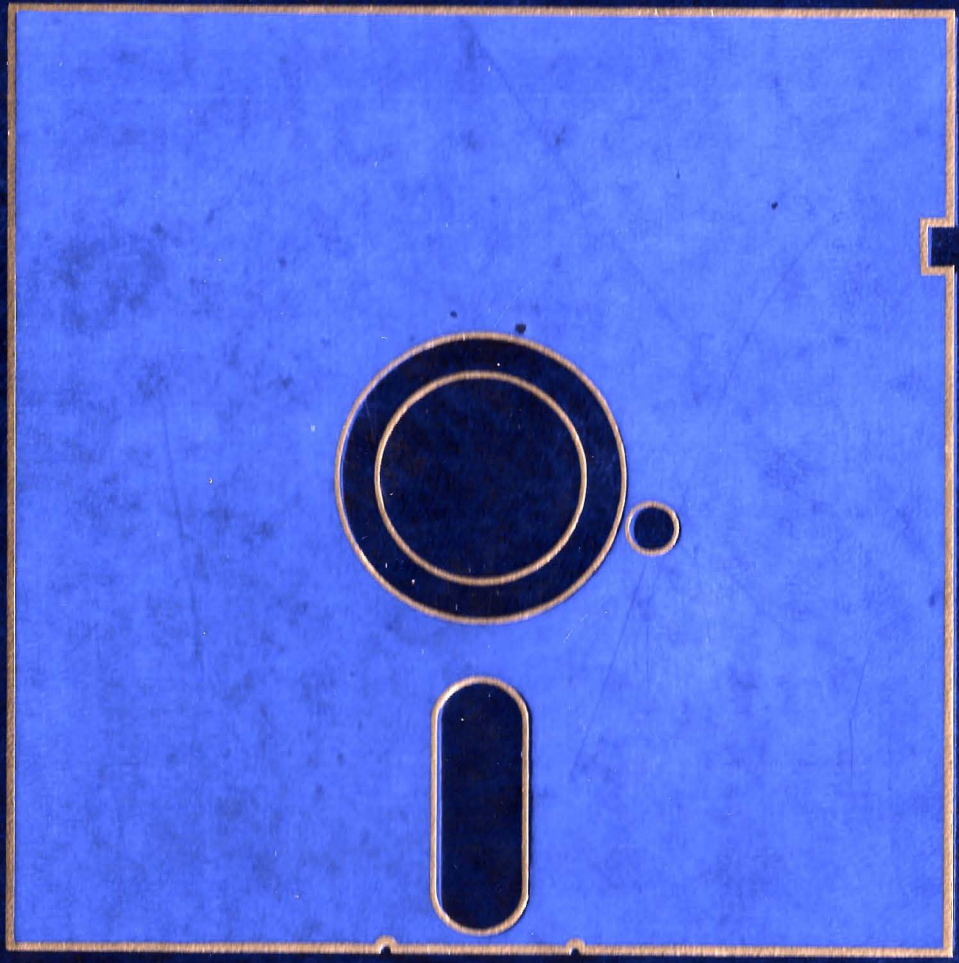
SOFTWARE



SOFT
WARE

SOFT WARE

Nueva Lente/Ingelek

una publicación

EDICIONES NUEVA LENTE, S. A.,
y EDICIONES INGELEK, S. A.

Director-Editor:
Por NUEVA LENTE, S. A.:
MIGUEL J. GOÑI
Por INGELEK, S. A.:
ANTONIO M. FERRER
Director de producción:
RICARDO ESPAÑOL
Jefe de producción:
SANTOS ROBLES

Director de la obra:
FRANCISCO LARA (PL/3)
Colaboradores:
MANUEL MUÑOZ
DAVID SANTAOLALLA
SANTIAGO RUIZ
LUIS COCA
MIGUEL ANGEL VILA
MIGUEL ANGEL SANCHEZ
VICENTE ROBLES

Diseño e ilustración:
JOSE OCHOA (PL/3)
Maquetación:
JUAN JOSE DIAZ SANCHEZ
Fotografía:
(Equipo Gálata)
EDUARDO AGUDELO y
ALBINO LOPEZ
Redacción:
PL/3 calc

Dirección, Redacción y
Administración:
Benito Castro, 12. 28028 Madrid
Tels. 245 45 98 y 246 73 67

ISBN, tomo primero: 84-7534-103-9
ISBN, tomo segundo: 84-7534-104-7
ISBN, tomo tercero: 84-7534-105-5
ISBN, tomo cuarto: 84-7534-106-3
ISBN de la obra: 84-7534-101-2

Fotomecánica:
OCHOA, S. A.
Ricardo Ortiz, 74. 28017 Madrid
Impresión:
GRAFICAS REUNIDAS, S. A.
Avda. de Aragón, 56. 28027 Madrid
Depósito legal: M. 41.955-1984
PRINTED IN SPAIN

El ordenador útil

Programando aplicaciones

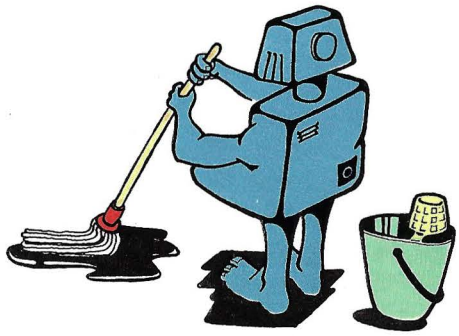
Tras algunos capítulos dedicados a la explicación de nuevos conceptos, volvemos en éste a ocuparnos de la práctica del BASIC. En anteriores capítulos prácticos se presentaron ejemplos en los que se hacía uso de los comandos BASIC para fines fundamentalmente lúdicos. En éste utilizaremos la potencia del BASIC para resolver algunos problemas cotidianos.

Un ordenador es una máquina de calcular. Sus posibilidades, sin embargo, superan a las de cualquier calculadora. Con un ordenador se pueden realizar tareas mucho más complejas y con mayor facilidad. Y es para eso, precisamente, para lo que está pensado el ordenador.

A lo largo del presente capítulo se construirá una base de datos. Esta no será ni mucho menos tan potente como una base de datos comercial, sin embargo la podremos diseñar a nuestra medida y, desde luego, resultará bastante más económica.

UNA BASE DE DATOS

En principio, una base de datos consiste en un conjunto organizado y más o menos voluminoso de datos. Estos datos mantienen ciertas relaciones entre sí. Un ejemplo de base de datos lo constituye el propio listín telefónico. En él coexisten dos tipos de datos que se relacionan mutuamente: el nombre del abonado y el número de su teléfono. En general, una base de datos puede disponer de más tipos de datos, denominados normalmente "campos". Cada grupo de datos diferentes recibe el nombre de "ficha" o "registro". De esta forma, un listín telefónico puede

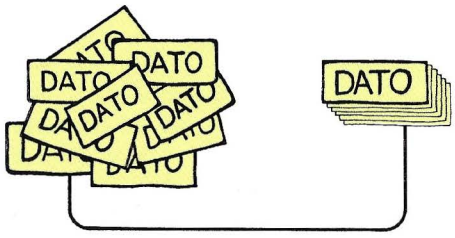


El ordenador útil ha de colaborar con el usuario en sus labores cotidianas.

constar, por ejemplo, de 100 registros de dos campos.

El método más adecuado para guardar esta información pasa por el uso de un "array" o matriz. Se puede definir un array de dos dimensiones, 100×2, asegurándose así el espacio suficiente para los datos. Al tratarse de datos numéricos y alfanuméricos no se puede emplear una matriz numérica. Lo más conveniente, pues, es crear la matriz como un array de caracteres, transformando los datos numéricos en alfanuméricos (función STR$). Si más adelante fuera necesario realizar cálculos con los números almacenados, éstos serían reconvertidos previamente con la función VAL.

En una base de datos son necesarias al menos tres operaciones: creación, actualización y visualización de los datos. La primera es obvia, sin crearla no existiría tal base de datos. Una vez creada será necesario actualizarla de tiempo en tiempo; esto es, modificar datos o introducir otros nuevos. Y todo ello no sirve de nada si no es posible consultar esos datos, visualizándolos en pantalla o volcándolos en la impresora. Acometeremos pues estos tres procesos viendo las distintas posibilidades que ofrece el BASIC para su programación.



El manejo y organización de los datos es una de las tareas en las que el ordenador se manifiesta como un consumado especialista.

LA CREACION

A la hora de crear la base de datos es cuando se ha de dimensionar la matriz correspondiente. En este punto los ordenadores difieren unos de otros, dependiendo de la forma en la que tratan los datos de tipo cadena de caracteres. Los ordenadores Sinclair y Atari almacenan las cadenas en forma de arrays, conteniendo cada elemento un solo carácter. De esta forma, si A$ contiene la cadena "JAUME" el valor de A$(3) será "U".

Esto no ocurre en los restantes equipos. Lo normal es almacenar una cadena entera como elemento del array. Si en el BASIC de Sinclair y Atari la orden DIM A$(5) reserva espacio para cinco caracteres, en los demás permitirá almacenar cinco cadenas de caracteres de longitud ilimitada.

Atendiendo a estas diferencias, el dimensionamiento de la matriz puede presentar dos aspectos. Para el caso de una matriz de 100×2, en la que cada cadena ha de tener un máximo de 20 caracteres, sus formulaciones serían las siguientes. En los equipos Atari y Sinclair:

```
10 DIM A$(100,2,20)
```

Mientras que en los restantes dialectos BASIC:

```
10 DIM A$(100,2)
```

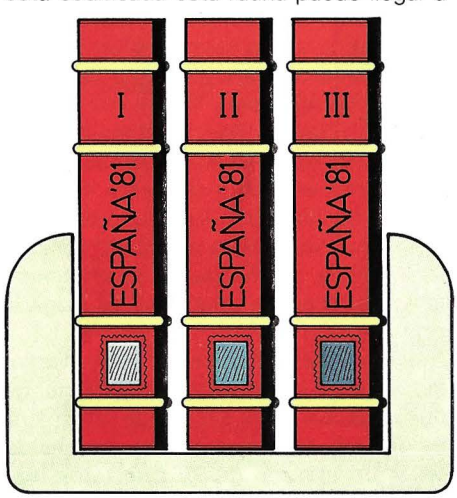
A lo largo del presente capítulo nos acogeremos a la segunda formulación, más genérica. No obstante, si su ordenador pertenece a los nombrados, no se olvide de realizar las modificaciones indicadas.
Como tamaño general de la base de datos elegiremos 100 registros de 10 campos cada uno. Con ese presuesto, la línea de dimensionado se reduce a:

```
10 DIM A$(100,10)
```

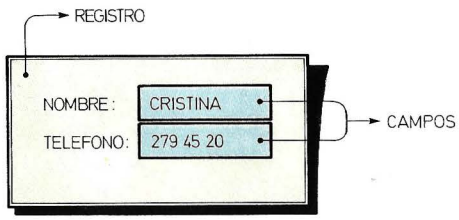
En ella se indica que el primer subíndice corresponde a los registros y el segundo a los campos. Una vez dimensionada la matriz, hay que proceder a "rellenarla" con datos. Estos se han de introducir por medio del teclado, de ahí que sea conveniente utilizar el comando INPUT. Para recorrer los campos de todos los registros se hace uso de un doble bucle.

```
1000 FOR REG=1 TO 100
1010 FOR CAM=1 TO 10
1020 INPUT B$
1030 LET A$(REG,CAM)=B$
1040 NEXT CAM
1050 NEXT REG
■
```

Este irá pidiendo los datos uno a uno hasta completar toda la tabla. Tal y como está codificada esta rutina puede llegar a



Nuestra base de datos servirá para guardar información de forma organizada y accesible.



En las bases de datos, éstos suelen agruparse en registros; a su vez, cada registro acoge a los datos guardándolos en los respectivos campos.

aburrir el ver 100×10=1000 veces el signo "?" mientras introducimos los datos. Lo más adecuado será visualizar un indicador del elemento que se está introduciendo. Así, al mismo tiempo tendremos constancia directa del dato que introducimos y dónde lo hacemos. Para lograrlo basta con introducir una línea intermedia en el programa:

```
1015 PRINT "REGISTRO: ";REG;
     "CAMPO: ";CAM
```

De esta forma, la ejecución de esta primera rutina daría el siguiente resultado en pantalla:

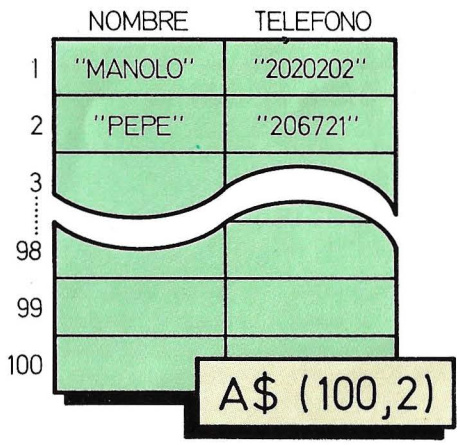
```
RUN<CR>
REGISTRO: 1 CAMPO: 1
?MANUEL
REGISTRO: 1 CAMPO: 2
? 2340700
.....
REGISTRO: 100 CAMPO: 10
? FIN
■
```

En la mayoría de los casos no se hará uso de todos los registros y campos. Por lo tanto, no será imprescindible rellenarlos todos. Para evitar la molestia de pasar por todos los campos de cada registro es preciso introducir nuevas líneas. En ellas se ha de inspeccionar el dato introducido y, si se trata de una orden de salto de registro o de fin de tarea, realizar dichas acciones.
Para indicar el final de la entrada de datos hemos de teclear la orden al efecto. Esta consistirá en un conjunto de caracteres plenamente diferenciados de un dato ordinario. Por ejemplo, puede emplearse un carácter especial, poco usual, que sea identificado como indicativo de un comando. Cualquier carácter especial (#, $, %, *, £, @, etc.) puede servir al efecto. En nuestro caso, utilizaremos el carácter "@"; si usted no dispone de este carácter o no resulta de su agrado, puede utilizar otro, siempre y cuando lo tenga en cuenta en las siguientes líneas de programa.
En definitiva, podemos definir el comando de "fin de entrada de datos" como "@F" (F de fin), y el de salto de registro como "@R" (R de registro).
Ahora hay que lograr que el programa contemple estas posibilidades. El salto de registro implica pasar al primer campo del siguiente registro, lo que significa hacer CAM=1 y REG=REG+1. Sin embargo, como existe una línea NEXT CAM y otra NEXT REG, habrá que poner los valores anteriores (los comandos NEXT incrementan la variable a la que hacen referencia). Por consiguiente, los valores adecuados son CAM=10 y REG tal como estaba. Esto habrá que realizarlo cuando se haya tecleado el comando apropiado (@R). La línea en cuestión adoptará el aspecto siguiente:

```
IF B$="@R" THEN LET CAM=10
```

Esta ha de colocarse después de la lectura de B$ y antes de trasladar dicho valor al array (línea 1030).
No interesa que el "comando" se vea reflejado en la base de datos como un dato más, por ello, en tal caso, se ha de saltar la línea 1030. Así pues la correcta programación del citado "comando" coincidirá con:

```
1025 IF B$="@R" THEN LET
     CAM=10:GOTO 1040
```



Nuestra base de datos confeccionada a título de ejemplo, emplea un "array" de cadenas de caracteres para almacenar organizadamente los distintos datos.

