada uno de los cuales se puede decidir mediante la aplicación de reglas deductivas. Este conjunto de reglas se conoce como lógica de predicados. Las reglas deductivas de la lógica se aplican a todos los problemas, pero cuando tomamos decisiones cotidianas sencillas no somos conscientes de ellas.

Lenguaje lógico

El PROLOG (abreviatura de *Programming Logic*: lógica de programación) se desarrolló a principios de los años 70 en el seminario de inteligencia artificial de la Universidad de Marsella, si bien uno de los principales protagonistas de su desarrollo y promoción es el norteamericano Robert Kowalski, del Imperial College de Londres. Basado en algunos de los principios de la lógica humana, es el lenguaje que tiene más probabilidades de utilizarse en los ordenadores de la quinta generación. Es también muy útil para crear y tratar bases de datos y para fines educativos

Un experto necesita algo más que un buen cerebro. Pongamos por caso un doctor, que necesita muchos años de entrenamiento para acumular su conocimiento médico. De la misma manera, un KIPS debe poseer un banco de datos sobre el cual puedan operar las reglas de inferencia. Además, el sistema ha de ser sumamente amable con el usuario si se desea que el KIPS no exija su propia cantera de expertos para operarlo. Una máquina KIPS con la cual uno pueda mantener una conversación en el lenguaje que se elija debe ser producto de la investigación en el campo de la inteligencia artificial, que es un área de estudio muy controvertida. Por consiguiente, los objetivos que los japoneses se han fijado a sí mismos abarcan una amplia gama de la ciencia del ordenador: hardware, software, interfaces, sistemas especializados (véase p. 72) y los problemas de la inteligencia artificial.

El proyecto japonés se ha concebido para ir más allá de los adelantos en la tecnología del chip. A medida que aumenta la densidad de transistores de

los circuitos integrados, menor es la distancia que han de recorrer los electrones entre un componente y otro y, en consecuencia, los circuitos operarán con mayor rapidez. Pero no es la velocidad en sí lo que importa a los japoneses, y por eso sus esfuerzos se centran en el software. En una partida de ajedrez, por ejemplo, son tantas las posibles secuencias de movimientos (alrededor de 10^{120}) que se calcula que el tiempo necesario a la mente humana para explorar todas las posibilidades se aproxima al que le queda de vida a nuestro Sol. Uno de los objetivos del proyecto es la producción de la máquina que pueda realizar 100 millones de inferencias lógicas (es decir, que pueda aplicar 100 millones de reglas) por segundo. En este sentido se dice 100 millones de LIPS (*Logical Inferences Per Second*: inferencias lógicas por segundo).

Otra manera de aumentar la velocidad sería cincelando las funciones de software en el diseño del chip, en vez de cargarlas en la memoria y procesarlas por medio de un chip de finalidad universal. Este intento de superar la distinción entre hardware y software es una de las metas más interesantes del proyecto. Ya existen memorias "asociativas" que tienen circuitos de búsqueda lógica incorporados en las celdas de memoria. Estos dispositivos pueden posicionar un dato exclusivamente a partir del significado del propio dato, sin necesidad de especificar una dirección de memoria.

Los avances de este tipo producirán un interesante maridaje entre los procesadores lógicos y los bancos de datos. Cincelar circuitos con rutinas de programación en un procesador recuerda los primeros ordenadores como el ENIAC (véase p. 140), pero las máquinas de la quinta generación se diferenciarán de la arquitectura de Von Neumann en un aspecto fundamental: incorporarán muchos procesadores distintos que funcionarán todos simultáneamente (en paralelo), en lugar de una sola unidad central de proceso. Esto exigirá especial esmero en cuanto a la sincronización y el control de las operaciones internas, pero eliminará esa merma de velocidad que impone la ejecución secuencial de las instrucciones. El lenguaje interno escogido para el KIPS es el PROLOG, un lenguaje que se desarrolló en Francia y Gran Bretaña y que se basa en la lógica de predicados. Pero el KIPS tendrá la capacidad de comunicarse con sus usuarios en muchas lenguas.

La traducción del habla natural humana es otro de los objetivos del proyecto, con el directo propósito de un 95 % de precisión. La capacidad para reconocer palabras individuales pronunciadas por diferentes interlocutores va muy a la zaga comparada con el éxito de la voz sintética. No obstante, la NEC Corporation, de Japón, ya ha creado una máquina capaz de reconocer el habla normal.

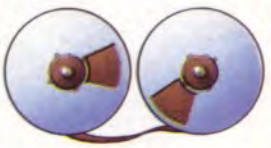
Con respecto a la palabra escrita, el proyecto tiene en preparación un diccionario y un programa japonés-inglés de 100 000 palabras, y se espera que permita traducir con una precisión del 90 %.

Japón ya tiene precedentes en cuanto al éxito de proyectos de investigación a largo plazo: el proyecto PIPS (*Pattern Information Processing Systems*) de la década de los setenta está resultando muy útil para el desarrollo de bancos de datos visuales e interfaces de fácil manejo para el usuario. Un KIPS podrá ser capaz de mirar una imagen y extraer sus contornos y sus características más destacadas, para hacerse una idea preliminar. En el metro de Tokio ya existe una máquina que explora los corredores con una cámara de video y presenta un cuadro del flujo de pasajeros en las diferentes líneas.

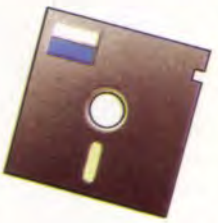
En Estados Unidos, la tecnología de la información representó en 1983 un movimiento por valor de 88 billones de dólares. Con una perspectiva decreciente del nivel de empleo en la industria fabril semejante al de la agricultura a principios de este siglo (del 40 % de la mano de obra empleada en agricultura al comienzo del siglo se pasó al 3 % en la actualidad), la sociedad se ha visto abocada cada vez más a una cultura de la información. Bajo este prisma, Japón está intentando algo muy ambicioso con su proyecto para la quinta generación. El plan es optimista y discurre por unos derroteros "preseleccionados" que tendrán o no viabilidad (al fin y al cabo, todavía esperamos transitar por la tan anhelada senda de la fusión nuclear controlada). Pero se trata de un enfoque positivo por parte de un país eminentemente exportador, muy similar a cualquier estado occidental avanzado. Sin embargo, mientras que, por ejemplo, en Gran Bretaña, la inversión en investigación ha disminuido en la última década, Japón sigue especulando con su futuro.

Generaciones en liza

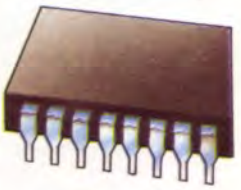
Primer asalto
La primera generación de ordenadores electrónicos se desarrolló alrededor de la tecnología de la válvula termoiónica. Con muy poca memoria interna, los datos generalmente se almacenaban en tarjetas perforadas



Segundo asalto
La segunda generación se desarrolló a partir del transistor, que aumentó la capacidad de memoria, aunque aún se utilizaba el almacenamiento externo (en forma de cinta magnética)



Tercer asalto
La invención del circuito integrado incrementó sobremanera la potencia del ordenador y permitió el nacimiento del microordenador, caracterizado por la unidad de disco flexible



Cuarto asalto
Ahora estamos pasando de la tercera generación a la cuarta, que se basará en la tecnología del chip VLSI. La memoria RAM será tan grande que el almacenamiento externo tendrá cada vez menos importancia

Quinto asalto
La quinta generación de ordenadores, que se está desarrollando fundamentalmente en Japón, en realidad atañe al software y no al hardware. No obstante, se basa en el supuesto de que la memoria para el usuario será tan grande que el tamaño del programa dejará de ser un factor digno de consideración

Kevin Jones

Research Machines 380Z

Un sistema físico sólido y unos soberbios gráficos en alta resolución han popularizado este microordenador en las escuelas y entre los militares

Los productos de Research Machines Limited se encuentran entre los modelos de ordenadores más duraderos que existen. Aunque estas máquinas no son especialmente innovadoras ni su precio es competitivo, están diseñadas y construidas con suma solidez, están bien respaldadas y son extraordinariamente fiables. El ordenador más popular de la empresa, el RML 380Z, puede que falte en muchos hogares, pero al ser una de las máquinas más solicitadas por los pedagogos, muchos niños realizaron con él sus primeras experiencias de informática.