Por lo que respecta al final de la entrada de datos, el método varía muy poco. La línea correspondiente difiere tan sólo en los nuevos valores de CAM y REG. Para

que el bucle se dé por terminado, estos valores han de coincidir con los valores límite de cada bucle FOR: 10 y 100, respectivamente:

```
1027 IF B$="@F" THEN LET CAM=10:LET
     REG=100:GOTO 1040
```

Con esta línea queda "casi" completa la rutina de creación de la base de datos. Esta será utilizada a modo de subrutina, por lo que será necesario completarla con una línea en la que figure un comando RETURN (retorno al punto de origen del salto a subrutina). El listado completo es el que aparece a continuación:

```
10 DIM A$(100,10)
1000 FOR REG=1 TO 100
1010 FOR CAM=1 TO 10
1015 PRINT "REGISTRO: ";REG;"CAMPO:
     ";CAM
1020 INPUT B$
1025 IF B$="@R" THEN LET CAM=10:
     GOTO 1040
1027 IF B$="@F" THEN LET CAM=10:LET
     REG=100:GOTO 1040
1030 LET A$(REG,CAM)=B$
1040 NEXT CAM
1050 NEXT REG
1100 RETURN
```

EDICION

Después de crear la base de datos será necesario, con cierta frecuencia, alterar alguno de los datos. Es posible que se hayan cometido errores al introducir los datos por vez primera, o simplemente que los datos deban ser actualizados.
Para cambiar el contenido de un campo específico hay que localizar el registro al que pertenece y, por supuesto, el campo implicado dentro de ese registro. El método más directo para cambiar el contenido será asignar el nuevo valor, recogido en un INPUT, al elemento correspondiente. Ello se consigue por medio de la siguiente rutina:

```
2100 INPUT B$
2110 LET A$(REG,CAM)=B$
```

En todo caso, es obvio que antes de ello habrá que depositar los valores adecuados en REG y CAM. Estos dos datos pueden también captarse a través de sendas instrucciones INPUT. Por lo demás, también es conveniente mostrar el contenido original del campo y su identificación. Todo ello queda contemplado en las siguientes líneas de programa:

```
2010 INPUT "REGISTRO";REG
2020 INPUT "CAMPO";CAM
2030 PRINT "REGISTRO ";REG;"
     CAMPO ";CAM
2040 PRINT "CONTENIDO: ";A$(REG,CAM)
```

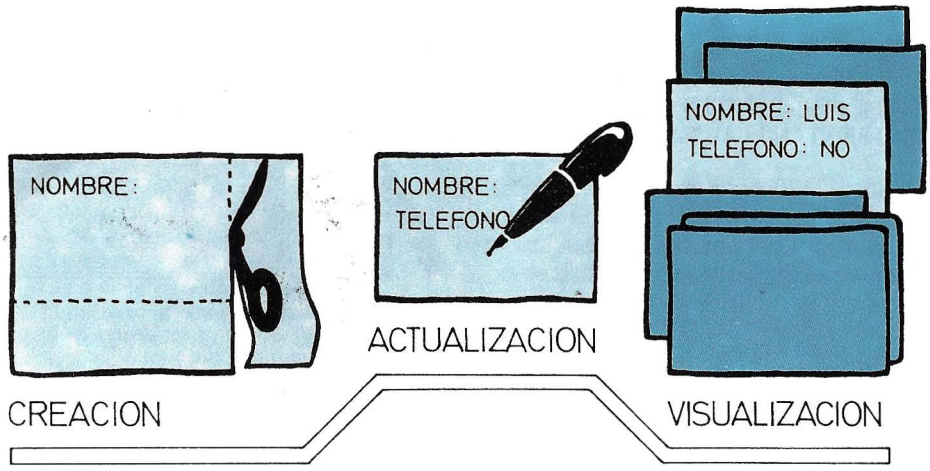
Ahora resultará fácil la edición de los datos contenidos en la base. Se elige un registro y un campo, se ve su contenido y se introduce el nuevo valor.
La ejecución de la rutina que nos ocupa producirá un resultado similar al que sigue:

```
RUN<CR>
REGISTRO? 12<CR>
CAMPO? 3 <CR>
REGISTRO 12 CAMPO 3
CONTENIDO: RUIIZ
? RUIZ
■
```

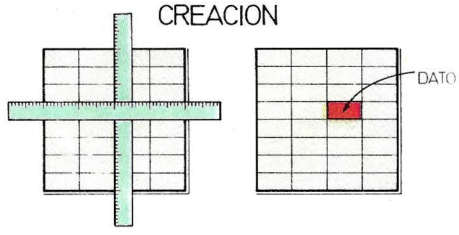
Desde luego, aún cabe realizar algunas mejoras en la rutina de edición. Es conveniente contar con la posibilidad de dejar inalterado el contenido de un campo, en el caso de que sea el correcto. También resultaría interesante la posibilidad de editar más de un campo de un tirón.
Para solventar el primer problema es suficiente con introducir una condición; por ejemplo: si no se introduce ningún valor, el contenido del campo en cuestión no debe ser alterado. Dicho efecto se consigue con la línea siguiente:

```
2104 IF B$="" THEN GOTO 2120
```

Esta nueva instrucción hace que cuando no se introduzca un dato se salte a la línea 2110, en la que se asigna el nuevo valor. El salto incondicional (GOTO) se efectúa a una línea que todavía no ha sido escrita; ésta debe ser la que permita editar varios campos en una misma operación.
Una vez introducidos los datos de registro y campo que determinan el dato a editar (líneas 2010 y 2020), se pueden seguir editando los campos que aparezcan a continuación. Para ello basta con modificar el valor de REG y CAM. Esta acción debe realizarse inmediatamente después de la edición del dato correspondiente.



Las tres acciones fundamentales a realizar en una base de datos se concretan en su creación, actualización del contenido y visualización del mismo.



En la etapa de creación de la base de datos, dimensionaremos la matriz de almacenamiento y procederemos a la introducción de los primeros datos.

```
2120 LET CAM=CAM+1
2130 IF CAM=11 THEN LET CAM=1:LET
     REG=REG+1
2140 IF REG <101 THEN GOTO 2030
2150 PRINT "NO HAY MAS DATOS"
```

La línea 2120 es la encargada de incrementar el contador de campos. En el caso de agotar los campos de un registro, se ha de pasar al siguiente registro. Ello se ordena en la línea 2130. Mientras no se acaben los registros (mientras REG sea menor que 101) se regresará a la zona de edición, mostrando el campo y pidiendo un nuevo dato (GOTO 2030). En el caso de llegar al final de la base de datos aparecerá en la pantalla el mensaje "NO HAY MAS DATOS".

Al igual que en la etapa de creación de la base, en la edición puede no ser necesario recorrer todos los datos. En este punto, introduciremos dos "comandos" que harán más cómoda esta zona del programa. El primero de ellos servirá para abandonar la edición, el otro permitirá saltar al siguiente registro.

Para mantener un paralelismo con los

Dentro de nuestro ejemplo se procede al diseño de algunos pseudo-comandos destinados a facilitar el manejo de la base de datos.

"comandos" empleados en la zona de creación, utilizaremos los mismos caracteres. Así pues, para dar por terminada la edición se tecleará "@F". La línea detectora de este comando se formula como sigue:

```
2106 IF B$="@F" THEN RETURN
```

El motivo de incluir el RETURN en el argumento de THEN obedece a que, como en el caso de la creación, se trata de una subrutina.

Al tropezar con este comando, el programa volverá a la zona principal de cuyo estudio nos ocuparemos más adelante. El salto al registro siguiente se obtendrá con los caracteres "@R". En este caso, la línea correspondiente es idéntica a la utilizada en la rutina de creación.

La segunda etapa se concreta en la modificación de los datos residentes en la base, si ello fuera preciso, y en la introducción de nuevos datos.

```
2108 IF B$="@R" THEN LET CAM=1:LET
     REG=REG+1:GOTO 2030
```

Tal y como se observa, el salto (GOTO) se realiza al punto en el que se permite la edición del dato elegido.

La rutina al completo adopta el siguiente aspecto:

```
2010 INPUT "REGISTRO";REG
2020 INPUT "CAMPO";CAM
2030 PRINT "REGISTRO ";REG;"
     CAMPO ";CAM
2040 PRINT "CONTENIDO: ";A$(REG,CAM)
2100 INPUT B$
2104 IF B$="" THEN GOTO 2120
2106 IF B$="@F" THEN RETURN
2108 IF B$="@R" THEN LET CAM=1:LET
     REG=REG+1:GOTO 2030
2110 LET A$(REG,CAM)=B$
2120 LET CAM=CAM+1
2130 IF CAM=11 THEN LET CAM=1:LET
     REG=REG+1
2140 IF REG <101 THEN GOTO 2030
2150 PRINT "NO HAY MAS DATOS"
2160 FOR I=1 TO 1000:NEXT I
2170 RETURN
```

En ella aparecen dos nuevas líneas, que introducen un retardo para que sea posible leer con comodidad el mensaje antes de regresar al programa principal.

VISUALIZACION

A estas alturas se han confeccionado ya las rutinas capaces de crear y editar una base de datos; queda por crear una tercera zona que permita visualizar los datos. Este es, precisamente, el cometido del apartado en el que nos encontramos: estudiar la rutina de visualización.

En principio, esta rutina no tiene mayor complicación que la de dar un aspecto agradable a la presentación de los datos que, por supuesto, deben ser presentados agrupadamente, por registros. Una vez elegido el registro, éste se mostrará al completo.

Para que la presentación sea satisfactoria, han de visualizarse, junto con el contenido de los campos, el número de registro y campo de cada dato. Los distintos campos se recorren por medio de un bucle de tipo FOR/NEXT. Veamos una primera aproximación a las instrucciones de la nueva rutina.

```
3100 CLS
3110 PRINT "REGISTRO NUMERO ";REG
3120 PRINT
3130 FOR CAM=1 TO 10
3140 PRINT "CAMPO ";CAM;"=",
     A$(REG,CAM)
3150 NEXT CAM
```

Con esta rutina, los datos serán visualizados después de borrar la pantalla. Cada registro que se visualice ocupará toda la pantalla. Primero, como ya se ha indicado, se borra la pantalla, y tras ello se procede a mostrar el número correspondiente al registro, en la línea 3110. La siguiente línea sirve para dejar un espacio entre esa cabecera y el grupo de datos. El contenido de cada campo se visualizará en una línea de pantalla, precedido por el indicativo del campo en cuestión.

El número del registro a visualizar debe encontrarse almacenado en la variable REG. Sin embargo, esta variable no ha

De nada sirve una base de datos si ésta no permite una adecuada presentación del contenido de los registros.

Aritmética binaria

Sin lugar a dudas, el sistema de numeración más profusamente utilizado es el decimal. Este es el habitual para el ser humano. Sin embargo, sus características lo hacen muy poco idóneo para el uso interno del ordenador. Los ordenadores utilizan el sistema de numeración binario, basado en los dígitos 1 y 0.

Para acercarse a las interioridades del ordenador resulta imprescindible conocer el sistema binario. El primer paso debe ser, forzosamente, aprender a contar en binario.

Las diferencias respecto a la representación decimal quedan patentes en la tabla adjunta. De los diez primeros números (las cifras decimales del 0 al 9), tan sólo coinciden las dos primeras. El número decimal 2 se convierte en 10 en binario, el 3 en 11, el 4 en 100, y así hasta el 9, cuya formulación binaria es 1001.

El siguiente número, esta vez ya con dos cifras, se genera sumando una unidad al anterior. Así pues, para poder seguir la cuenta es preciso conocer los fundamentos de la suma binaria.

Al igual que la suma decimal, la binaria también se realiza cifra a cifra, respetando sus posiciones respectivas; o lo que es lo mismo, operando entre sí los dígitos del mismo peso.

Por ejemplo, para sumar en decimal 17 y 35 se agrupan las unidades (7 y 5) y se suman dichos dígitos. Del resultado de esa primera operación (7+5=12) se toma el dígito de la derecha (2). El dígito sobrante (denominado de acarreo) se suma con los de las decenas (1+1+3=5). En definitiva, el resultado final es 52: cinco decenas y dos unidades.

El mismo método es el que se aplica en la suma binaria. Como en binario sólo existen dos dígitos, resulta muy fácil aprender la "tabla de sumar". Dicha tabla es la que se relaciona a continuación:

0+0=0
0+1=1
1+0=1
1+1=10

Las tres primeras expresiones coinciden con sus equivalentes en el sistema decimal. Cosa que no ocurre con la cuarta: al sumar 1 y 1 se obtiene un resultado 0 con acarreo de 1 (1+1=0 ... y me llevo una).

Con estos ligeros conocimientos no debe resultar difícil sumar números de cualquier cantidad de dígitos. Por ejemplo, los números binarios 10 y 11: primero los dígitos de menor peso 0+1=1, luego los siguientes 1+1=10. Ordenando el resultado queda 101. Un valor totalmente lógico, ya que 10 es dos y 11 es tres, su suma ha de ser 101 (el cinco decimal).

La multiplicación binaria tampoco encierra misterio alguno. El método vuelve a ser semejante al utilizado en el sistema decimal. La tabla de multiplicar binaria es la siguiente:

0×0=0
0×1=0
1×0=0
1×1=1

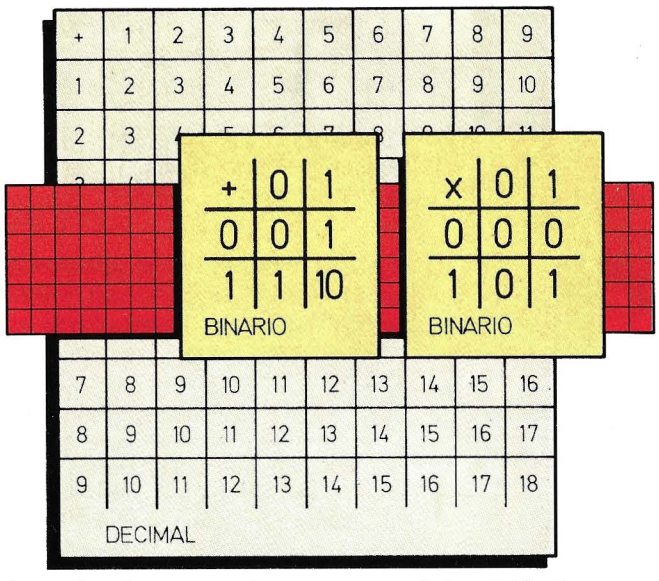
A partir de esta tabla se puede ya realizar cualquier multiplicación binaria. Como ejemplo se muestra una sencilla multiplicación de números de dos cifras

```
  10
×11
  10
10-
 110
```

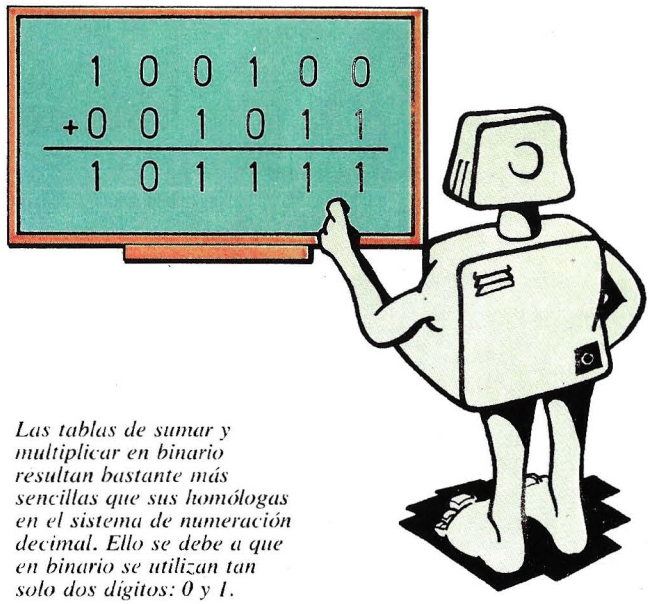
Al igual que en el caso de la suma, tampoco ahora fallan las matemáticas: tres por dos es igual a seis en cualquier sistema de numeración.

Conociendo la suma y la multiplicación binarias, la división y la resta son fácilmente deducibles. A partir de ahí trabajar en binario es tan sencillo como hacerlo en decimal.

DECIMAL	BINARIO
0	0
1	1
2	10
3	11
4	100
5	101
6	110
7	111
8	1000
9	1001



Los ordenadores operan internamente en el sistema binario.



Las tablas de sumar y multiplicar en binario resultan bastante más sencillas que sus homólogas en el sistema de numeración decimal. Ello se debe a que en binario se utilizan tan solo dos dígitos: 0 y 1.

sido inicializada en el cuerpo de la rutina que nos ocupa. Para determinar dicho valor, se ejecutará un bucle de búsqueda cuya filosofía es la siguiente: se proporciona un dato y el ordenador busca el primer registro que contenga ese dato. Una vez localizado el número del registro afectado, se pasa a la visualización del mismo. Hay que tener en cuenta que junto con el dato a buscar debe indicarse el campo en el que se ha de realizar la búsqueda.
Veamos cómo es posible programar tal proceso:

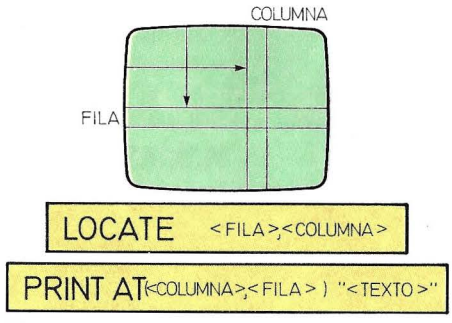
```
3010 INPUT "BUSCO EN EL CAMPO
     NUM.";CAM
3020 INPUT "QUE DATO";D$
3030 FOR REG=1 TO 100
3040 IF $$(REG,CAM)=D$ THEN GOTO 3100
3050 NEXT REG
3060 PRINT "NO ENCUENTRO ESE DATO"
3070 INPUT "BUSCO OTRO DATO (S/N)";B$
3080 IF B$="S" THEN GOTO 3010
3090 RETURN
```

Las referidas líneas de programa tienen encomendada la búsqueda del dato elegido. Ello se realiza dentro del bucle FOR/NEXT delimitado por las líneas 3030 a 3050. Dicho bucle recorre todos los registros, comparando el campo mencionado con el dato propuesto. Cuando se halla un campo cuyo contenido coincide con el elegido, se procede a visualizar el registro implicado (GOTO 3100). En el caso de no encontrar el dato propuesto, concluirá el bucle mostrando el correspondiente mensaje (línea 3060). De ocurrir tal circunstancia, el programa preguntará al usuario si desea realizar otra búsqueda, para regresar, en consecuencia, a la línea 3010 ó al programa principal.
La ejecución de esta zona de programa es la que aparece a continuación, dando por supuesto que han sido introducidos los correspondientes valores.

```
BUSCO EN EL CAMPO NUM.? 2 <CR>
QUE DATO? LUIS <CR>
NO ENCUENTRO ESE DATO
BUSCO OTRO DATO (S/N)? S <CR>
BUSCO EN EL CAMPO NUM.? 2 <CR>
QUE DATO? JUAN <CR>
■
```

La zona de visualización de la base de datos confeccionada incluye una rutina para la búsqueda de registros.

Para seleccionar la operación a realizar en la base de datos, se hace uso de un menú de opciones ejecutables.



El comando LOCATE constituye una alternativa para aquellos ordenadores que no disponen del comando PRINT AT.

Tras la introducción del dato correcto, se borra la pantalla mostrando, de inmediato, el contenido del registro apropiado.

```
REGISTRO NUMERO 12

CAMPO 1=      MADRID
CAMPO 2=      JUAN
CAMPO 3=      PEREZ
CAMPO 4=      RUIZ
CAMPO 5=      2340022
CAMPO 6=
CAMPO 7=
CAMPO 8=
CAMPO 9=
CAMPO 10=
■
```

Al concluir las líneas de programa que gestionan la visualización hay que introducir algunas instrucciones adicionales que garanticen la continuación del programa. Siguiendo con la misma filosofía, lo lógico sería que se pasara al siguiente registro. Si se desea buscar otro registro se empleará el comando "@R", mientras que si se opta por finalizar la inspección se tecleará "@F".
Estas nuevas posibilidades se añaden al final de la rutina de visualización. Las adecuadas líneas de programa presentan una gran similitud con las que ejecutan una función análoga en las dos rutinas precedentes:

```
3200 PRINT "ENTER.....REGISTRO
     SIGUIENTE"
3210 PRINT "@R.....NUEVO REGISTRO"
3220 PRINT "@F.....FINALIZACION"
3230 INPUT "OPCION";B$
3240 IF B$="" THEN REG=REG+1:GOTO
3100
3240 IF B$="@R" THEN GOTO 3010
3240 IF B$="@F" THEN RETURN
3250 GOTO 3100
```

Las tres primeras instrucciones muestran en la pantalla los comandos a utilizar y definen su función. De esta forma, el operador siempre tendrá presentes las instrucciones que puede utilizar. La línea 3230 es la encargada de captar la orden que introduzca el usuario. A su vez, los comandos IF que siguen desvían la ejecución dependiendo de la orden recogida. Si el usuario no teclea ninguno de los comandos propuestos, se llega a la línea 3250 cuyo cometido es repetir la visualización.

AHORA TODO JUNTO

Las tres rutinas comentadas gestionan las tres acciones enunciadas al comienzo del presente capítulo. Sin embargo, éstas han sido concebidas como subrutinas de un programa principal. Lo que queda por programar es precisamente el programa principal.

Dado que las rutinas anteriores son independientes entre sí, el cuerpo principal del programa accederá a cada una de ellas de forma independiente. En definitiva, el cuerpo principal del programa debe proponer al usuario la elección entre uno de los tres procesos: creación, edición o visualización. De ello se encargará un "menú" visualizado en la pantalla.

La confección de menús es una tarea ya descrita en otro punto de la obra. De ahí que no profundicemos al respecto en este capítulo. Por el contrario, sí vamos a estudiar un nuevo comando que en determinados ordenadores realiza la misma función que el conocido PRINT AT.

Se trata del comando LOCATE, cuya especialidad es situar el cursor en un determinado punto de la pantalla. Tras ejecutarlo, el siguiente comando PRINT empezará a escribir a partir de dicho punto. En los dialectos BASIC que incluyen el comando LOCATE en lugar del PRINT AT, la instrucción PRINT AT 2,4 "HOLA" habrá de desdoblarse en dos: la de posicionamiento y la de impresión. La primera se resuelve con LOCATE 4,2 y la segunda con PRINT "HOLA". Obsérvese que los parámetros de LOCATE están intercambiados con respecto a los de la partícula AT. La razón de ello la encontramos en la distinta formulación de este nuevo comando:

(Número de línea) LOCATE <fila>, <columna>

Como ilustración práctica del uso del nuevo comando, y siguiendo con el ejemplo desarrollado en el presente capítulo, se incluyen a continuación las primeras líneas del menú.

```
200 REM MENU
210 LOCATE 5,10
220 PRINT "MENU"
230 LOCATE 10,5
240 PRINT "1.....ENTRADA DE DATOS"
250 LOCATE 12,5
260 PRINT "2.....EDICION DE DATOS"
270 LOCATE 14,5
280 PRINT "3.....VISUALIZACION
    DE DATOS"
```

Las referidas instrucciones llevarán a la pantalla las opciones de que dispone el usuario.

Recuerde que si su ordenador no incluye el comando LOCATE, puede realizar la

TABLA DE CONVERSION

ORDENADOR	POSICIONAMIENTO EN PANTALLA	
	PRINT AT (X, Y)<arg>	LOCATE Y, X
APPLE II (APPLESOFT)	— (2)	—
APRICOT (M-BASIC)	—	—
ATARI	—	POSITION X, Y
CBM 64	—	—
DRAGON	PRINT <n>, <arg> (1)	—
EQUIPOS MSX	—	LOCATE X, Y
HP-150	—	—
IBM PC	—	LOCATE Y, X
MPF	— (2)	—
NCR DM-V (MS-BASIC)	—	—
NEW BRAIN	—	—
ORIC	PRINT X, Y; <arg>	—
SHARP MZ-700 (MZ-BASIC)	—	—
SINCLAIR QL	—	CURSOR X, Y
SPECTRAVIDEO	—	LOCATE X, Y
ZX-SPECTRUM	PRINT AT Y, X; <arg>	—

(1) En el Dragon, el número <n> especifica una posición de la pantalla.
(2) Utilizan HTAB y VTAB para indicar las coordenadas horizontal y vertical respectivamente.

LOCATE

Sitúa el cursor en el lugar de la pantalla indicado en su argumento.

Formato: (n.° de línea) LOCATE <fila>,<columna>

Ejemplos: 10 LOCATE 10,8 : PRINT "HOLA"
50 LOCATE F,C : PRINT DATO

```
1 REM --------------------
2 REM ---BASE DE DATOS---
3 REM --------------------
10 DIM A$(100,10)
199 REM ------------
200 REM ---MENU---
201 REM ------------
210 LOCATE 5, 10
220 PRINT "MENU"
230 LOCATE 10, 5
240 PRINT "1.....ENTRADA DE DATOS"
250 LOCATE 12, 5
260 PRINT "2.....EDICION DE DATOS"
270 LOCATE 14, 5
280 PRINT "3.....VISUALIZACION DE DATOS"
300 LOCATE 20,5
310 INPUT "ELJA OPCION (1-3) ",N$
320 N=VAL(N$)
330 ON N GOSUB 1000,2000,3000
340 GOTO 200
997 REM ---------------
998 REM ---CREACION---
999 REM ---------------
1000 FOR REG=1 TO 100
1010 FOR CAM=1 TO 10
1015 PRINT "REGISTRO: ";REG;" CAMPO: ";CAM
1020 INPUT B$
1025 IF B$="@R" THEN LET CAM=10: GOTO 1040
1027 IF B$="@F" THEN LET CAM=10:LET REG=100:GOTO1040
1030 LET A$(REG,CAM)=B$
1040 NEXT CAM
1050 NEXT REG
1100 RETURN
2000 REM -------------
2001 REM ---EDICION---
2002 REM -------------
2010 INPUT "REGISTRO";REG
2020 INPUT "CAMPO";CAM
2030 PRINT "REGISTRO ";REG;"  CAMPO ";CAM
2040 PRINT "CONTENIDO: ";A$(REG,CAM)
2100 INPUT B$
2104 IF B$="" THEN GOTO 2120
2106 IF B$="@F" THEN RETURN
2108 IF B$="@R" THEN LET CAM=1:LET REG=REG+1: GOTO 2030
2110 LET A$(REG, CAM)=B$
2120 LET CAM=CAM+1
2130 IF CAM=11 THEN LET CAM=1:LET REG=REG+1
2140 IF REG <101 THEN GOTO 2030
2150  PRINT "NO HAY MAS DATOS"
2160 FOR I=1 TO 1000:NEXT I
2170 RETURN
3000 REM ---------------------
3001 REM ---VISUALIZACION---
3002 REM ---------------------
3010 INPUT "BUSCO EN EL CAMPO NUM.";CAM
3020 INPUT "QUE DATO";D$
3030 FOR REG=1 TO 100
3040 IF A$(REG,CAM)=D$ THEN GOTO 3100
3050 NEXT REG
3060 PRINT "NO ENCUENTRO ESE DATO"
3070 INPUT "BUSCO OTRO DATO (S/N)";B$
3080 IF B$="S" THEN GOTO 3010
3090 RETURN
3100 CLS
3110 PRINT "REGISTRO NUMERO ";REG
3120 PRINT
3130 FOR CAM=1 TO 10
3140 PRINT "CAMPO ";CAM;"=",A$(REG,CAM)
3150 NEXT CAM
3200 PRINT "ENTER.....REGISTRO SIGUIENTE"
3210 PRINT "@R.....NUEVO REGISTRO"
3220 PRINT "@F.....FINALIZACION"
3230 INPUT "OPCION";B$
3240 IF B$="" THEN REG=REG+1:GOTO 3100
3240 IF B$="@R" THEN GOTO 3010
3240 IF B$="@F" THEN RETURN
3250 GOTO 3100
```

misma función con PRINT AT. Por supuesto, también puede adecuar el menú a su pantalla variando las coordenadas de LOCATE (o de PRINT AT).

Después de presentar el menú, es preciso recoger la opción elegida por el usuario. Tarea que se resuelve con un nuevo comando INPUT. Para situar el punto de entrada del dato volvemos a hacer uso de LOCATE.

```
300 LOCATE 20,5
310 INPUT "ELJA OPCION (1-3),",N$
320 N=VAL(N$)
330 ON N GOSUB 1000,2000,3000
340 GOTO 200
```

En el argumento de INPUT figura una variable de cadena. Su presencia evitará la generación del mensaje de error correspondiente a la introducción accidental de una letra. La línea 320 extrae de N$ el dato numérico, depositándolo en la variable N. Esta es la variable que se utiliza en la línea 330 para el salto a la subrutina adecuada. En el caso de no aportar un dato correcto, el GOTO de la siguiente línea hará que la secuencia de ejecución retorne a la línea 200 para visualizar de nuevo el menú de opciones.

Forth (1)

La filosofía del lenguaje Forth

Todos los lenguajes de programación acogen una serie de características primordiales que los hacen más o menos idóneos para determinadas aplicaciones. Tales características derivan de la propia filosofía del lenguaje, y en menor grado de la presencia o ausencia de determinadas instrucciones dentro de su vocabulario.