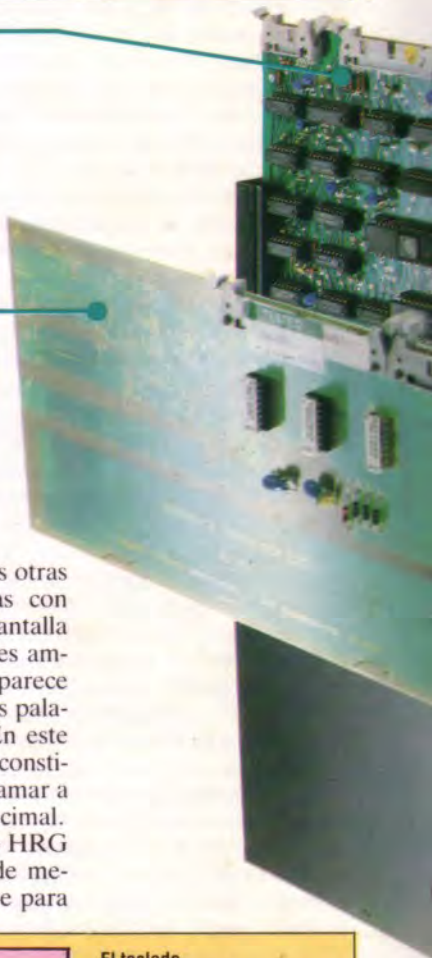
Comparado con la mayoría de las máquinas, el 380Z es enorme: los tableros del circuito principal están alojados en una sólida carcasa de 48 cm de ancho, con asas a los lados. Al quitar la tapa de esta "caja negra" descubrimos por qué es tan grande, ya que alrededor de la cuarta parte del espacio interior está ocupado por la fuente de alimentación eléctrica. Forrada de metal, su peso denota que no es una unidad de alimentación eléctrica avanzada de tipo conmutación, sino un sólido transformador de núcleo de hierro con condensadores. Puede parecer anticuado, pero tiene la ventaja de que es casi imposible sobrecargarlo o dañarlo.

Quizá sea esta fiabilidad lo que haya hecho tan popular el 380Z en el Ministerio de Defensa británico, que está empleando gran número de estas máquinas para el control de existencias y funciones similares. En aquellas escuelas y facultades que imparten matemáticas, física y ciencias a nivel más elevado, la máquina se ve especialmente favorecida en virtud de sus gráficos de alta resolución, que resultan de gran utilidad para demostraciones ilustradas de diversos puntos de los temarios de estudios.

El paquete de gráficos de alta resolución (HRG: *High Resolution Graphics*) es un conjunto de rutinas en código de lenguaje máquina que se llaman desde el programa del usuario y que pueden modificar la visualización generada por la ficha HRG. Ésta es imprescindible para producir cualquier gráfico; y, a pesar de que se introdujo hace algunos años, sigue siendo uno de los mejores sistemas existentes. Mediante la alteración del contenido de ciertas posiciones de memoria, la ficha puede generar visualizaciones en varias resoluciones con los colores normales (rojo, amarillo, verde, azul, magenta y cian). Según la resolución que se elija, que puede oscilar entre 160 × 96 y 320 × 192, a estos seis colores más el blanco se les pueden dar distintos niveles de brillo, aumentando de esta manera la gama de colores a 1 786 (siete veces 255, más el negro). Por otra parte, se pueden utilizar algunos de los bits normalmente destinados a especificar la densidad para producir múltiples páginas de gráficos, si bien la cantidad de intensidades diferentes será proporcionalmente inferior.

El monitor
Para aprovechar al máximo las facilidades del 380Z es esencial un monitor en color con una interface RGB. La máquina se puede comprar con 40 u 80 columnas en pantalla como estándar, lo que determina el tipo de monitor necesario

Chris Stevens



Tablero para control de la unidad de disco
Además de llevar un chip especializado para control de disco, este tablero posee un chip reloj-sincronizador Z80 (CTC) y un chip de input/output en serie 8521, que juntos proporcionan óptimas facilidades para comunicaciones

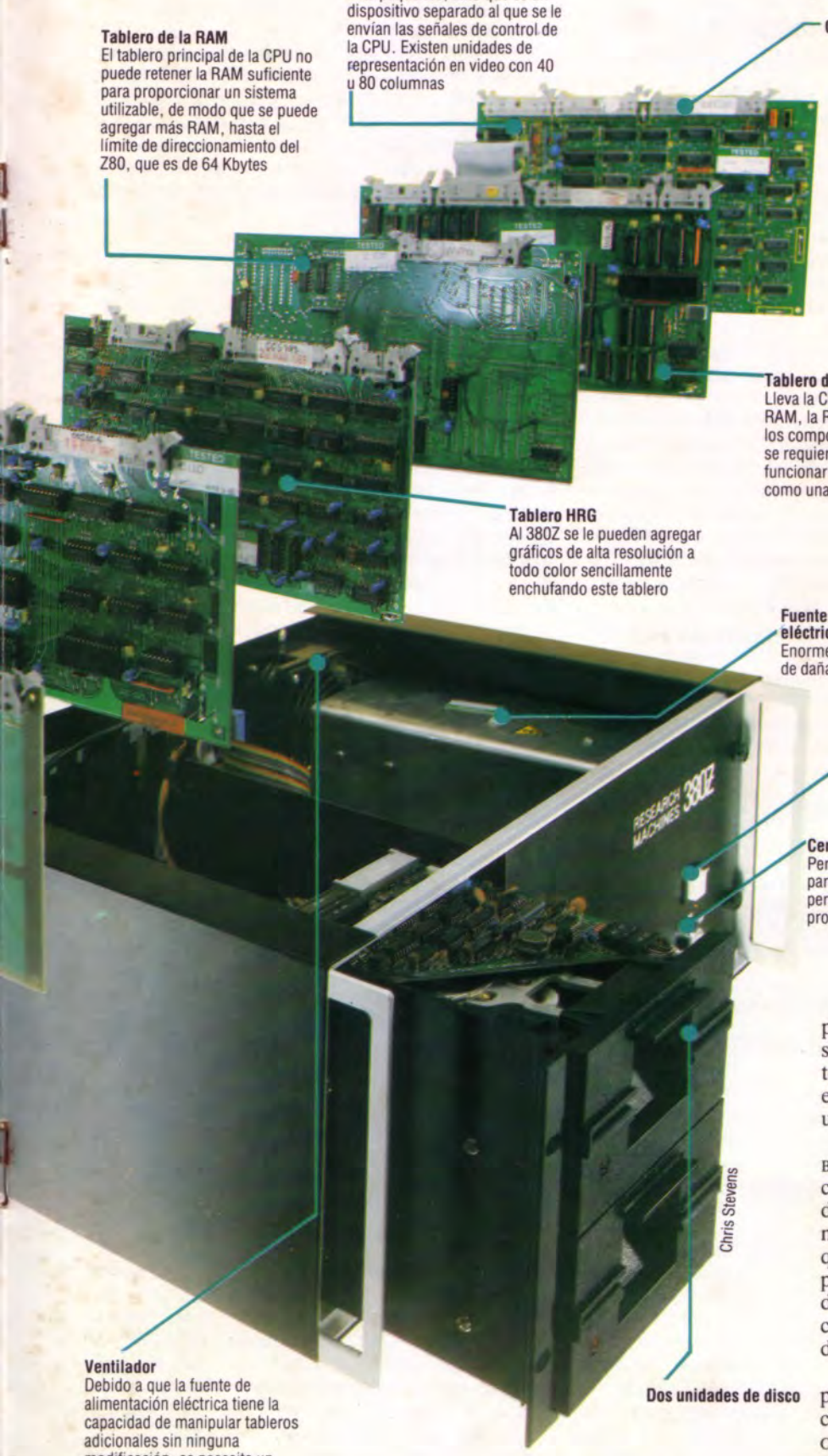
Tablero terminador de bus
Está situado en el extremo del bus más alejado del tablero de la CPU, y protege contra las interferencias a las diversas líneas eléctricas

Un juego completo de estas llamadas, más otras para manipular las cuestiones relacionadas con ellas, como *dump* de impresora (copiar la pantalla en el papel), se proporcionan como versiones ampliadas del BASIC RML. Esta versión se parece mucho al BASIC Microsoft y la mayoría de las palabras clave se emplean de forma idéntica. En este sentido, la única excepción importante la constituye el empleo de etiquetas con texto para llamar a las subrutinas, en vez de una dirección en decimal.

Sin embargo, el intérprete más el paquete HRG ocupan los dos una considerable cantidad de memoria, y puede que no dejen lugar suficiente para

El teclado
El teclado que se proporciona con el RML 380Z está montado en una caja metálica pequeña pero pesada. Las teclas se disponen de acuerdo a un patrón bastante estándar y son de gran calidad, con un tacto sólido pero agradablemente ligero. Es obvio que están diseñadas para resistir un uso intensivo, factor este que hace que el teclado sea ideal para la escuela

Chris Stevens



Tablero de la VDU
El generador de visualizaciones no forma parte del tablero principal, como en las máquinas más pequeñas, sino que es un dispositivo separado al que se le envían las señales de control de la CPU. Existen unidades de representación en video con 40 u 80 columnas

Tablero de la RAM
El tablero principal de la CPU no puede retener la RAM suficiente para proporcionar un sistema utilizable, de modo que se puede agregar más RAM, hasta el límite de direccionamiento del Z80, que es de 64 Kbytes

Conexiones del bus Z80

Tablero de la CPU
Lleva la CPU Z80, parte de la RAM, la ROM y la mayoría de los componentes esenciales que se requieren para hacer funcionar la máquina, tales como una puerta para el teclado

Tablero HRG
Al 380Z se le pueden agregar gráficos de alta resolución a todo color sencillamente enchufando este tablero

Fuente de alimentación eléctrica
Enorme, pesada y muy difícil de dañar

Mando de borrado

Cerradura
Permite cerrar la máquina para impedir que cualquier persona pueda interferir programas esenciales

Ventilador
Debido a que la fuente de alimentación eléctrica tiene la capacidad de manipular tableros adicionales sin ninguna modificación, se necesita un ventilador para ayudar a la refrigeración

Dos unidades de disco

Chris Stevens

Research Machines 380Z

DIMENSIONES
595 x 425 x 215 mm

CPU
Z80

VELOCIDAD DEL RELOJ
4 MHz

MEMORIA
Hasta 6 Kbytes de ROM
56 Kbytes de RAM

VISUALIZACION EN VIDEO
24 líneas de 40 u 80 caracteres, 7 colores con hasta 255 tonos. Resolución para gráiicos de 320 x 192 y 160 x 96

INTERFACES
RS232 en serie, cassette, impresora en paralelo

LENGUAJE SUMINISTRADO
BASIC Research Machines ampliado

OTROS LENGUAJES DISPONIBLES
ALGOL, FORTRAN y CP/M estándar

VIENE CON
Manuales de instalación, CP/M, sistema de disco, sistema de cassette y programas de utilidades BASIC en disco

TECLADO
60 teclas de calidad de procesador de textos

DOCUMENTACION
Excelente, aunque algo árida. La información es exhaustiva y resulta de fácil acceso para el usuario

programas de usuario sofisticados. Por esta causa se proporcionan tres versiones de BASIC, incluyendo todo el paquete HRG, parte del mismo o bien nada en absoluto, según las necesidades de memoria del usuario.

El empleo del paquete HRG no se limita al BASIC, pues funciona como un sencillo archivo en código máquina, al que se puede acceder a través de cualquier lenguaje. Dado que el 380Z por lo general opera bajo CP/M (véase p. 410), ello significa que la gama de lenguajes disponibles es amplia, pero el hecho de que la máquina posea una versión de ALGOL configurada para funcionar con ella, la convierte en algo insólito en el mundo del microordenador.

Este lenguaje, que muchos científicos europeos prefieren al FORTRAN (el lenguaje favorito para fines científicos en Estados Unidos), se asemeja al PASCAL y es especialmente eficaz en cálculos matemáticos complejos, como los que exige el diseño estructural. Éste es otro factor que hace que la máquina les resulte atractiva a los educadores.

Sonido ideal

El BASIC del Commodore 64 no responde al notable nivel de sus facilidades para sonido

Entre los ordenadores personales más populares, es el Commodore 64 el mejor equipado para la producción de sonido, y esto gracias a un chip especial denominado *Sound interface device* (dispositivo interface de sonido) o SID, que son sus siglas de brega.

El SID tiene unas posibilidades similares a las de un sintetizador monofónico comercial. Hay tres osciladores con una amplitud de ocho octavas (de 0 a 3900 Hz en 65 536 intervalos); un control principal de volumen, de 0 a 15; cuatro formas de onda, o timbres, para cada oscilador (triangular, aserrada, pulso de anchura variable y ruido); sincronización de osciladores y generadores de envolventes susceptibles de un control ADSR para cada oscilador. Otras configuraciones son: modulación circular; filtro programable con pasada baja, pasada de banda, pasada alta, de salida escalonada (que bloquea una banda estrecha de frecuencias) y resonancia variable; filtración de envolvente; dos interfaces para potenciómetro analógico-digital que se pueden emplear para controlar las facilidades del SID, y una salida de audio externa, que permite conectar entre sí chips SID adicionales. Con las salidas de SID estándar se puede dar entrada, filtrar y mezclar a otras señales de audio.

Sería imposible detallar el funcionamiento de cada una de estas configuraciones (existen libros muy buenos sobre la materia), pero lo que sí podemos hacer es explicar el significado de todos estos términos. En primer lugar, la sincronización de osciladores hace que dos señales (en este caso, dos voces especificadas) se acoplen entre sí armónicamente, produciendo a partir de las dos señales separadas un solo tono más complejo.

La modulación es la modificación de una señal por otra, que afecta o bien la frecuencia o la amplitud (volumen) del sonido. La modulación circular es la modulación de amplitud de una voz con otra, lo cual produce un tono que es claro pero que posee un efecto disonante y chillón; se puede utilizar para producir sonidos del tipo campana similares a los producidos por un bidón vacío. Se dice que estos sonidos poseen sobretonos inarmónicos.

Los filtros permiten eliminar de una señal escalas de frecuencia especificadas. Los diferentes tipos de filtración posibles en el Commodore 64 tienen los efectos que sugieren sus nombres: los filtros de pasada baja descartan las frecuencias que sean mayores que una frecuencia especificada; los filtros de pasada de banda eliminan las frecuencias por arriba y por debajo de una «banda» de frecuencias especificada; los filtros de salida escalonada son el opuesto de los filtros de pasada de banda: bloquean una

Luz espacial

Los gráficos Player-Missile son uno de los puntos fuertes de las máquinas Atari

Los gráficos Player-Missile

Los gráficos *Player-Missile* (jugador-misil) o "PM" constituyen un aspecto muy importante de las capacidades para gráficos de Atari. Su naturaleza es similar a la de los gráficos sprite que existen para el Commodore 64 (véase p. 408) y el Sord M5, permitiendo al programador el diseño y control de hasta ocho figuras diferentes en alta resolución. Estas figuras móviles operan con independencia de la visualización del fondo y se las puede programar para que se desplacen por delante o por detrás de otras figuras que pueden estar dibujadas en la pantalla. Ello le permite al programador incorporar una tercera dimensión en los efectos de pantalla. Los gráficos PM se pueden mover suave y velozmente por la pantalla, y por ello son ideales para los juegos recreativos de acción rápida. También son útiles para crear visualizaciones estáticas más coloridas que las que se consiguen empleando las modalidades normales para gráficos, porque los objetos PM se pueden colorear por separado e independientemente de la visualización del fondo.

Al igual que con todos los gráficos sprite, el secreto de las facilidades para los gráficos PM radica en un hardware *ad hoc*. Hay registros especiales diseñados para controlar el movimiento, el color y la visualización en pantalla de los objetos PM. Todo cuanto el programador debe hacer es colocar ciertos valores en estos registros para manipular los objetos. En BASIC, esto se realiza mediante la orden POKE. Una vez que se ha colocado (POKE) un número en el registro adecuado, el propio hardware de Atari se hace cargo del resto del trabajo. Esto se efectúa a la velocidad del código de lenguaje máquina y, por consiguiente, mucho más rápido que si el proceso se controlara mediante BASIC.