El FORTH es un lenguaje que se caracteriza por permitir un gran aprovechamiento de todas las posibilidades y recursos de la máquina. Son muchas las facultades del FORTH que lo acercan a los lenguajes de bajo nivel; y ello, tratándose de un lenguaje de alto nivel, es ya decir mucho. Semejante propiedad hace que gran parte de sus instrucciones no sean todo lo potentes que cabría esperar del repertorio de un lenguaje de alto nivel; aunque, bien es cierto, que sus virtudes compensan con creces este inconveniente. No terminan aquí sus peculiaridades. La mayor parte de los lenguajes de alto nivel se suelen clasificar en compilados o interpretados; si bien, algunos (el BASIC, por ejemplo) admiten ambas posibilidades. El FORTH no se ajusta a esta clasificación. Su peculiar estructura no es ni compilada ni interpretada. Un programa FORTH se compone de una serie de rutinas y, a su vez, el mismo programa es una rutina más con la que es posible construir nuevos programas. Pues bien, estas rutinas, una vez creadas son compiladas, y quedan así en la memoria del ordenador, aunque el trabajo con el lenguaje se realiza en modo interpretado. En el ámbito del FORTH anidan tres o cuatro conceptos fundamentales que difieren de los usuales en otros lenguajes de programación. Estos conceptos no son especialmente difíciles, pero son nuevos, y familiarizarse con ellos puede llevar algún tiempo. Ante todo, hay que borrar de la mente ideas preconcebidas: en realidad, el FORTH es un lenguaje tan fácil como cualquier otro.

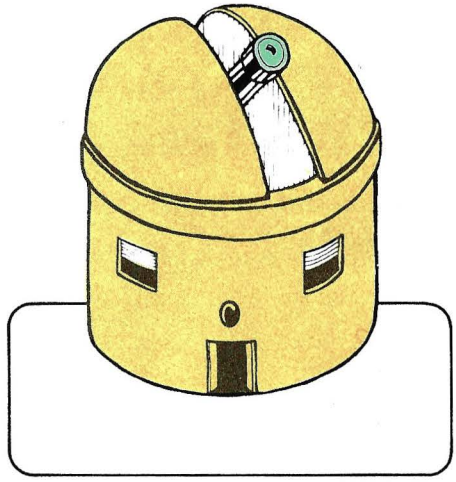
EL CONCEPTO DE PILA

El primer concepto a definir es el de *pila*. Para enunciarlo, lo más cómodo es buscar un ejemplo en la vida real. Y no es difícil encontrarlo: por ejemplo, un apilamiento de platos fregados es una pila.

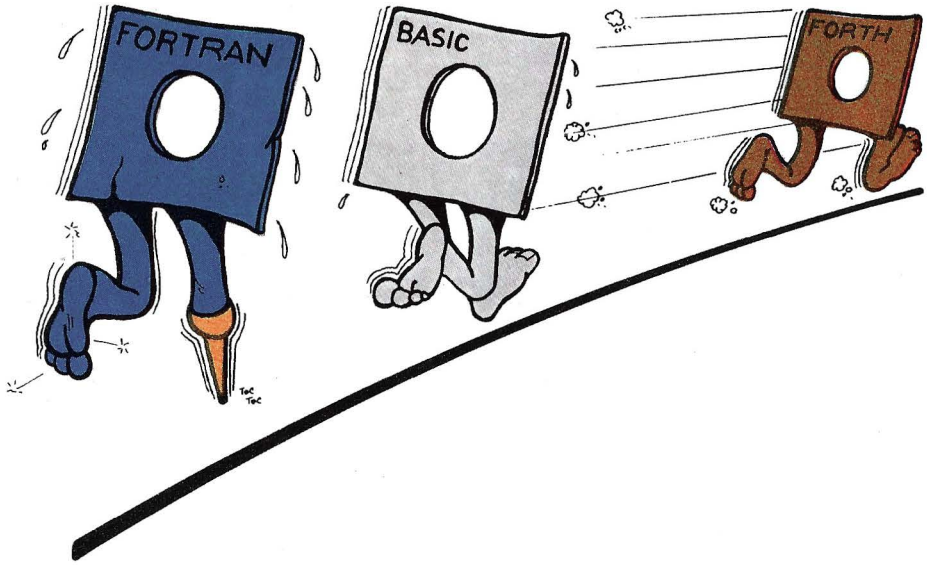
Observemos la forma de trabajar con uno de estos apilamientos. En primer lugar, una vez fregados, los platos se van depositando en la pila... ¿Pero en qué lugar de la pila? Resulta evidente que en la parte baja no se pueden colocar. Por tanto sólo es posible colocarlos en la cima.

¿Y de dónde iremos recogiendo los platos a medida que los vayamos necesitando? Pues de la parte baja desde luego que no: la pila podría terminar en un verdadero estropicio. No cabe duda que habrá que irlos tomando, ordenadamente, de la zona superior, de la cima de la pila.

Este es, precisamente, el modo de traba-



El Forth nació como una herramienta de ayuda con la que formular los complicados cálculos necesarios en los observatorios astronómicos.

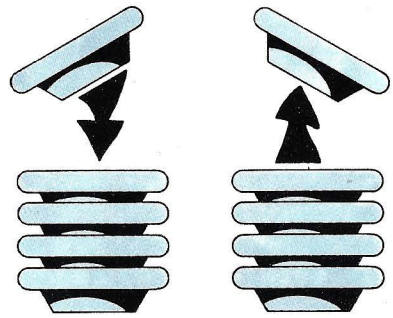


El FORTH es un lenguaje que destaca por su celeridad: tanto por la velocidad de ejecución de los programas, como por la rapidez con la que es posible desarrollar nuevos programas, partiendo de otros programas y rutinas ya existentes.

jar con la pila. La pila es una zona de memoria en la que los datos se almacenan y se extraen de la misma forma que los platos de nuestro ejemplo. La estructura de almacenamiento de la pila es conocida con el nombre de *lista LIFO* (last input first output); o lo que es lo mismo, pero en castellano: el primero en entrar es el primero en salir. Esta estructura condiciona la forma de trabajo con este lenguaje.
Realmente, el FORTH da entrada a dos pilas: en una de ellas se almacenan los datos con los que vamos a trabajar, se trata de la pila normal; la segunda es la llamada pila de retorno, y se emplea para almacenar valores de índices y variables de bucles. Desde luego, además de estos dos, hay que contar con la habitual pila del sistema.

PALABRAS

La palabra es uno de los conceptos básicos del FORTH y tal vez el más importante. Del mismo modo que en el lenguaje corriente bautizamos con una palabra a las cosas que todavía no tienen un nombre, en FORTH podemos bautizar a una secuencia de acciones o instrucciones por medio de una palabra.
De esta forma, al emplear una palabra, lo que haremos es referirnos al nombre con el que hemos bautizado a una serie de acciones. De ello se deduce que una palabra es un ente abstracto que asocia una serie de acciones o "definición" a un nombre. Una palabra FORTH consiste en un conjunto de caracteres delimitados por espacios. Dada la importancia que tienen en el FORTH los espacios, en su función de separadores, una palabra no puede contener ningún espacio en su interior. En principio, las palabras pueden ser de cualquier longitud, pero sólo los primeros caracteres serán significativos (por lo general, los 31 primeros).
De lo explicado hasta ahora, se puede deducir fácilmente que un programa FORTH es extraordinariamente modular. En efecto, la confección de un programa consiste en ir construyendo nuevas estructuras basadas en estructuras antiguas ya existentes. A esto hay que unir el hecho de que estructuras imprescindibles en otros lenguajes, como el GOTO, carecen



Secuencia de entrada y salida de datos de la pila.

aquí de sentido, con las consiguientes ventajas.

EL DICCIONARIO

Cuando se creó el lenguaje FORTH, se pretendía garantizar el rápido acceso a un catálogo de funciones y poder definir otras nuevas funciones a partir de las ya existentes.
Recordemos que la palabra constituye el elemento esencial del FORTH e implica una acción determinada.
El conjunto de palabras FORTH conforma el diccionario. Un diccionario que puede enriquecerse mediante la creación de nuevas palabras que sumar a las ya existentes.

NUMEROS

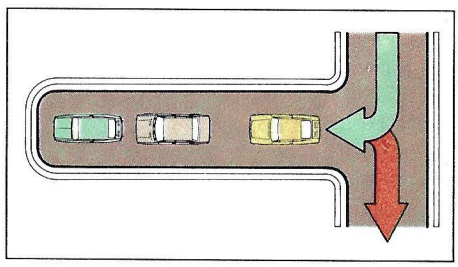
El ordenador, al analizar lo que haya introducido el usuario, intentará, en primer lugar, asimilarlo como una palabra; si no lo consigue, entenderá que se trata de un número.
Los números pueden expresarse en cualquier base, de la 2 a la 36. En todo caso, al proceder a su conexión, el equipo trabajará en decimal, el sistema de numeración más común; no obstante, en cualquier momento puede ordenarse un cambio de base.
Los números pueden ser de diversos tipos; por ejemplo, enteros de simple o doble precisión. Los de precisión simple sólo pueden llegar al valor 32767 para positivos y a 32768 para negativos.

NOTACION POLACA INVERSA

Esta notación representa otra de las particularidades del FORTH. Consiste en suministrar el operador aritmético tras los datos sobre los cuales habrá de operar. Por ejemplo, para sumar dos números, la operación sería la siguiente:

```
2 2 +
```

La explicación es muy sencilla si pensamos en la "pila". Los datos son tomados y almacenados en la pila y, a continuación, cuando el procesador observa la palabra "+" depositada en el punto superior de la pila, realiza la acción de tomar los dos primeros números que encuentre en la



Un callejón sin salida es también un ejemplo ilustrativo de una "pila": el último coche en entrar ha de ser el primero en salir.

pila (retirándolos de la misma), los suma y coloca el resultado en la cima de la pila.

ARITMETICA

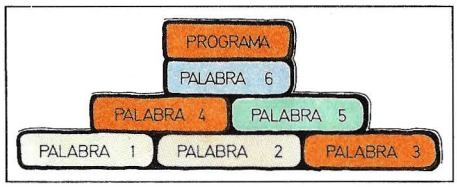
Como ya se ha indicado, los operandos han de estar depositados en la pila antes de que se realice la operación. Por ejemplo:

```
5 27 + . <CR>
```

5: coloca cinco en la pila.
27: pone el valor veintisiete en la pila.
+: suma los dos números de la parte superior de la pila y los elimina de la misma, depositando el resultado de la operación en la cima de la pila.
.: es la palabra FORTH que se emplea para escribir en la pantalla el valor que se encuentra en la cima de la pila.

ASPECTO DE UN PROGRAMA FORTH

En el programa FORTH cuyo listado se adjunta, aparece un elevado número de signos de puntuación y una peculiar distribución de los espacios.
Un punto muy importante en todo programa FORTH es la correcta colocación de los espacios en blanco. Los espacios en blanco son los separadores más importantes. De la adecuada colocación de estos espacios depende la interpretación de las palabras introducidas. De cualquier forma, el editor se encargará normalmente de advertirnos del uso de palabras no existentes en su diccionario.



El Forth es un lenguaje fundamentado en la creación de palabras que, posteriormente, pueden ser empleadas para crear nuevas palabras más evolucionadas.

OPERADORES ARITMETICOS

Los operadores aritméticos del FORTH también pertenecen a la categoría de las palabras. Se caracterizan por su peculiar forma de trabajo con la pila; de ahí que una parte importante de su explicación haya de versar sobre la manipulación de la pila. Veamos cuáles son los operadores aritméticos más significativos del FORTH:
+ Obtiene la suma de los dos elementos que lo siguen en la pila, y deposita el resultado en la cima de la misma.

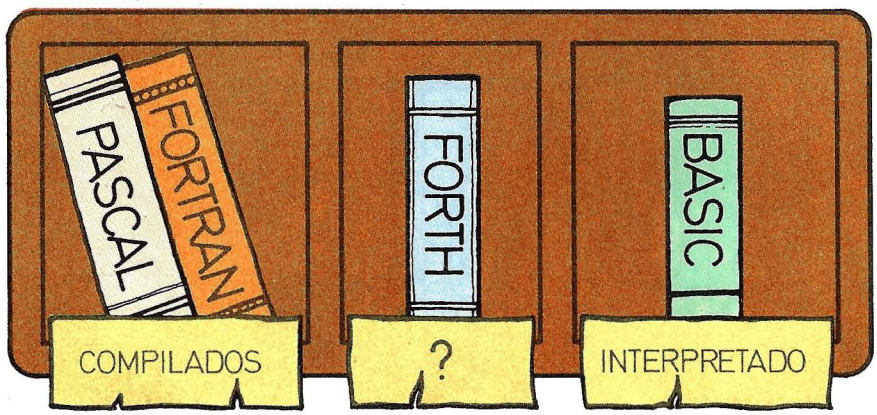
```
2 2 + . <CR>
2 2 + . 4 OK
■
```

Estas operaciones se pueden encadenar, y para hacerlo hay que tener en cuenta el modo de trabajo de la pila. Veamos, por

Historia del FORTH

Aunque el lenguaje FORTH ya existía anteriormente en forma de rutinas sueltas, este no pasó a convertirse en un auténtico lenguaje de programación normalizado hasta el año 1969. Fue Charles H. Moore, del National Radioastronomy Observatory, localizado en Charlottesville (Virginia U.S.A.), quien lo desarrolló y propuso como herramienta de trabajo destinada a realizar programas para controlar los grandes radiotelescopios empleados en astronomía. En aquellos tiempos, el dominio correspondía a los grandes ordenadores. Y sobre estos ordenadores nació el FORTH. Más tarde, con la aparición de los microordenadores, el FORTH ha recibido un notable impulso al adaptarse fácilmente a ellos. De hecho, existen ordenadores domésticos que emplean el FORTH como lenguaje residente; tal es el caso del JUPITER ACE.
Este lenguaje está orientado hacia el desarrollo rápido de programas. Sus primeras aplicaciones se concretaron en el campo de la astrofísica. Campo en el que la aportación de este lenguaje ha sido muy grande, llegando a ser adoptado como lenguaje oficial por la Unión Astronómica Internacional.
En la actualidad, otro terreno que parece abonado para el FORTH es el control de procesos industriales, y dentro de él, el mundo de la robótica. Sus facultades en este ámbito derivan de la facilidad para crear rápidamente un programa a partir de una serie de rutinas independientes.
La filosofía del FORTH se distancia de la propia de otros lenguajes de uso más común; razón ésta por la que ha sido considerado siempre como un lenguaje difícil. Lo cierto es que se trata de un lenguaje ciertamente peculiar; sin embargo, no resulta mucho más difícil de aprender que cualquier otro si se parte de cero. En todo caso, sus características de modularidad, estructuración e interactividad, hacen del FORTH un lenguaje de indudable interés y atractivo.

Pasado algún tiempo desde su nacimiento, y en un intento de unificar criterios con respecto a este lenguaje, apareció el F.I.G. Forth (Forth Interest Group). A esta siguieron otras versiones, más recientes, cuyo número da idea del creciente interés que existe por este lenguaje. En la actualidad existe una firma americana, FORTH INC, dedicada a la comercialización de todo lo relativo a este interesante lenguaje. Cabe afirmar que hoy en día pueden encontrarse versiones de FORTH para prácticamente todos los microordenadores del mercado.