Analicemos ahora el proceso para crear objetos PM y los registros para controlarlos. Los jugadores se diseñan a partir de una franja vertical, de ocho pixels de ancho y 128 o 256 de alto. Cada una de las filas de la franja se representa en la memoria del ordenador como un único byte. Mediante la colocación (POKE) de los códigos binarios adecuados, se puede definir la forma de un jugador utilizando un método similar al que se emplea para crear caracte-

```
10 SID = 54272
20 POKESID + 23,0
30 POKESID + 24,15
40 POKESID + 5,40
50 POKESID + 6,201
60 FOR N = 1T05
70 READ FH,FL,D
80 POKESID + 1,FH:
   POKESID,FL:
   REM*TOCAR
   NOTA*
90 POKESID + 4,33
100 FORI = 1T0300*D:
    NEXT I
110 POKESID + 4,32
120 FORI = 1T0100:
    NEXT I
130 NEXT N
140 FORI = 1T02000:
    NEXT I
150 POKESID + 24,0
160 REM**FH FL D**
170 DATA 57,172,1
180 DATA 64,188,1
190 DATA 51,97,1
200 DATA 25,177,1
210 DATA 38,126,2
220 END
```

banda específica; los filtros de pasada alta cortan las frecuencias que sean inferiores a una especificada; y la resonancia variable se puede aplicar a todos los filtros anteriores para enfatizar las frecuencias alrededor de los puntos de corte. La filtración de envolvente constituye un caso especial: su efecto difiere de las otras en que los valores digitalizados de ADSR establecidos para la envolvente 3 se pueden leer desde el chip SID y aplicar a una señal, de modo tal que la estructura armónica se vaya modificando en el transcurso de una nota. Funciona como un filtro variable.

Estas múltiples configuraciones permiten construir, con sonidos sumamente complejos, interesantes efectos y convincentes imitaciones de instrumentos convencionales. La faceta desalentadora del SID es que el BASIC CBM V2, la versión con que viene el 64, no dispone en absoluto de ninguna orden directa de sonido. Éste se obtiene utilizando PEEK y POKE para los 29 registros de control del SID. Por consiguiente, se requiere muchísima codificación para generar hasta los efectos más sencillos y, en algunos casos, el BASIC no es lo suficientemente veloz como para hacer justicia a la gama completa de posibilidades del SID.

No podemos describir a fondo los registros de control del SID, pero, con lo dicho, puede usted tocar notas agradables, con programas como el de la izquierda.

Aunque la longitud del programa es de 22 líneas, sólo toca cinco notas de una melodía sencilla en un oscilador. La línea 20 desconecta el filtro de los osciladores; la línea 30 establece en el máximo el volumen principal; y las líneas 40 y 50 especifican una envolvente similar al piano. La línea 80 establece la frecuencia de las notas; la 90 y la 100 empiezan y terminan el ciclo ADSR y seleccionan una onda aserrada para la voz 1; y la sincronización para los bucles FOR...NEXT se obtiene mediante las líneas 100, 120 y 140.

Programar sonido en BASIC en un Commodore 64 requiere gran esfuerzo tanto para aprender el código como para escribirlo. Incluso puede ser un ejercicio desmoralizador, porque la única manera de descubrir si un conjunto de sentencias, ya de por sí complicadas, funciona a un compás aceptable, es mediante el ensayo y el error. Si usted desea métodos más sencillos para generación de sonido, vale la pena investigar los muchos programas para edición de sonido que existen a la venta en el mercado. Éstos suelen estar escritos en lenguaje máquina, y aprovechan al máximo las maravillosas configuraciones del Commodore 64.

res definidos por el usuario (véase p. 246). De esta manera se pueden definir hasta cuatro jugadores, ocupando cada uno de ellos sus 256 o 128 bytes de memoria.

Cada uno de los cuatro jugadores posee una figura de misil relacionada con él cuya anchura es de dos bits. Para crear jugadores y misiles es necesario colocar (POKE) los patrones de bits que definen su forma en una zona determinada de la memoria. La zona de RAM utilizada la puede elegir el programador, pero se le debe informar de ello al ordenador estableciendo un indicador al principio de la zona.

Si el programador opta por emplear una resolución vertical de un solo pixel, entonces se necesita el doble de la memoria requerida para una resolución vertical de dos pixels. El siguiente programa diseña al jugador 0 en resolución vertical de dos pixels como una nave espacial:

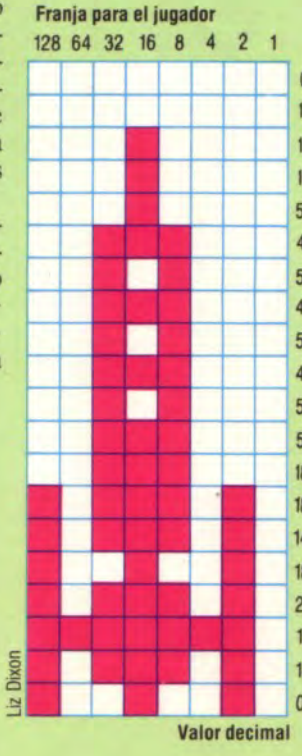
```
10 REM ***DEFINIR UN JUGADOR***
20 P = PEEK(106) -8: REM ESTABLECE EL P EN
   2K POR DEBAJO DE LA PARTE SUPERIOR DE
   RAM
30 POKE 54279,P: REM ESTABLECE PUNTERO EN
   ZONA PM
40 BASE = 256*P:REM ESTABLECE LA DIRECCION
   DE BASE ZONA PM
50 FOR I = BASE + 512 TO BASE + 640
60 POKE I,0: REM LIMPIA ZONA JUGADOR 0
70 NEXT I
80 FOR I = BASE + 512 + 50 TO
   BASE + 530 + 50
90 READ A: POKE I,A: REM DEFINIR FIGURA
100 NEXT I
110 DATA 16,16,16,56,40,56,40,56,40
120 DATA 56,56,186,186,146,186,254,186,146
```

Cada figura de jugador tiene varios registros relacionados con ella. Estos registros controlan el color, la posición horizontal y el tamaño. El último permite que el programador multiplique el ancho de un jugador por el factor dos o el cuatro. Otros registros controlan la prioridad del jugador con respecto al fondo. Los misiles toman el color del jugador que los guía, pero su tamaño se puede modificar independientemente. Para las aplicaciones de juegos se dispone de una serie de registros que detectan las colisiones en pantalla entre los jugadores, los misiles y el fondo. No obstante, no hay registro de posición vertical ni para misiles ni para jugadores. El movimiento vertical de un jugador se obtiene elevando, a través de la zona de la memoria reservada a ese jugador, el contenido de cada una de las posiciones que guardan los patrones de bits para la figura. En lenguaje ensamblador esta tarea es bastante directa, pero en BASIC es lenta.

Los gráficos Player-Missile amplían considerablemente el potencial para gráficos de Atari, aunque no son tan versátiles ni fáciles de utilizar como los sprites del Commodore 64. Ofrecemos una continuación del programa que empezamos anteriormente, para colorear la nave espacial y desplazarla de izquierda a derecha a través de la pantalla.

```
130 POKE 559,46: REM VISUALIZA LINEA 2 PM
140 POKE 53277,3: REM VISUALIZA PM
150 POKE 704,88: REM COLOREA ROSA
    JUGADOR 0
160 GRAPHICS 0
170 SETCOLOUR 2,8,2: REM ESTABLECE FONDO
    AZUL OSCURO
180 FOR I = 0 TO 320
190 POKE 53248,I: REM ESTABLECE POSICION
    HORIZONTAL
200 NEXT I
210 END
```

Cohete PM
Antes de definir una pieza de juego hay que dibujarla, calculando después los valores decimales para cada una de las filas de pixels



Laboratorio de idiomas

Para resumir lo hasta aquí explicado, analicemos con ojo crítico el lenguaje BASIC y algunas de las alternativas al mismo

Vamos a analizar brevemente los puntos fuertes y los puntos débiles del BASIC respecto a los otros lenguajes de programación.

El BASIC deriva del FORTRAN, uno de los primeros lenguajes de programación. A diferencia de la mayoría de los otros lenguajes, el BASIC se interpreta. Esto significa que cuando se ejecuta un programa otro programa especial, situado en algún lugar de la memoria del ordenador, capta el código sentencia a sentencia, y lo convierte en código de lenguaje máquina. He aquí lo que sucedería con un breve programa en BASIC como éste:

```
10 CLS
20 PRINT "DIGITE UN NUMERO"
30 INPUT X
40 PRINT "DIGITE UN SEGUNDO NUMERO"
50 INPUT Y
60 "PRINT" "El PRODUCTO DE LOS DOS NUMEROS
   ES: ";
70 PRINT X*Y
80 PRINT
90 PRINT "¿DESEA OTRA PRUEBA?"
100 PRINT "PULSE 'S' PARA PROBAR OTRA VEZ"
110 PRINT "O BIEN 'N' PARA TERMINAR"
120 FOR X = 1 TO 1
130 LET A$ = INKEY$
140 IF A$ <> "S" AND A$ <> "N" THEN X = 0
150 NEXT X
160 IF A$ = "S" THEN GOTO 10
170 END
```

Cuando el intérprete de BASIC se encuentra con la línea 10, elabora el código de lenguaje máquina necesario para limpiar la pantalla. En la línea 20 prepara en código de lenguaje máquina las instrucciones necesarias para enviar a la pantalla el mensaje DIGITE UN NUMERO. En la línea 30, determina el espacio de memoria necesario para almacenar un número real, espera la entrada desde el teclado y después convierte el número digitado a binario para almacenarlo en el espacio destinado a la variable X. Todo esto se repetiría para las líneas desde la 40 a la 60. Si el usuario deseara repetir el programa, digitando S, el intérprete retrocedería hasta la línea 10, y repetiría otra vez todos los cálculos.

Por el contrario, la mayoría de los demás lenguajes se "compilan". Ello significa que después de que se ha escrito un programa, para que se lo pueda ejecutar, antes lo debe procesar un *compilador*. El compilador es un programa independiente que examina detenidamente el "código fuente" (el programa original) y produce una segunda versión del mismo en código de lenguaje máquina. Cuando se ejecuta el programa compilado, es lógico que funcione mucho más rápido que un programa interpretado, porque ya se han efectuado todas las traducciones a lenguaje máquina, que requieren bastante tiempo.

Si los programas compilados son más rápidos que los interpretados, puede que se pregunte por qué razón no todos los lenguajes de programación utilizan compiladores. Varias son las ventajas que ofrecen los programas interpretados, como el BASIC. La mayoría de éstas derivan del hecho de ser un lenguaje interactivo, es decir, que se puede verificar y depurar "frente al teclado", mientras se está desarrollando el programa. El BASIC, por ejemplo, admite la inserción de la orden STOP en cualquier punto del programa. En el momento en que el intérprete la encuentra, deja inmediatamente de interpretar el programa y permite que se impartan "órdenes" desde el teclado.

Las órdenes son instrucciones que el intérprete puede ejecutar directamente cuando no se está ejecutando el programa. El BASIC posee una gran cantidad de órdenes, y éstas pueden ser de enorme valor para la depuración. Después de que un programa en BASIC se ha ejecutado (es decir, después de que el intérprete se ha encontrado con la sentencia END), o cuando el intérprete se encuentra con una sentencia STOP, se pueden imprimir (PRINT) los valores de todas las variables. Intente ejecutar el programa de la agenda de direcciones, por ejemplo. Ejecute el programa y digite 9 para salir del mismo. Si funciona de principio a fin sin que aparezca ningún mensaje de error, debería acabar con una muletilla típica (por lo general, OK, > o *). Luego, digite PRINT RMOD <CR>. El intérprete debe imprimir un 0 en la pantalla (¡siempre y cuando no se hubiera agregado ningún registro!). Luego pruebe PRINT TAMA <CR>. La consecuencia será que el intérprete imprimirá en la pantalla un número superior en una unidad al número de registro que usted tenga en el archivo de datos.

Ventajas del BASIC

Se suele decir que el BASIC es el lenguaje ideal para el programador sin experiencia, pues permite eliminar los *bugs* desde el teclado. Posee otra importante ventaja: es comparativamente fácil de aprender. Por ejemplo, en esta veintena de capítulos dedicados a la programación BASIC hemos tocado todos los puntos fundamentales y muchos de los aspectos avanzados del BASIC en apenas 86 páginas. Los errores sintácticos tales como 40 PRINT A(12) provocan por lo general mensajes de error de fácil comprensión al ejecutar el programa, como SYNTAX ERROR IN 40. Con sólo una mirada al número de línea aludido adivinamos dónde está el error, y con la misma facilidad se elimina mediante EDIT 40 <CR> (seguido de unas pocas órdenes de edición) o volviendo a digitar la línea del modo correcto. Siempre que el intérprete de BASIC se encuentra con un error de sintaxis o de lógica, interrumpe la ejecución del programa e informa del error. Eliminar

los *bugs* es tan sencillo como probar con una nueva línea en sustitución de la equivocada y digitar RUN <CR> otra vez.

Desventajas del BASIC

Algunas desventajas son muy sutiles, pero las hay flagrantes. Dado que se interpreta (hay sólo unas pocas versiones compiladas de BASIC), funciona con mucha lentitud. Si la velocidad no es tan decisiva (p. ej., en un programa para calcular el saldo de su cuenta bancaria), la lentitud del BASIC interpretado es soportable. Pero si la velocidad resulta esencial (como es el caso de un programa para animación en pantalla que utilice gráficos, o cuando se trata, por ejemplo, de un "reloj" empleado para medir las reacciones en un experimento de laboratorio), hay muchas probabilidades de que el BASIC interpretado resulte demasiado lento.

Si usted necesita velocidad en sus programas tiene dos caminos a seguir: programar en código de lenguaje máquina o bien en lenguaje ensamblador (véase p. 448), que es difícil y laborioso, o programar en un lenguaje compilado como el PASCAL o el FORTH. Los lenguajes compilados no son difíciles de aprender, pero el código fuente (el programa original) casi con toda seguridad contendrá errores, que el compilador descubrirá cuando intente compilar el programa. Éstos son difíciles de rectificar, en comparación con los errores del BASIC. Una vez que se ha corregido el código fuente, el programa se tendrá que volver a compilar todo de nuevo. La mayoría de los compiladores dan dos o tres "pasadas" por el código fuente, y es probable que cada una de estas sucesivas pasadas produzca mensajes de error, cada uno de los cuales, a su vez, se habrá de corregir antes de que el programa se pueda volver a compilar.

Realizar un programa correctamente compilado puede resultar un proceso que requiera más tiempo que obtener un programa en BASIC interpretado. Por añadidura, el BASIC puede "viciar" al programador novato, a quien, ahorrándole la puerta estrecha, le permitirá malas técnicas de programación que lenguajes altamente estructurados como el PASCAL rechazarían. El BASIC tolera programas muy poco cuidados, llenos de sentencias GOTO, por ejemplo, y estos malos hábitos arraigados pueden llegar a dificultar de forma notoria la transición a lenguajes más avanzados.

Y después del BASIC, ¿qué?

El BASIC es un lenguaje flexible no difícil de aprender. Posee excelentes facilidades para manipulación de variables, pero es lento y no consigue sacar el máximo partido del potencial de un ordenador personal. Por otra parte, lenguajes más modernos, como el PASCAL y el FORTH, ofrecen facilidades de programación que en BASIC son difíciles o bien imposibles.

El PASCAL también se ideó como lenguaje de enseñanza y se diseñó específicamente para favorecer el desarrollo de programas bien construidos, "estructurados". El PASCAL es un lenguaje compilado, lo que significa que los usuarios se pueden encontrar con numerosos errores captados por el compilador (después de que se ha escrito el código fuente y antes de que se pueda ejecutar el código objeto compilado), y esto puede ser muy frustrante. Los programadores poco expertos en PASCAL hallan además en los requerimientos del lenguaje, tales como la necesidad de definir todas las variables al comienzo del programa (y no sólo eso, sino también la precisión de tener que definir de qué tipo son, reales, enteras, etc.), un impedimento para la programación libre y ágil.

Por otra parte, el PASCAL exige que el programador clarifique la lógica del programa antes de escribirlo. Es probable que los programas en este lenguaje arrojen numerosos errores sintácticos en el código fuente, pero también cabe la posibilidad —que incluso puede decirse que es más probable— de que estén bien diseñados, en cuyo caso difícilmente contendrán errores fundamentales de lógica.

Últimamente el FORTH se ha convertido en una alternativa muy popular al BASIC como lenguaje de programación para micros personales. Aunque aprender FORTH no es tan difícil como aprender lenguaje ensamblador o código de lenguaje máquina, es menos "intuitivo" que el BASIC o el PASCAL. Aun así, el FORTH posee muchos méritos exclusivos que hacen de él un buen candidato a ser el segundo lenguaje aconsejable para un programador.

A pesar de que el FORTH es un lenguaje de alto nivel, funciona casi tan rápido como el código de lenguaje máquina, debido a la forma exclusiva en que trabaja. Mientras que un lenguaje como el BASIC posee un número fijo de sentencias y órdenes, los usuarios de FORTH pueden definir su propio vocabulario.

La palabra clave PRINT en BASIC significa que cualquier carácter que venga a continuación entre comillas dobles se imprimirá en la pantalla. El programador no puede alterar este hecho. En FORTH, PRINT se podría definir para que haga, por ejemplo, un listado en la pantalla, impreso en una columna vertical, de los equivalentes hexadecimales de los códigos ASCII correspondientes a los caracteres que componen una variable.