La peculiar estructura del FORTH no es ni interpretada ni compilada... ¡sino todo lo contrario!

ejemplo, la forma de realizar dos sumas asociadas de distinta forma:

```
2 3 3 + + . <CR>
2 3 4 + + . 8 OK
```

Operación que equivale a la siguiente:

```
2 3 + 3 + . <CR>
2 3 + 3 + . 8 OK
```

NOTA: En FORTH, el punto (.) sirve para mostrar por pantalla el contenido de la cima de la pila. El dato visualizado desaparece de la cima de la pila.

— Resta el segundo número introducido en la pila del introducido en primer lugar.

```
5 3 - . <CR>
5 3 - . 2 OK
```

* Multiplica los dos números situados en las posiciones superiores de la pila. Deposita en la cima de la pila el resultado de la multiplicación.

```
16000 2 * . <CR>
16000 2 * . OK
```

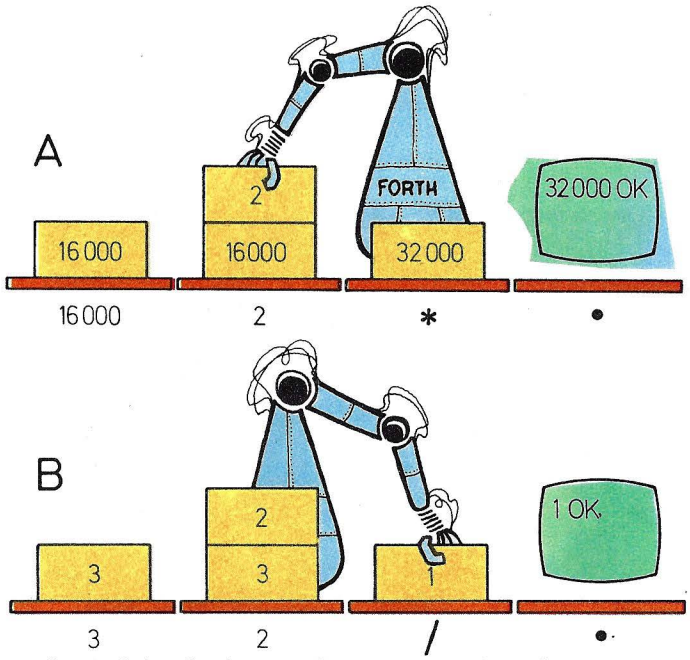
/ Obtiene la división de los dos números de la cima de la pila, considerando como divisor al más cercano a la cima (el último que se introdujo) y al otro como dividendo. El resultado de la operación sólo incluye el módulo del cociente.

Programa en FORTH

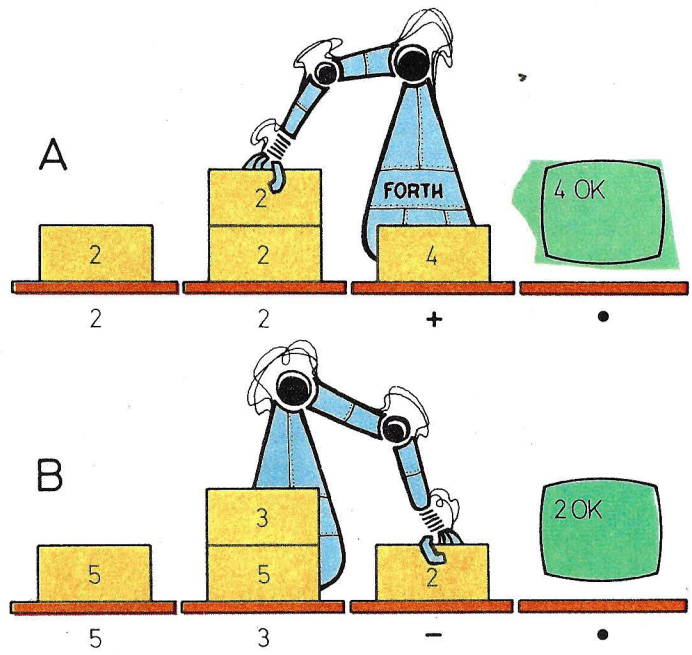
```
: FORMATEO
SWAP 9 + 12 / MOD
1 - ROT +
;
: ANO
DUP 4 / +
;
: MES
SWAP 306 * 5 + 10 / +
;
: DIA
+2 + 7 MOD 1 +
;
: FECHA
FORMATEO ANO MES DIA
;
```

```
3 2 / . <CR>
3 2 / . 1 OK
```

En sucesivos capítulos dedicados a este interesante lenguaje tendremos ocasión de estudiar las restantes palabras aritméticas del FORTH.



Evolución de la pila al ejecutar las operaciones de producto (A) y división (B).



Evolución del contenido de la pila a ejecutar las operaciones de suma (A) y resta (B).

Apple ProDOS (1)

La alternativa de Apple para disco rígido

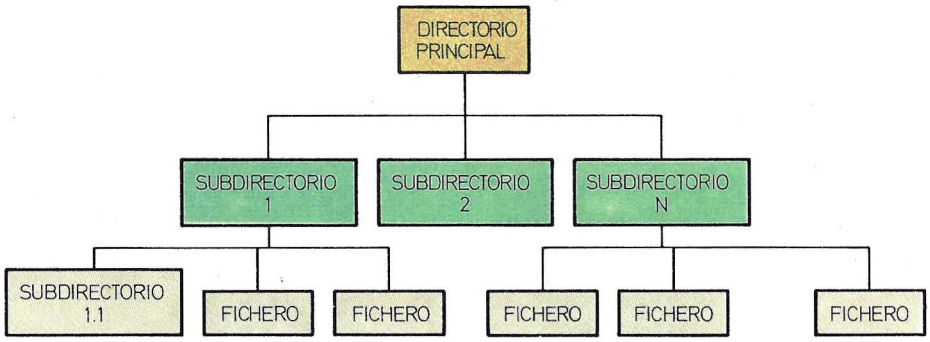
Apple lanzó al mercado su familia de ordenadores Apple II y con ella un sistema operativo simple y sencillo denominado AppleDOS. Este permitía el control de las unidades de disquete conectadas al ordenador. Habida cuenta que su destino inicial era el ámbito doméstico o de "hobby", dicho sistema operativo resultó un éxito, ya que era simple de comprender y utilizar.

El creciente ritmo de la informatización, así como la constante innovación tecnológica, pusieron a disposición de los pequeños usuarios una amplia lista de periféricos que suponían nuevos requerimientos para sus equipos. De estos periféricos, uno de los más importantes y de mayor efecto sobre la filosofía y necesidades de estos pequeños usuarios fue el disco rígido. Su gran capacidad de almacenamiento, alta fiabilidad y velocidad de acceso, abrió a estos usuarios la posibilidad de poner en marcha aplicaciones anteriormente vedadas.

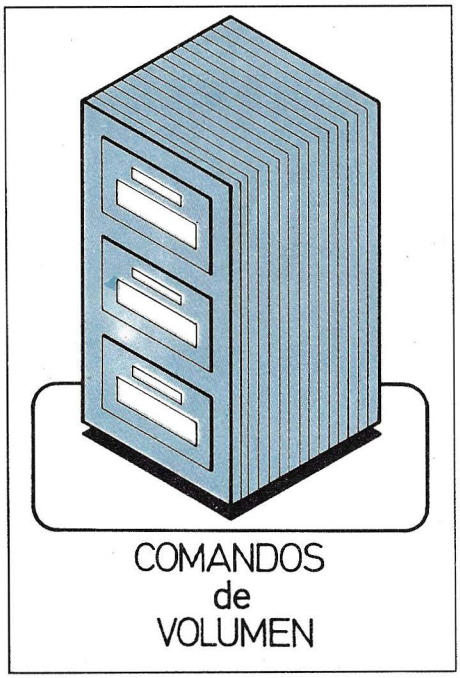
Aunque el AppleDOS operaba adecuadamente con una, dos o más unidades de disquete, la conexión de un disco rígido excedía su capacidad y evidenciaba toda serie de limitaciones. La más importante residía en el hecho de dividir el disco rígido en el equivalente a varios disquetes "lógicos", lo que provoca una disminución en el tamaño efectivo de los ficheros, ya que éstos no pueden rebasar el tamaño de los disquetes lógicos.

Asimismo, esta división suponía dispersar la información sobre el disco, de tal forma que pudiera acomodarse al tamaño de estos disquetes lógicos. En definitiva, se producía un importante desperdicio del espacio de almacenamiento efectivo.

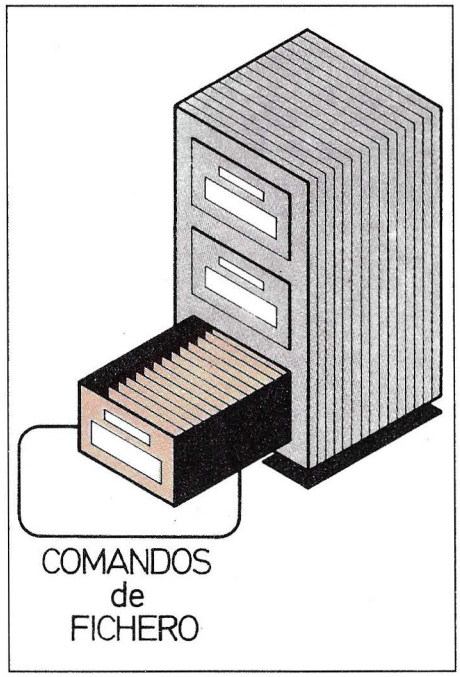
Una persona que tuviese un ordenador por hobby podía disculpar hasta cierto punto estas deficiencias, en base a la sencillez de manejo del sistema operativo. Si bien, otros usuarios que cobraban cada vez mayor importancia en el contexto de la microinformática, como los profesionales liberales, arquitectos, médicos o minoristas, se veían rechazados por estas limitaciones. En efecto, una característica pri-



El sistema operativo ProDOS permite estructurar los ficheros de forma jerárquica.

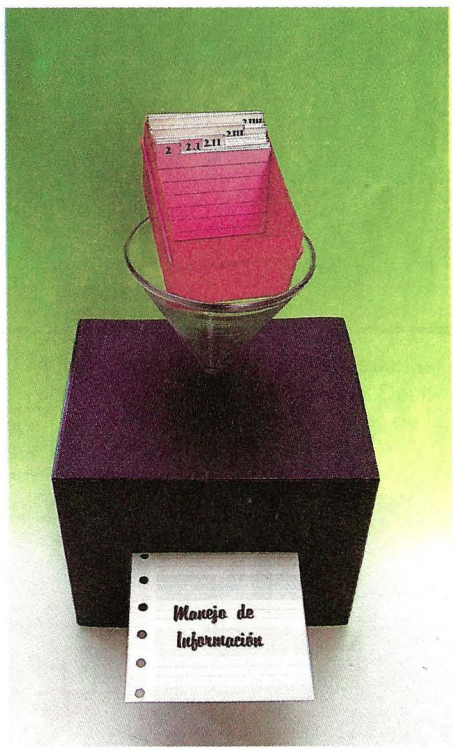


Los comandos de volumen actúan sobre el conjunto de informaciones contenidas en un soporte de almacenamiento externo.



Los comandos de fichero actúan únicamente sobre los ficheros residentes en las unidades de almacenamiento.

mordial y definitoria de sus necesidades reside en poder almacenar cantidades masivas de información en un mismo fichero (para un médico la lista de clientes y sus direcciones; para un minorista, la lista de productos en stock junto con los proveedores que suministran cada uno de ellos).
Considerando todos estos problemas que afectaban a su posición en el mercado, la firma Apple lanzó un nuevo sistema operativo orientado a la gestión de discos rígidos: el Apple ProDOS. Aunque muchos de sus comandos coincidan con los del AppleDOS, el ProDOS no es una mera extensión de éste. Como característica más importante destaca el que puede direccionar hasta 32 Mbytes en disco con una sola unidad lógica, eliminando de esta manera los problemas asociados con el disco rígido en el caso del AppleDOS. Permite, asimismo, la creación de directorios y subdirectorios, con lo cual se genera una jerarquía entre los ficheros que permite un más fácil acceso a los mismos.
El sistema es, al igual que su predecesor, sumamente simple de utilizar; si bien, no es totalmente compatible con el mismo. Existen, sin embargo, ciertas utilidades que permiten la conversión de los ficheros pertinentes a ciertas versiones de AppleDOS a ficheros ProDOS. Quizás esta falta de compatibilidad directa sea el óbice más importante de este sistema operativo, junto con el hecho de no poder utilizar los discos rígidos de alta capacidad que no estén equipados con el controlador propio de Apple.



El sistema operativo ProDOS ofrece una notable versatilidad y eficacia en la gestión de informaciones, ya sea en disco flexible o rígido.

OPERANDO CON EL PRODOS

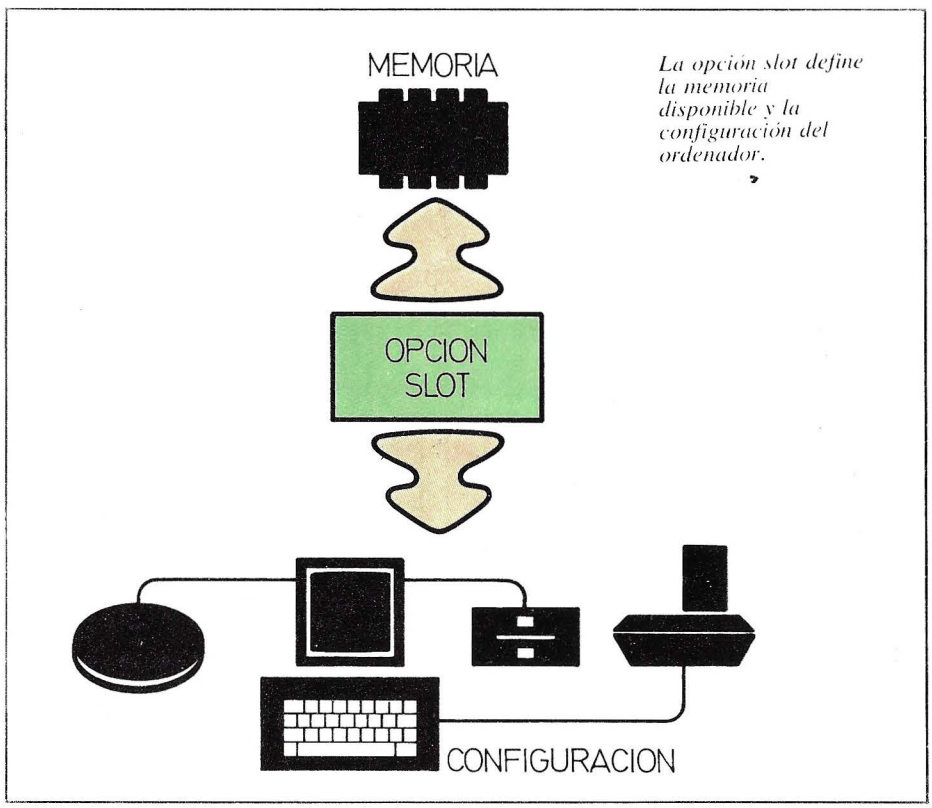
Una característica importante de este sistema operativo es, como ya hemos indicado, su facilidad de empleo. Tal facilidad viene dada por el hecho de operar íntegramente a través de menús. Al usuario se le presentan en la pantalla una serie de opciones, algunas de las cuales provocan una respuesta inmediata, mientras que otras generan una nueva lista de opciones entre las cuales habrá que efectuar una nueva selección; el proceso se repite hasta que la tarea que deseemos llevar a cabo se ejecute finalmente.
Una analogía ilustrativa de este método, cabe encontrarla en la selección de un billete de avión para realizar un viaje en una fecha y a destino determinados. Primero, se nos presenta una lista de las compañías aéreas donde elegir. Una vez seleccionada la compañía, hay que consultar la lista de vuelos correspondientes a la fecha indicada, y escogido uno de estos, hay que optar entre las diversas categorías de asientos disponibles; tras esta última selección, el proceso estará completo.
El menú básico de este sistema operativo incluye seis opciones. Cada una de ellas, dependiendo de su objetivo, genera submenús con nuevas opciones. Veamos, someramente, cuáles son estas opciones básicas.