El FORTH le otorga al programador el poder de definir cualquier palabra para significar lo que desee y para producir los resultados deseados cada vez que sea empleada a partir de entonces. Es no sólo sumamente flexible en este sentido, sino que ofrece al usuario la ventaja adicional de que también produce programas que se pueden compilar en código objeto (véase p. 184), que son casi tan compactos y de rápido funcionamiento como los programas en lenguaje máquina.

Aunque existen muchos lenguajes de programación, la mayoría de los aficionados se inclinarán, después del BASIC, a escoger entre el lenguaje ensamblador, el PASCAL y el FORTH. Muy brevemente sintetizaremos a continuación las ventajas y las desventajas de cada uno de ellos:

BASIC
Fácil de aprender
Fácil de recordar
Fácil de depurar
De lenta ejecución
Ocupa mucha memoria
No favorece la programación estructurada

Lenguaje ensamblador
No tan fácil de aprender
No tan fácil de recordar
Difícil de depurar

De muy rápida ejecución
Otorga al usuario un control completo sobre el microprocesador

PASCAL
Bastante fácil de aprender
Bastante fácil de recordar
Más difícil de depurar que el BASIC
Favorece mejores técnicas de programación
De ejecución más rápida que el BASIC y más lenta que el ensamblador
Necesita compilarse, lo que lleva su tiempo; después de compilado, funciona casi tan rápidamente como el ensamblador
Otorga bastante control sobre el microprocesador, pero menos que el ensamblador; el manejo de variables no es tan sencillo como en BASIC

FORTH
No resulta muy fácil de aprender; más fácil de asimilar por los principiantes absolutos que por programadores en BASIC
Bastante fácil de recordar
Muy fácil de depurar en modalidad de intérprete
Se puede compilar; se ejecuta casi tan rápidamente como el lenguaje ensamblador
Otorga control completo sobre el microprocesador
Economiza memoria
Más fácil de aprender que el lenguaje ensamblador, aunque menos "intuitivo" que el BASIC

Complementos al BASIC

LYNX 96

```
1 REM *CREAR ARCHIVO DE DATOS*
2 DIM N$(30)
3 LET N$ = "@VACIO"
4 DIM F$(15)
5 LET F$ = "FICTICIO"
6 LET Z = 2
7 EXTBACK 1
8 EXTSTORE 1,Z,N$,N$,N$
9 EXTSTORE 1,F$,F$,F$,F$
10 INPUT "INSERTAR CINTA DE DATOS,
   PULSAR RECORD, Y DIGITAR 'Y'";A$
11 SSAVE 1, "DAT.AGCO"
12 PRINT "PARAR LA CINTA Y
   REBOBINAR"
13 END
```

Nota: Éste es el programa de inicialización para el Lynx de 96 K; carecemos de información acerca de la manipulación de archivos en cinta para los otros modelos.

Variables del programa principal
Copiar el listado para el Spectrum con estas sustituciones para las variables numéricas:

Cambiar	TAMA	por Z
	RMOD	por R
	CLAR	por D
	CURS	por C
	OPCN	por O
	INF	por i
	MED	por m
	SUP	por s

y efectuar las siguientes modificaciones, sustituciones y supresiones de líneas:

```
1100 REM S/R *CREMAT*
1110 DIM N$(30)(50)
1120 DIM M$(30)(50)
1130 DIM C$(30)(50)
1140 DIM D$(15)(50)
1150 DIM P$(15)(50)
1160 DIM T$(15)(50)
1170 DIM X$(15)(50)
1180 DIM Z$(30)
1210 LET Z = 0
1220 LET R = 0
1230 LET D = 1
1240 LET C = 0
1250 LET Z$ = "@VACIO"
1260 LET Q$ = ""
1300 RETURN
.
1400 REM S/R *LEARCH*
1405 PRINT "INSERTAR CINTA DE DATOS Y
     PULSAR PLAY"
1410 GOSUB 3100
1420 SLOAD 1, "DAT-AGCO"
1430 PRINT "PARAR LA CINTA"
1440 GOSUB 3100
1450 EXTBACK 1
1460 EXTFETCH 1,Z
1470 FOR K = 1 TO Z-1
1480 EXTFETCH 1,
     N$(K),M$(K),C$(K),D$(K),P$(K),T$(K),
     X$(K)
1490 NEXT K
1500 LET Q$ = N$(1)
1510 RETURN
.
3120 IF KEYN <> 32 THEN LET L = 0
.
3780 LET A$ = KEY$
.
3810 LET O = VAL(A$)
3820 IF (O < 1) OR (O > 9) THEN LET
     L = 0
.
4500 REM S/R *MODNOM*
4510 REM CONVERSION A MAYUSCULAS
4520 LET R$ = UPC$(N$(Z))
     (suprimir desde línea 4530 a 4590)
.
4600 LET S$ = ""
4601 LET A$ = ""
4602 LET T = LEN(R$)
4603 LET S = 0
.
4610 REM LOCALIZAR ULTIMO ESPACIO
.
4630 IF MID$(R$,L,1) = " " THEN LET
     S = L
.
4670 IF MID$(R$,L,1) > "@" THEN LET
     S$ = S$ + MID$(R$,L,1)
.
4710 IF MID$(R$,L,1) > "@" THEN LET
     A$ = A$ + MID$(R$,L,1)
```

Desde la 5410 a la 5460, las líneas se deben reducir a sentencias simples, por ejemplo:

```
5410 LET U$ = N$(L):LET
     N$(L) = N$(T):LET
     N$(T) = U$ se convierte en
5410 LET U$ = N$(L)
5411 LET N$(L) = N$(T)
5412 LET N$(T) = U$
```

y así sucesivamente, sin otros cambios.

```
.
5600 REM S/R *GRDREG*
5605 PRINT "INSERTAR CINTA DE DATOS
     Y PULSAR RECORD"
5610 GOSUB 3100
5620 EXTBACK 1
5630 EXTSTORE 1,Z
5640 FOR K = 1 TO Z-1
5650 EXTSTORE 1,
     N$(K),M$(K),C$(K),D$(K),P$(K),T$(K),
     X$(K)
5660 SSAVE 1, "DAT-AGCO"
5670 PRINT "PARAR LA CINTA"
5680 GOSUB 3100
5690 RETURN
.
5855 LET X = 0
5860 LET m = INT((i + s)/2)
5870 IF M$(m) = U$ THEN LET X = 1
```

```
5880 IF U$ > M$(m) THEN LET i = m + 1

6080 LET A$ = KEY$
6110 IF A$ = " " THEN RETURN
6120 GOSUB 6200
6130 RETURN

6730 FOR I = 1 TO 1
6735 LET I = 0
6740 LET A$ = KEY$
6750 IF (A$ = E$) OR (A$ = " ") THEN
     LET I = 1
6760 NEXT I
```

Este fragmento se debe reproducir en las líneas 6880-6910, 6990-7030, 7110-7140, 7220-7250, 7640-7670

DRAGON 32

BBC MICRO

COMMODORE 64

VIC-20

Programa de inicialización
Éste es el programa de inicialización para el Dragon 32.

```
1 REM *CREAR ARCHIVO DE DATOS*
2 LET Z = 2
3 LET N$ = "@VACIO"
4 OPEN "0", #-1, "DAT-AGCO"
5 INPUT "INSERTAR CINTA DE DATOS,
  PULSAR RECORD Y DIGITAR 'Y'";A$
6 PRINT# - 1,Z,N$,N$,N$,N$,N$,N$,N$
7 CLOSE# - 1
8 PRINT "PARAR LA CINTA Y REBOBINAR"
9 STOP
```

En el BBC Micro, sustituir las líneas 4, 6 y 7 por:

```
4 F1 = OPENOUT("DAT-AGCO")
6 PRINT #F1,Z,N$,N$,N$,N$,N$,N$,N$,
7 CLOSE #F1
```

En el Commodore 64 y en el Vic-20, reemplazar las líneas 4, 6 y 7 por:

```
4 OPEN 1,1,2,"DAT-AGCO"
6 PRINT #1,Z,N$,N$,N$,N$,N$,N$,N$,
7 CLOSE 1
```

Variables del programa principal
En el Dragon, los Commodore y el BBC Micro, copiar el listado del Spectrum (publicado completo en p. 458) con estas sustituciones para las variables numéricas.