- TUTOR: ProDOS Explanation

Esta opción revela las funciones del sistema operativo, así como la forma de llevarlas a cabo, y permite al usuario tener un cierto conocimiento de sus interioridades.

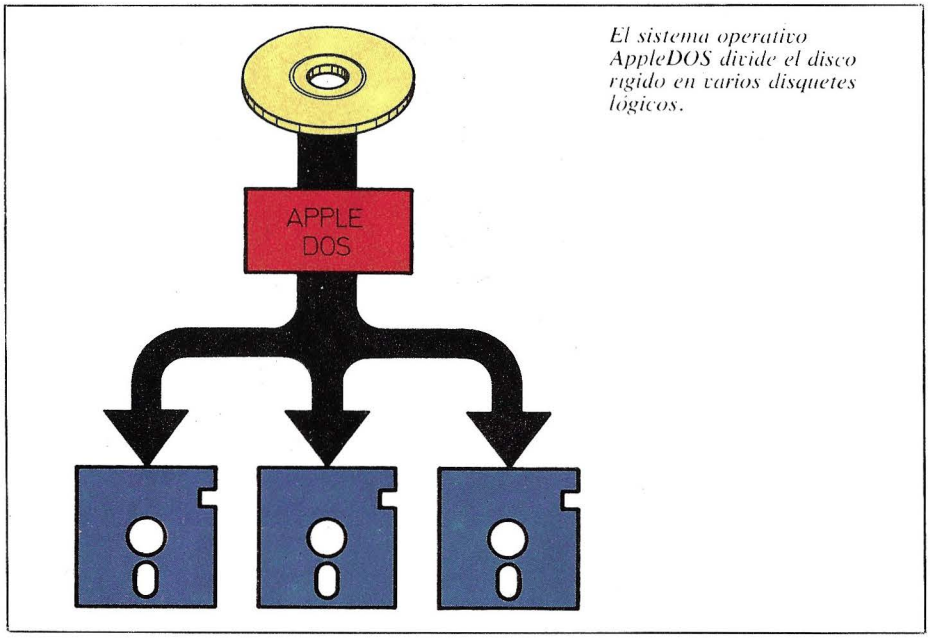
- ProDOS Filer

Su cometido radica en el control exhaustivo de la información almacenada en los discos o disquetes. Para llevarlo a cabo tiene a su disposición dos grandes grupos



La opción slot define la memoria disponible y la configuración del ordenador.

de comandos: los comandos de volumen y los comandos de fichero.
Los comandos de volumen operan con el elemento de almacenamiento como un todo, permitiendo copiar disquetes o discos enteros, formatear los discos, etc. Los comandos de ficheros ejecutan tareas similares, aunque esta vez sobre los ficheros contenidos en el disco o disquete. En esencia, permiten organizar la información contenida en dichos soportes; así, facilitan las tareas de copiar el contenido de diferentes ficheros, editar y modificar la información contenida en cada uno de ellos...



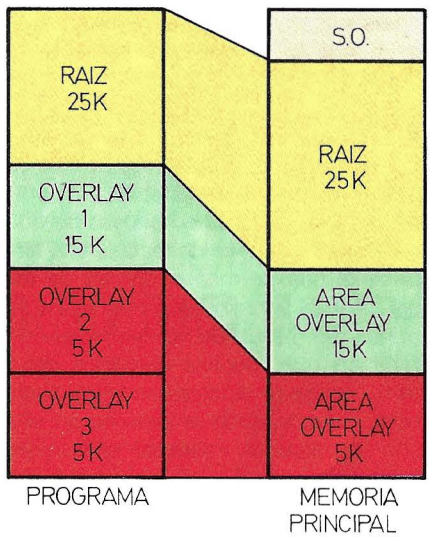
El sistema operativo AppleDOS divide el disco rígido en varios disquetes lógicos.

- DOS-ProDOS Conversion

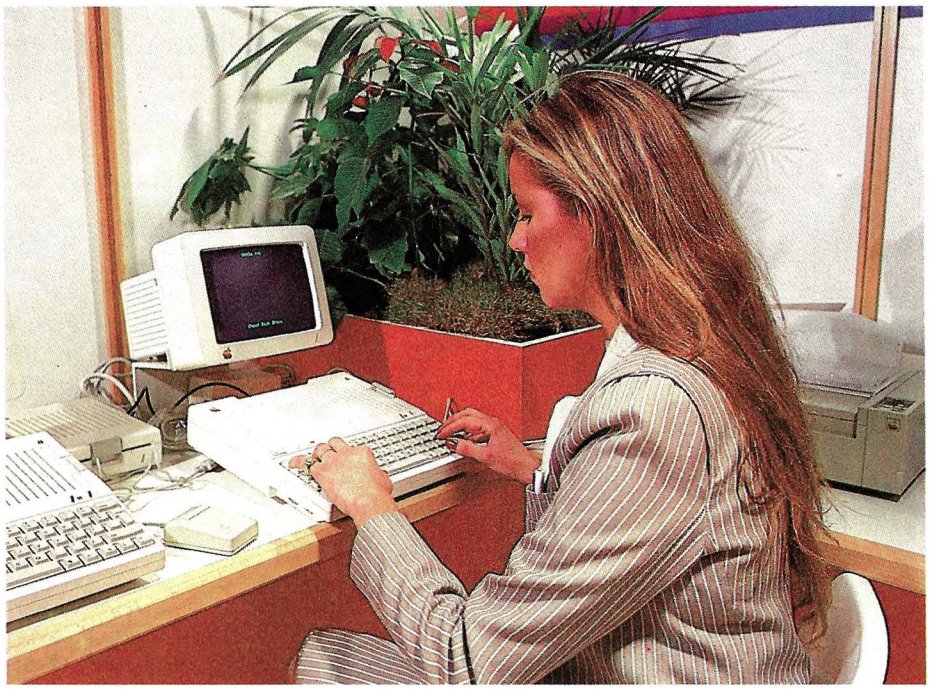
Esta opción reviste una gran importancia, ya que permite la conversión de programas y aplicaciones creadas para su uso con el sistema operativo AppleDOS a un formato que permita su compatibilidad con el ProDOS. Este proceso también puede repetirse en sentido inverso, esto es: pasar del formato ProDOS a otro compatible con Apple DOS.
Esta conversión se lleva a cabo contemplando la versión 3.3 del Apple DOS, no preservando la compatibilidad con versiones anteriores de este sistema operativo.
Existe, sin embargo, una utilidad, denominada Muffin, mediante la cual es posible transformar ficheros con el formato propio de la versión 3.2 del Apple DOS, al formato de la versión 3.3. Una vez acondicionados a este último formato, ya pueden ser tratados por el programa de conversión al formato del ProDOS.
Ciertamente, esta opción del menú permite aliviar una de las debilidades que se señalaban respecto al ProDOS: su no compatibilidad con el sistema operativo del que deriva. Sin embargo la transformación no es completa, ya que se limita a las versiones del DOS señaladas y, en algún caso, aun es necesaria la colaboración de paquetes de software adicionales.

Almacenamiento virtual

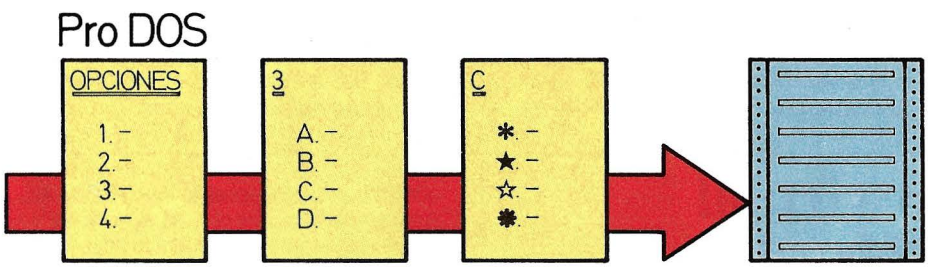
Uno de los problemas con los que se enfrenta un programador radica en el hecho de que, en ocasiones, el espacio de memoria principal disponible para los programas y datos es demasiado reducido para responder a las necesidades de una determinada aplicación. Frente a tal problema sólo caben dos soluciones. Una es la optimización del programa, de forma que disminuya el espacio de memoria principal necesario para su almacenamiento. En el caso que este método no dé resultado, no quedará otra alternativa que considerar el empleo de un ordenador con capacidad para la gestión de memoria virtual.
La memoria virtual resulta de introducir en el ordenador un hardware y software que permitan utilizar los elementos de almacenamiento masivo para almacenar, durante el funcionamiento de la aplicación, aquellos programas que son demasiado grandes para ser memorizados internamente. Mediante su empleo, el programa es dividido en una serie de segmentos que son cargados en el ordenador a medida que es necesaria su presencia.



Un primer intento en esta dirección fue el empleo de lo que se denomina "overlay". Según este método, una porción del programa (la raíz) permanece en memoria constantemente, mientras que el resto del programa se divide en porciones de igual tamaño a la denominada "área de overlay" de la memoria principal. Durante la operación, una parte del programa se carga desde el disco en el área de overlay, y se ejecuta. Cuando la ejecución termina, se pasa el control al área del programa; se carga una nueva porción del programa en el área de overlay y se repite el proceso.
El método de almacenamiento virtual es un refinamiento del método de overlay. En él, el programador no tiene necesidad de dividir el programa en segmentos y gestionarlos. Un hardware y software especializados dividen el programa en una serie de porciones denominadas páginas de un determinado tamaño. Estas páginas se almacenan sobre disco rígido y son cargadas en memoria principal y gestionadas de forma similar a la indicada para el método anterior. De lo señalado, resulta evidente que este método sólo es posible en ordenadores de una cierta envergadura y equipados con disco rígido.



El ProDOS es el sistema operativo básico del modelo más reciente de la familia de ordenadores Apple II: el Apple IIc.



En el ProDOS, la gestión de las tareas realizables se realiza a través de un conjunto de menús encadenados.

● Slot Assignements
Una opción sumamente útil: presenta una visión global de la configuración del ordenador que estamos empleando, indicando la memoria disponible y los periféricos conectados a los diversos "slots" de expansión.
Un "slot" de expansión es un conector, localizado en el equipo, que permite la comunicación entre los periféricos y la CPU a través de las adecuadas tarjetas de control insertadas en él. La identificación de disquetes y discos se establece a través del número del slot; si bien, en el caso de las unidades de disquete hay que indicar el número asociado a cada una de ellas, ya que a cada slot pueden conectarse dos de estas unidades numeradas como 1 ó 2. En estas condiciones, tenemos cuatro unidades de disquete y un disco rígido conectados al ordenador se identificarán de la siguiente forma: "disco rígido slot X", "unidades de disquete 1 y 2 del slot Y" y "unidades de disquete 1 y 2 del slot Z".

● Display / Set time
Por medio de esta opción es posible establecer manualmente la fecha y la hora actuales. Esto sucede en el caso de que el ordenador no tenga conectada una tarjeta de reloj que permita el control de estos parámetros por sí mismo y de forma independiente. La utilidad de esta opción estriba en que muchas utilidades del sistema emplean la información de fecha y hora, memorizando el instante en el que son utilizadas. A través de esta información, es posible distinguir entre diferentes versiones de una misma información, ya que cada una de ellas incluirá la fecha en que fue generada.

● Applesoft BASIC
A través de esta opción se entra en el ámbito del lenguaje BASIC. Para regresar al control del sistema operativo, bastará con teclear RUN STARTUP.

GESTION INTERNA DEL PRODOS

El sistema operativo ProDOS está constituido por una serie de programas de sistema, denominados respetivamente:

PRODOS
BASIC.SYSTEM
STARTUP
FILER
CONVERT

Cada uno de estos programas lleva a cabo una función determinada, controlando un bloque de actividades del sistema operativo. Sin embargo, no pueden ser almacenados en su conjunto dentro de la memoria principal del ordenador; por lo que, dependiendo de la función a realizar, será preciso almacenar el correspondiente programa en memoria.
Durante la secuencia de arranque en frío, se carga inicialmente en memoria el programa PRODOS; éste actúa como núcleo del sistema y permanece siempre en memoria. A continuación se carga el programa BASIC.SYSTEM que contiene el traductor de lenguaje BASIC. Acto seguido, éste segundo programa busca en el disquete que contiene el sistema operativo al programa STARTUP, el cual contiene el menú principal, e inicia la carga del mismo en memoria. Tras ello, el proceso de arranque en frío finaliza y el ordenador queda en disposición de ser utilizado.
Los programas CONVERT y FILER corresponden a sendas opciones del menú principal: DOS-ProDOS conversión y ProDOS Filer, respectivamente. Cada vez que estos programas son invocados, toman en la memoria el lugar ocupado por los dos programas BASIC.SYSTEM y STARTUP. Cuando termina de operar con ellos y se regresa al menú principal, los programas BASIC.SYSTEM y STARTUP vuelven a ser cargados en memoria. Tal y como está estructurado el sistema operativo, no es posible pasar de los programas FILER a CONVERT directamente, ya que el menú principal STARTUP ha de actuar de puente entre ellos.

Software de comunicaciones

Conceptos básicos sobre comunicaciones

El número de dispositivos "enchufables" a un ordenador es cada vez mayor y más variado. Un primer problema asociado a esta difusión es la necesidad de que el ordenador y los dispositivos se entiendan, es decir, se comuniquen. Otro de los objetivos más acuciantes en el mundo de la Informática es la posibilidad de conectar ordenadores situados a grandes distancias; de nuevo estamos hablando de comunicación.

Obviamente, para poder permitir cualquiera de los dos tipos de comunicación descritos en el párrafo anterior, se deben conjugar dos elementos: unos físicos por donde fluirá la información y otros lógicos. Estos últimos suelen englobarse bajo el apelativo de *software de comunicación.*

En el presente capítulo vamos a iniciar el estudio de este tipo de programas; para ello, antes de comenzar a evaluar las características de determinados paquetes comerciales, vamos a definir algunos conceptos básicos en comunicaciones.

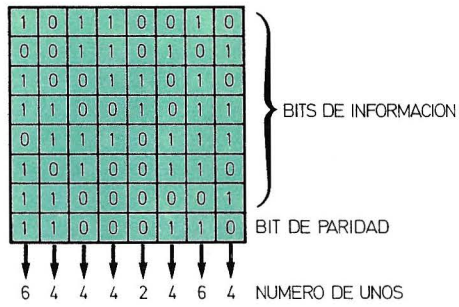
ALGUNAS PREGUNTAS BASICAS

- ¿Qué se transmite?

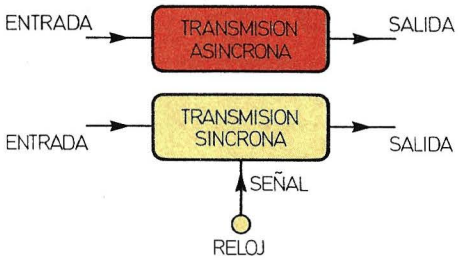
La información dentro de un ordenador se basa siempre en dígitos binarios, es decir, en ceros y unos. Por lo tanto, los caracteres que van a permitir establecer la comunicación en todos los casos van a ser dos: "0" y "1".

- ¿Qué puede representar una cadena de ceros y unos?

Cualquier cosa. Para ello se utiliza un sistema de codificación que, básicamente, consiste en representar cualquier carácter mediante una serie fija de dígitos binarios.



El bit de paridad se elige de forma que el número total de "unos" de cada carácter sea siempre par o impar; de esta forma será posible detectar los errores cometidos en la transmisión.



Las comunicaciones de tipo síncrono están acompasadas por las señales procedentes de un reloj, éste dicta los intervalos en los que se realiza cada intercambio elemental de información. Por el contrario, en las comunicaciones de tipo asíncrono cada intercambio elemental se produce al terminar el anterior.

- ¿Existe un único sistema de codificación?

Desgraciadamente no; la existencia de innumerables sistemas de codificación es uno de los problemas que debe resolverse para comunicar a dos ordenadores que trabajan con distintos códigos.

- ¿Cuáles son los principales sistemas de codificación?

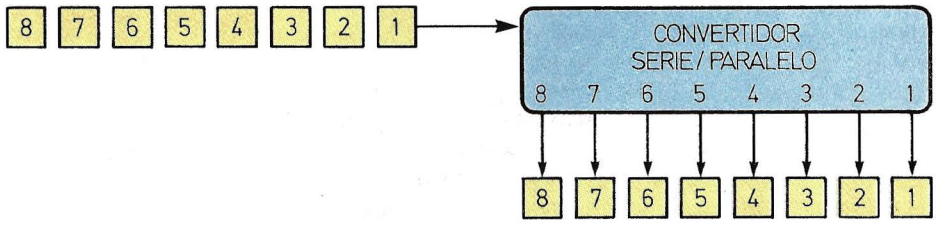
Podemos destacar dos. El código ASCII que corresponde a las siglas de "American Standard Code of Information Interchange"; utiliza series, normalmente, de 7 bits para codificar cualquier carácter. El segundo sistema de codificación entre los más difundidos es el denominado EBCDIC; éste utiliza series de 8 bits para codificar los diversos caracteres. El prinpal promotor de este sistema de codificación es la firma IBM.

- ¿A qué se llama carácter?

A un conjunto de bits que representa a un símbolo específico. Este es uno de los principales motivos de confusión; según estemos hablando de códigos ASCII o EBCDIC, un carácter estará formado por 7 u 8 bits.

- ¿Existe alguna relación entre carácter y sistema de codificación?

Evidentemente sí. Como ya apuntábamos



Los convertidores paralelo/serie y serie/paralelo son los dispositivos que hacen posible la comunicación entre equipos que trabajan con distinto formato de entrada/salida.

en el párrafo anterior, un carácter se compone de un conjunto de bits que representan "algo"; por lo tanto, el concepto carácter está directamente relacionado con el sistema de codificación, que puede considerarse como un diccionario bilingüe entre un alfabeto tradicional y otro binario.

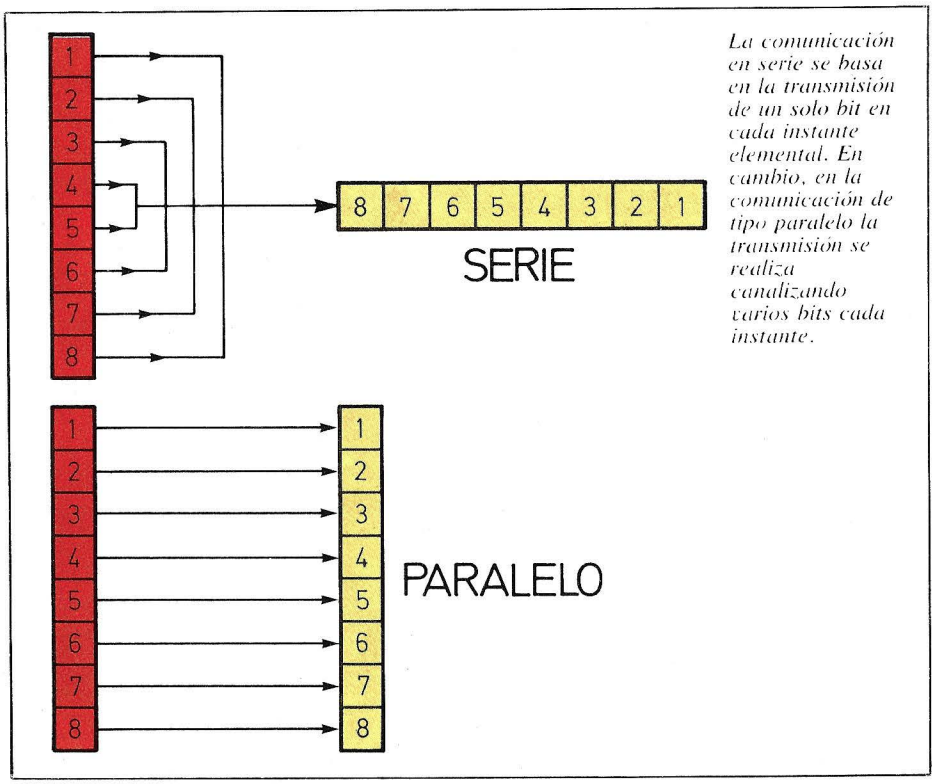
CONCEPTOS FUNDAMENTALES SOBRE COMUNICACIONES

1. *Paridad*

Como elemento físico para el intercambio de información entre equipos informáticos, se utilizan cables especiales, líneas telefónicas, etc. Debido a errores acaecidos en "el camino", es posible que el mensaje recibido no sea exactamente el emitido. Esta situación implica el cambio de un 0 por un 1, o viceversa; y, por supuesto, la alteración de un simple bit puede ser lo suficientemente trascendente como para dar al traste con todo el proceso de comunicación. Más aún, en algunos casos, cuando la comunicación se utiliza en proyectos vitales (comunicación de ordenadores con satélites, comunicación de ordenadores con equipos médicos, etc.) la presencia de un error puede resultar trágica.

Dado que la posibilidad de fallo físico es inevitable, la solución de este problema debe ser lógica. Es decir, mediante un proceso programable (incluso microprogramable) a ser posible detectar al menos la existencia de errores, y en el mejor de los casos, además de detectarlos, corregirlos. La técnica más utilizada para la detección y corrección de errores consiste en la denominada control de paridad.

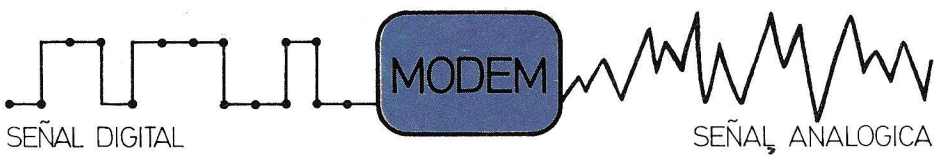
La estrategia del control de paridad consiste en introducir bits redundantes. Por ejemplo, si el sistema de codificación utiliza siete bits por carácter, en la transmisión de incluirá un bit adicional, cuyo valor (cero o uno) se determinará de forma que la suma del número de bits que valen uno sean siempre par. De esta forma, en el equipo receptor se puede garantizar que no se ha producido error en la transmisión. Sólo cuando el número de bits falseados sea mayor que 1, el sistema podrá fallar, y ello es muy improbable. En cambio, si la suma no resulta par, se tendrá la certeza de la existencia de un error.



La comunicación en serie se basa en la transmisión de un solo bit en cada instante elemental. En cambio, en la comunicación de tipo paralelo la transmisión se realiza canalizando varios bits cada instante.

2. *Tipos de transmisión*

Existen dos métodos fundamentales para establecer comunicación: transmisión en serie y transmisión en paralelo.



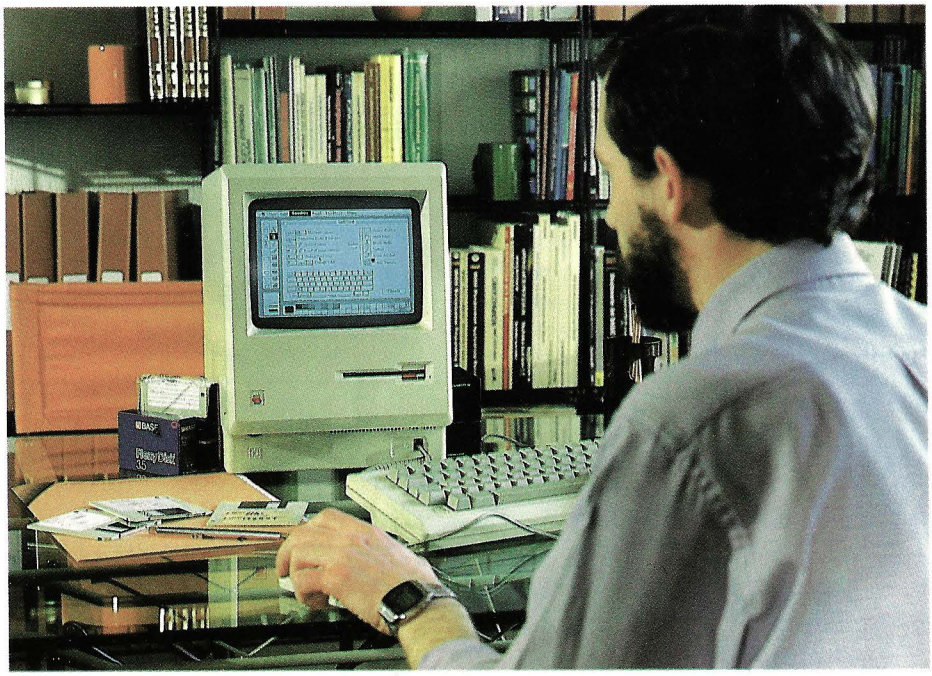
La misión de un modulador-demodulador (modem) consiste en transformar señales analógicas en digitales y viceversa.

Con la colaboración de sendos modems es posible la comunicación entre ordenadores distantes a través de las líneas telefónicas convencionales.

Cuando el proceso sea muy delicado y exija garantizar completamente la no existencia de un error, el método más frecuente consiste en incluir un mayor número de bits redundantes que aseguren la detección e incluso la corrección de los dígitos modificados. En algunos casos, la redundancia llega a ser tan grande que el número de bits adicionales puede superar al número de bits originales.

- TRANSMISION EN SERIE

En este caso, el elemento básico de comunicación es el bit y, por lo tanto, la información fluye por la línea de comunicación un bit tras otro. Ya habíamos apuntado inicialmente que el elemento codificado está formado por un número deter-



La simbiosis de informática y telecomunicaciones da lugar al concepto de "Telemática". Una realidad que rompe las barreras de la distancia por medio de la comunicación entre ordenadores. Entre las consecuencias de la aplicación de estas técnicas se encuentra el hecho de permitir "el trabajo en casa".

minado de bits; por lo tanto, habrá que esperar "n instantes" para recibir un carácter formado por n bits.

- TRANSMISION EN PARALELO

El segundo método de transmisión, denominado en paralelo, se basa en el envío simultáneo de un determinado número de bits; generalmente, el mismo número que se utiliza para formar un carácter. De esta forma, en cada instante fluirá un carácter completo a través de las líneas de comunicación.

- CONVERSIONES ENTRE METODOS DE COMUNICACION

Cualquiera de los dos sistemas descritos anteriormente tiene sus ventajas e inconvenientes. La transmisión en modo serie resulta más lenta que en modo paralelo; en cambio, esta última es más costosa que la primera. En consecuencia, algunos equipos trabajan en serie, mientras que otros lo hacen en paralelo. Si se necesita comunicar dos equipos con distinta modalidad de trabajo, resulta imprescindible utilizar un conversor de formato serie a paralelo y viceversa.

3. *Tipos de sincronización*

El concepto sincronización, dentro de la terminología informática para comunicaciones, hace referencia a la coordinación en la transmisión. Podemos distinguir dos tipos de funcionamiento, según dicha

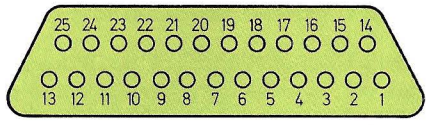
RS/232 C

Los distintos componentes de un ordenador deben comunicarse entre sí; incluso, a veces, la comunicación debe establecerse entre componentes de distintos fabricantes. Con objeto de evitar la "torre de Babel" que resultaría si cada fabricante utilizase distintos métodos de conexión, existen ciertas normas que garantizan una fácil comunicación.
Entre ellas, una de las más utilizadas es la norma o estándar oficialmente denominada: "The Electronic Industries Association Standar RS/232 C"; más comúnmente llamado RS/232 C. Anterior a este estándar existían las variantes RS/232 A y RS/232 B. Las principales mejoras aportadas por el RS/232 C son las siguientes:

1. Nivel Mecánico.
Garantiza la posibilidad de conexión entre cables con un determinado número de clavijas.

2. Nivel Eléctrico.
El nivel de señal, el nivel de voltaje y otras consideraciones eléctricas se determinan de forma que sean compatibles entre todos los equipos.