Cambiar:		
	TAMA	por Z
	RMOD	por R
	CLAR	por D
	CURS	por C
	OPCN	por O
	INF	por IN
	MED	por MD
	SUP	por SP

y efectuar las siguientes modificaciones, sustituciones y supresiones de líneas:

```
1100 REM S/R *CREMAT*
1110 DIM N$(50)
1120 DIM M$(50)
1130 DIM C$(50)
1140 DIM D$(50)
1150 DIM P$(50)
1160 DIM T$(50)
1170 DIM X$(50)
```

Eliminar las líneas 1180-1190

```
1210 LET Z = 0
1220 LET R = 0
1230 LET D = 1
1240 LET C = 0
1250 LET Z$ = "@VACIO"
1260 LET Q$ = ""
1300 RETURN
```

Ésta es la versión de la subrutina 1400 para el Dragon 32:

```
1400 REM S/R *LEARCH*

1410 OPEN"I", #-1, "DAT-AGCO"
1420 PRINT "INSERTAR CINTA DE DATOS Y
     PULSAR PLAY"
1430 GOSUB 3100
1440 INPUT #-1,Z
1450 FOR K = 1 TO Z-1
1460 INPUT # - 1,1N$(K),M$(K),C$(K),
     C$(K),D$(K),P$(K),T$(K),X$(K)
1470 NEXT K
1480 Q$ = N$(1)
1490 CLOSE #-1
1500 PRINT "PARAR LA CINTA"
1510 GOSUB 3100
1520 RETURN
```

En el listado anterior, para el BBC Micro reemplazar la línea 1410 por:

```
1410 F1 = OPENIN("DAT-AGCO")
```

y sustituir #-1 por #F1 en las líneas 1440, 1460 y 1490

En el listado anterior, para el Commodore 64 y el Vic-20 reemplazar la línea 1410 por:

```
1410 OPEN 1,1,0,"DAT.AGCO"
```

Reemplazar #-1 por #1 en las líneas 1440 y 1460, y sustituir la 1490 por:

```
1490 CLOSE 1
```

En el BBC Micro, reemplazar INKEY$ por INKEY$(0) en todos los casos; y sustituir INPUT"...mensaje...";A$ por INPUT"...mensaje...",A$. En los Commodore, reemplazar LET A$ = INKEY$ por GET A$ en todos los casos; y sustituir IF INKEY$... por GET GT$:IF GT$...

En el BBC Micro, el Dragon y los Commodore, en la subrutina 4500 sustituir todas las referencias a R$(L) por MID$(R$,L,1). Reemplazar CODE... por ASC(...). Reemplazar PRIMER por ULTIMO en la línea 4610.
Suprimir: LET L = T del final de la línea 4630.

Ésta es la versión para el Dragon de la subrutina 5600; para las variaciones relativas al BBC y al Commodore, véanse las notas anteriores sobre el programa de inicialización.

```
5600 REM S/R *GRDREG*
5610 OPEN"0",#-1,"DAT-AGCO"
5620 PRINT "INSERTAR CINTA DE DATOS Y
     PULSAR RECORD"
5630 GOSUB 3100
5640 PRINT #-1,Z
5650 FOR K = 1 TO Z-1
5660 PRINT # - 1,Z,N$(K),M$(K),C$(K),
     D$(K),P$(K),T$(K),X$(K)
5670 NEXT K
5680 CLOSE #-1
5690 PRINT "PARAR LA CINTA"
5693 GOSUB 3100
5695 RETURN
```

En el BBC Micro, en la subrutina 6200 insertar:

```
6205 VDU 2
6275 VDU 3
```

y sustituir LPRINT por PRINT.

En los Commodore, insertar:

```
6205 OPEN 4,4:CMD 4
6275 PRINT #4:CLOSE 4
```

y sustituir LPRINT por PRINT.

En el Dragon, reemplazar LPRINT por PRINT #-2.

Cimientos sólidos

En la historia del microordenador, los avances en hardware y software van parejos, pues en ellos cuentan por igual tanto los productos como las personas

Ha habido momentos en la historia en que la magnitud del cambio tecnológico ha apabullado a la gente. Pero hasta la fecha, nada (ni siquiera la navegación aérea, desde los hermanos Wright hasta la exploración lunar) es comparable, en ritmo de progreso, a la revolución de la microelectrónica. El salto de los primitivos microprocesadores a los diseños actuales de 16 bits, de los primeros microsistemas a los actuales ordenadores de consola, se ha realizado en apenas una década. Y la velocidad de los avances es uniformemente acelerada.

Por el año 1971, varias de las nuevas firmas fabricantes de chips de California llegaron a la conclusión de que las principales funciones de un ordenador se podían alojar en una sola brizna de silicio. En aquel entonces no se intuía revolución alguna, ni se hablaba de ninguna "tecnología de la información". La idea se limitaba a producir un ordenador pequeño y barato que se pudiera utilizar para controlar las máquinas de las fábricas o los ascensores, y los primeros microprocesadores fueron muy adecuados para esas tareas.

Uno de estos fabricantes de chips, Intel, pasa por ser el inventor del primer microprocesador, denominado 4004. Los "cuatros" de la cifra aluden a su potencia: se trataba de un procesador de cuatro bits que manipulaba los datos en bloques de cuatro dígitos binarios. Estaba provisto de poca memoria, la suficiente como para albergar el programa de control de un ascensor, por ejemplo.

Hacia 1972 Intel había creado el chip 8008, un procesador de ocho bits, y los aficionados comenzaron a pensar en construirse ordenadores por sí mismos basándose en el nuevo chip. En las revistas norteamericanas para aficionados a la electrónica aparecían artículos describiendo cómo hacerlo, y aunque los ordenadores resultantes no poseían pantallas de monitor, ni teclados adecuados ni ningún otro medio auxiliar más evolucionado, aquéllas fueron las primeras máquinas personales. A partir de uno de estos proyectos para aficionados nació lo que podría considerarse el primer microordenador personal comercial que salió al mercado, el Altair-8800. No obstante, sólo se vendía como *kit* (es decir, como juego para montar).

Al año siguiente apareció el primer microprocesador "verdadero", el 8080, también producido por Intel. Operaba sobre bloques de datos de ocho bits y podía manipular hasta 64 Kbytes de memoria para programas más extensos. El reto fue recogido por las otras firmas de chips, que apretaron el paso. El chip 6800 de Motorola operaba de forma bastante similar al 8080. Las características de hardware eran parecidas, pero para hacerlo funcionar se requerían distintas instrucciones. Y aquí empezaron a suscitarse los problemas de la compatibilidad de software: los programas escritos para el 8080 no se podían ejecutar con el 6800, y viceversa.

Por la misma época, otras empresas estaban sacando al mercado procesadores similares, entre ellas National Semiconductor y Signetics and Advanced Micro Devices. Pero la impulsora principal de turno habría de ser MOS Technology, donde trabajaba uno de los protagonistas de nuestra historia, Chuck Peddle (véase p. 180). Peddle estaba en MOS Technology cuando la empresa creó un procesador muy parecido al 6800 de Motorola, denominado 6500. De hecho, era tan similar al 6800 que hubo que efectuar algunas modificaciones y al chip revisado al fin se le dio el nombre de 6502.

Los padres del invento
Aunque Chuck Peddle diseñó tanto el Commodore PET como el microprocesador 6502 en el cual se basaba, la contribución de Bill Gates, autor del BASIC Microsoft que está incorporado en la ROM del PET, fue igualmente importante

Cortesía de Commodore

Chuck Peddle

Commodore ya era una empresa muy conocida en Canadá en maquinaria para oficina y calculadoras electrónicas. Peddle se incorporó a la empresa con la idea de producir un ordenador personal completo, con pantalla, teclado, cassettes para el almacenamiento de programas y todo aquello que debía poseer un verdadero ordenador; todo, por supuesto, construido alrededor del procesador 6502. La máquina apareció en 1976 y se llamó PET 2001, un nombre amistoso (*pet* en inglés significa "animal de casa") que se escogió para que no se espantara el comprador de la calle considerándolo altamente técnico.

Pero mientras el PET entraba en el mercado, otros dos innovadores se preparaban en un garaje de California para comercializar su ordenador. Steve Wozniak (véase p. 155) siempre había deseado tener un ordenador, y al hacerse socio del Homebrew Computer Club comprobó que lo podía hacer. Diseñó un ordenador en un único tablero de

Tony Sleep

Bill Gates

circuitos y, con su amigo Steve Jobs, empezó a fabricarlo y venderlo. A su tablero lo llamaron el Apple I. Alojado en una caja con teclado, el ordenador finalmente se transformaría en el clamoroso Apple II. Esta máquina salió justo después del PET de Peddle e incubó toda una industria casera de fabricación de software y hardware.

La Tandy Corporation de Fort Worth (Texas) tenía sus propias ideas para el mercado del pequeño ordenador. La corporación fabricaba, y continúa haciéndolo, una amplia gama de artículos electrónicos, como equipos de alta fidelidad, sintetizadores y radios, vendiéndolos a través de su cadena de tiendas. El ordenador personal representaba una ampliación natural de su línea comercial, y a través de las tiendas Radio Shack ya contaba con una red de distribución para todo Estados Unidos. El resultado fue el TRS-80 Model 1, otro gran éxito en el mercado norteamericano. TRS corresponde a las siglas de Tandy Radio Shack, y el 80 alude al microprocesador utilizado, el Zilog Z80. Zilog era otra nueva firma de chips que había producido un procesador similar al Intel 8080 pero con sustanciales mejoras.

Al contar el TRS-80 Model 1 con un microprocesador Z80, y el Apple II y el Commodore PET con sendos 6502, los ordenadores personales comenzaron a diversificarse en el hardware. Pero junto con esta primera elección trascendental del tipo de consumidor, llegaron los problemas asociados con incompatibilidad de máquinas y software no estandarizado. La clase de microprocesador utilizado en las primeras máquinas es significativa, porque el chip determina la elección del software que terceras partes comercializan en el mercado. Mientras se desarrollaba el hardware, se estaban estableciendo, asimismo, los estándares relativos al software.

El alma de la empresa
Steve Wozniak diseñó y construyó, en el garaje de su casa, el primer Apple I (un TCI sin carcasa). Cuando se modificó el diseño y se lo colocó en una caja, lo que dio lugar a la creación del Apple II, su amigo Steve Jobs hizo del Apple el éxito comercial que es actualmente

Steve Jobs

Cortesía de Apple

En 1972, un joven llamado Gary Kildall era asesor de Intel. Su firma, Microprocessor Application Associates, estaba trabajando en un lenguaje para ordenador que los ingenieros de Intel pudieran utilizar para escribir software destinado a los nuevos chips de microprocesador fabricados por Intel. Kildall pensó que era posible conectar un microprocesador con memoria a una unidad de disco flexible de 8 pulgadas y a un teletipo, con el fin de proporcionarle a cada ingeniero un ordenador propio. Pero Intel prefirió continuar con su práctica de que todos sus ingenieros compartieran una máquina de unidad principal.

Kildall y su amigo John Torode, en otro garaje californiano, montaron ellos solos un sistema. Torode construyó el hardware para hacer que el disco flexible funcionara con el procesador, y Kildall escribió el software en virtud del cual el procesador podía manejar el disco. Al programa se lo llamó CP/M (*Control Program/Microcomputers*), nombre derivado del trabajo de Kildall con el lenguaje de programación de Intel, al que se le había dado la denominación de PL/M (*Programming Language/Microcomputers*).

El primer sistema operativo en disco para micros lo asumieron rápidamente los fabricantes de hardware que deseaban dotar a sus máquinas de unidades de disco. También el software influyó en su diseño: el CP/M sólo operaba en el 8080 y en los procesadores más rápidos 8085 de Intel, así como en el similar Z80 de Zilog. El Z80 se convirtió en el chip estándar para cualquier máquina CP/M, y la compatibilidad con el CP/M era la obsesión de las firmas de software.

Cortesía de Apple

Steve Wozniak

Gary Kildall
Los sistemas operativos más recientes los desarrollan grandes equipos de programadores, pero el CP/M lo escribió íntegramente Gary Kildall. Incluso algunas de las últimas versiones reflejaban el hecho de que lo compuso para un hardware muy imperfecto

Cortesía de Osborne

Adam Osborne
Algunas personas dicen de él que es un "cazador furtivo convertido en guardabosques"; Adam Osborne fue durante muchos años un periodista especializado en artículos sobre microordenadores, antes de que creara su propia empresa y produjera el primer ordenador portátil del mundo

Aparte de los sistemas operativos, los ordenadores personales necesitaban un lenguaje de programación en el cual la gente pudiera escribir sus programas. El BASIC, desarrollado en el Dartmouth College (Estados Unidos) como un lenguaje fácil de aprender, era la alternativa obvia.

Bill Gates, graduado en Seattle, ideó un intérprete de BASIC para micros, un programa de traduc-

ción que cabía en un chip de memoria limitada y que se podía incorporar a una máquina personal. Microsoft, la empresa de Gates, se convirtió en la productora estándar de lenguajes, así como Digital Research se erigió en la productora estándar de sistemas operativos, lo que reportó una fortuna tanto a una como a otra.

A estos progresos siguieron rápidos avances en hardware y en software de aplicaciones. Dan Bricklin y Bob Frankston produjeron el primer programa de hoja electrónica para micros, el VisiCalc, en su empresa Software Arts. Distribuido por Personal Software para el Apple II, éste se convirtió en el paquete de aplicaciones más vendido de todos los tiempos, y para denotar con mayor claridad su relación, Personal Software cambió su nombre por VisiCorp. MicroPro, de Seymour Rubinstein, produjo el WordStar, que se convirtió en el mayor éxito de ventas del mercado de tratamiento de textos CP/M.

El hardware en el que se ejecutaban estos paquetes se abarató y se hizo más potente. Adam Osborne, que se inició como escritor técnico, periodista y editor de software, después de trasladarse de Gran Bretaña a Estados Unidos, lanzó un celebrado ordenador de gestión empresarial que en su precio, ya de por sí competitivo, incluía una gran cantidad de software caro. Y está, por supuesto, sir Clive Sinclair, que estableció nuevos niveles de precio con el ZX80, ZX81, y ZX Spectrum, poniendo los ordenadores personales al alcance de millones de usuarios noveles.

Sir Clive Sinclair
Siguiendo a productos suyos tan innovadores como equipos de alta fidelidad, calculadoras, radios en miniatura, minitelevisores y relojes digitales, el éxito sin precedentes de sus microordenadores (ZX80, ZX81 y el Spectrum) le valió el título de *sir* en 1983

Cortesía de Sinclair Research

Cortesía de Sinclair Research

En los dos últimos años, el estándar del microordenador lo ha establecido la IBM con el IBM PC. Lanzada al mercado en 1982, la popularidad de esta máquina aumenta día a día. Virtualmente todas las casas de software y fabricantes de periféricos de hardware en la actualidad están produciendo material para el PC, lo cual, a su vez, es un factor de peso para que cada vez más personas opten por esta máquina.

El IBM PC ha reunido a varios de los pioneros de los primeros tiempos de la industria del micro. El microprocesador proviene de Intel, que fue la iniciadora de esa tecnología; el sistema operativo es de la Microsoft de Bill Gates, diversificadora de lenguajes, y de la Digital Research de Gary Kildall, y dos de los primeros paquetes de software con que contó la máquina fueron el VisiCalc y el WordStar.

Steve Wozniak y Steve Jobs dirigen todavía la Apple, que en líneas generales es la competidora directa de la IBM, y ha apostado todas las expectativas de su empresa en favor de la revolucionaria tecnología del Lisa (véase p. 261) y el Macintosh (una versión reducida del Lisa a un precio algo más asequible). Por su parte, Chuck Peddle fundó su propia compañía, la Sirius, y se apropió de un gran sector del mercado británico antes de que la poderosa IBM se implantara en el Reino Unido; desde entonces, la empresa fundada por este pionero ha atravesado dificultades financieras.

Pero seguramente Peddle saldrá adelante. La corta historia del negocio del micro demuestra que los iniciadores son también los supervivientes, incluso aunque las multinacionales intenten ganar la partida.

Cortesía de Hewlett Packard

Comienzos menudos
Sorprendentemente, la tecnología de los microordenadores se desarrolló más a partir de las sofisticadas calculadoras programables (como esta Hewlett-Packard HP65) que de la anterior generación de miniordenadores

Herman Hauser

Judy Goldhill

Fecunda colaboración
Aunque menos innovadora en cuanto a precio que Sinclair, la contribución de Chris Curry y Herman Hauser (como diseñadores y directores de los ordenadores Acorn) ha sido igualmente valiosa. Tanto el Acorn Atom como el BBC Microcomputer y el Electron son hitos por derecho propio

Judy Goldhill

Chris Curry

Ian McKinnell

Una de las grandes
La aceptación por parte de la IBM de la viabilidad del microordenador no tuvo lugar hasta 1982, pero aun así tuvo el efecto previsible. En la actualidad, casi todos los nuevos microordenadores de gestión alardean de ser compatibles con el IBM PC para capitalizar la inmensa base de software existente