3. Nivel Lógico.
La información que formará el elemento básico de la comunicación, tiene una disposición lógica idéntica en todos los casos.
De forma muy resumida, podemos afirmar que el RS/232 C permite realizar dos operaciones básicas, unas originadas por un dispositivo informático, como pude ser un terminal y otras por un modem. Las características de estos dos tipos de operaciones son las siguientes:



Estándar RS/232C (EIA)

Patilla	Cometido
1	(AA) Masa de protección
2	(BA) Emisión de datos
3	(BB) Recepción de datos
4	(CA) Petición para enviar
5	(CB) Borrado para enviar
6	(CC) Datos correctos
7	(AB) Masa de señal
8	(CF) Detector de señal en línea de recepción
12	(SCF) Detector de señal en línea de recepción, secundario
15	(DB) Reloj de emisión
17	(DD) Reloj de recepción
19	(SCA) Petición para enviar datos, secundario
20	(CD) Terminal de datos preparado

Observaciones:
- Conector del dispositivo receptor (terminal): macho
- Conector del dispositivo emisor: hembra

- Líneas de dispositivo informático

— Transmisión de datos producidos por el dispositivo (terminal).
— Recepción de datos producidos por el modem (ordenador).
— Comprobación de que la línea de comunicación está disponible y mensajes de aviso.

- Líneas de modem

— Transmisión de datos producidos por el modem.
— Recepción de datos producidos por el dispositivo.
— Comprobación de que la línea de comunicación está disponible y mensajes de aviso.

coordinación: la denominada comunicación síncrona y su complementaria, es decir, comunicación asíncrona.

- COMUNICACION SINCRONA

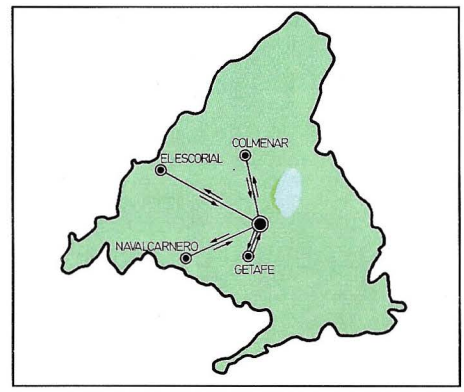
Si la comunicación se establece bajo el control de señales igualmente espaciadas y procedentes de un reloj, se dice que estamos utilizando un protocolo síncrono de comunicación.

- COMUNICACION ASINCRONA

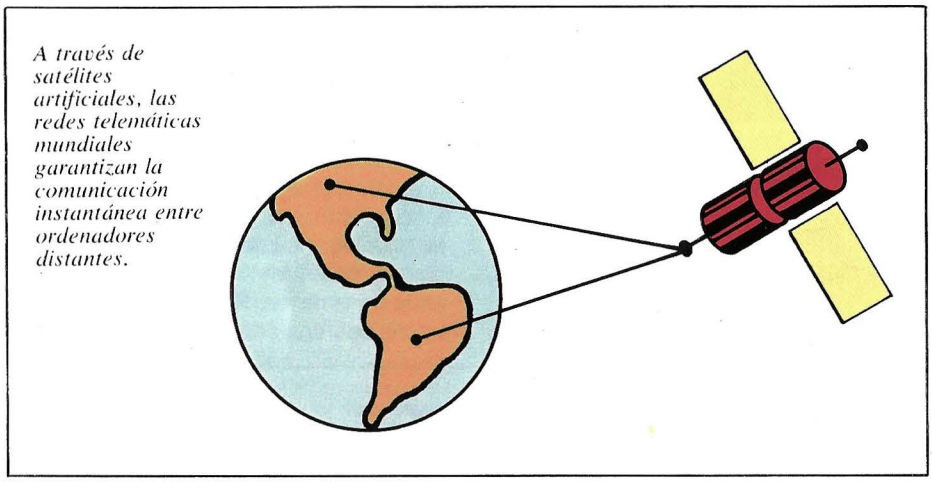
En este caso la comunicación se realiza con independencia del reloj; es decir, el fin de un proceso de transmisión marca el comienzo de un nuevo envío de información. Por lo tanto, podemos afirmar que en este caso el protocolo de comunicación es asíncrono.

4. *Modems*

Para poder aprovechar la infraestructura de líneas telefónicas ya existente, el canal más utilizado para la transmisión de información entre ordenadores es precisamente la línea telefónica. El único problema surge como consecuencia de la naturaleza analógica de las líneas telefónicas. Estas utilizan una gama continua de frecuencias y la amplitud de la señal varía muy rápidamente.
Precisamente, a la velocidad de oscilación es a lo que se llama frecuencia; ésta puede medirse en ciclos por segundo. Un oído humano puede percibir sonidos con frecuencias que varían entre 30 y 20.000 ciclos por segundo. Sin embargo, los circuitos telefónicos no transmiten la gama completa, puesto que es necesario para una conversación normal, sino que suelen transmitir entre 300 y 3.400 ciclos por segundo. En definitiva, tenemos un problema: ¿Cómo transmitir información digital por canales analógicos? La contestación es el nombre de uno de los equipos más importantes dentro del mundo de las comunicaciones en Informática: el modem. Su auténtica denominación "moduladordemodulador" se ha popularizado bajo el apelativo modem. En el fondo, estos equipos se basan en una filosofía muy sencilla: transforman la señal digital utilizada por los ordenadores en señal analógica para poder realizar transmisiones y, en sentido inverso, transforman la señal analógica recibida en señal digital procesable por el ordenador.



Las redes de comunicación formadas por ordenadores situados a distancias relativamente cortas, resultan productivas a la vez que baratas.



A través de satélites artificiales, las redes telemáticas mundiales garantizan la comunicación instantánea entre ordenadores distantes.

SOFTWARE DE COMUNICACION

Hasta aquí nos hemos limitado a describir características de tipo general sobre comunicación de equipos informáticos.
Como ya viene siendo habitual, es esta sección nos ocupamos del estudio de software de aplicación, es decir de programas preparados para solucionar problemas específicos al usuario de un ordenador personal. También dentro del campo de la comunicación existen una gran variedad de aplicaciones estándar. En un próximo capítulo dará comienzo el estudio de los principales paquetes dedicados al tema que nos ocupa.
La mayoría de los programas de comunicación comercializados tienen como misión fundamental establecer un canal entre dos ordenadores distintos; no obstante, también existen otros cuyo objetivo se limita a interconectar a un ordenador con algún dispositivo periférico complejo.
Dentro del primer grupo cabe establecer una clasificación según la categoría de los ordenadores que interactuarán. Así, podemos hablar de comunicación entre ordenadores personales, de ordenador personal a ordenador grande, o entre dos grandes ordenadores. Cuando el número de ordenadores conectados es superior a dos, podemos ya hablar de redes telemáticas. La complejidad de dichas redes puede ser muy diversa. Como ejemplo de red sencilla cabría citar a la formada por ordenadores personales ubicados en las sucursales de una entidad bancaria, en una única capital; y como ejemplo de red compleja se puede hablar de un sistema de comunicación entre grandes ordenadores ubicados en cualquier parte del mundo. En este último caso puede resultar útil el empleo de satélites de comunicación que contribuirán a minimizar los tiempos transcurridos entre los instantes de emisión y de recepción.
En todos los casos citados resulta imprescindible utilizar software especializado y técnicas de comunicación. Precisamente de esta doble faceta nació la denominación *telemática;* construida con el prefijo TELE-comunicación y el sufijo inforMATICA. Dado el carácter general de esta Enciclopedia, a partir de este momento siempre que hagamos referencia al término software de comunicación, entenderemos que el problema a resolver es el establecimiento de comunicación entre equipos integrados en el ámbito del ordenador personal.

Archivos en BASIC (1)

Introducción a los archivos secuenciales

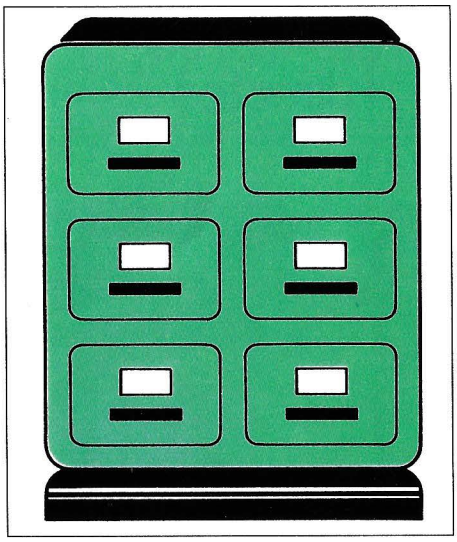
El principal inconveniente que tiene un ordenador es que sólo recuerda la información durante el tiempo en que está encendido. Esto significa que al apagar el equipo se pierde la información contenida en su memoria central. Para evitar, el engorro de volver a introducir los datos cada vez que se inicia una nueva sesión, hay que recurrir a la solución de almacenar la información en un soporte externo. En este punto, cabe recordar que los soportes más utilizados con los microordenadores son las cintas de casete y los discos flexibles.

ARCHIVANDO INFORMACION

La información almacenada en un dispositivo externo se agrupa en "archivos" o "ficheros". Un archivo consiste en un bloque homogéneo de datos o instrucciones. Así, por ejemplo, cuando se almacena un programa en cinta o disco se está creando un archivo. Sin embargo, éste no es el único método para crear bloques organizados de información.

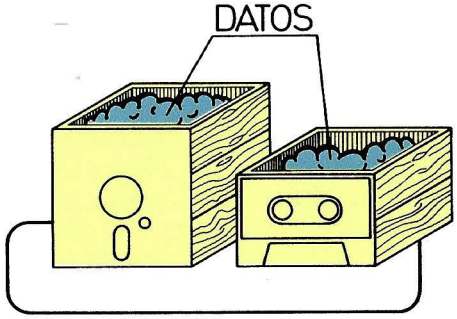
La estructura de un archivo va a depender de la forma en la que éste se almacena. En principio, un archivo se puede concebir de dos formas diferentes: secuencial o aleatorio.

Un archivo secuencial es, como su propio nombre indica, aquel al que se accede recorriendo sus elementos uno a uno (secuencialmente). Este es, precisamente, el método utilizado con las cintas en casete. En un casete es preciso pasar por los datos anteriores para acceder a un dato específico. Además, una vez leído el dato no es posible volver a él de inmediato (a no ser que se rebobine la cinta magnética manualmente). Tal es la filosofía de los archivos secuenciales.



Un lenguaje informático de propósito general, como es el BASIC, no podía relegar al olvido el apartado de tratamiento de archivos.

Un archivo de tipo aleatorio o de acceso directo funciona de forma bien distinta. En él se puede acceder a un dato directamente, sin necesidad de leer los anteriores. Concretamente, la información almacenada en un archivo se agrupa en registros y dichos registros son los que pueden ser recuperados de forma directa; esto es: se puede leer el registro quinto, luego el segundo y más tarde el octavo. Por su naturaleza, este tipo de archivos no puede ser creado en casete, sino tan sólo en soportes de acceso aleatorio como es el caso del disco.



Para el almacenamiento permanente de la información se utilizan soportes de memoria conectados externamente al ordenador. En el ámbito de la microinformática, los soportes más frecuentes son la cinta en casete y el disco magnético.

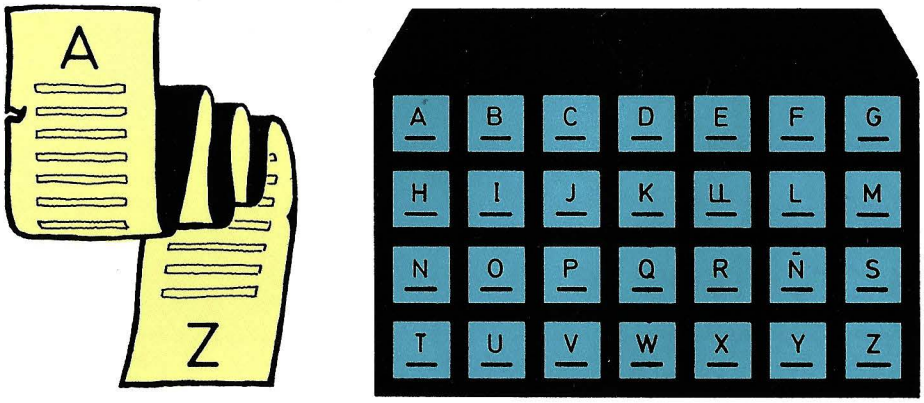
MANEJO DE ARCHIVOS

Para manejar los archivos es necesario seguir, paso a paso, una secuencia definida de acciones. Los pasos a dar se asemejan en cierta medida a los necesarios con un archivo físico. Esto es: apertura del archivo, consulta y cierre del mismo.

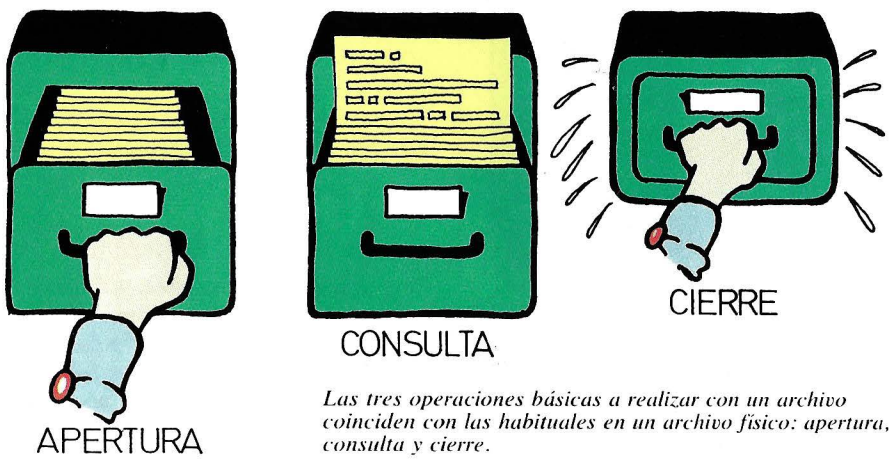
Para la apertura y cierre se hace uso de dos comandos BASIC genéricos: OPEN (abrir) y CLOSE (cerrar). El manejo de la información, sin embargo, depende del tipo de archivo.

Los archivos de tipo secuencial funcionan de modo muy semejante al teclado y a la pantalla. Para escribir en uno de ellos se mandan los datos uno tras otro, de la misma forma que los datos se mandan a la pantalla con un PRINT. La lectura se efectúa también de forma análoga, equivalente a la recogida de datos del teclado con la orden INPUT.

Si el archivo a tratar es aleatorio, habrá que tener presente el formato de los registros. Una vez definido el archivo en cuestión, la lectura y escritura se realiza trasladando a la memoria central del ordenador el registro adecuado. Con el registro en la memoria del ordenador, la lectura o escritura de sus elementos es similar a la lectura o escritura del contenido de una variable.



Atendiendo al método de acceso a la información almacenada, cabe diferenciar entre dos tipos básicos de archivos: los secuenciales y los de acceso directo o "aleatorio".



Las tres operaciones básicas a realizar con un archivo coinciden con las habituales en un archivo físico: apertura, consulta y cierre.

ARCHIVOS SECUENCIALES: APERTURA

Como se indicó anteriormente, el comando que realiza la apertura de un fichero es OPEN. Antes de proceder a la explicación de OPEN es necesario conocer algunos detalles acerca de la apertura de archivos.

Los archivos almacenados en un mismo soporte se identifican por medio de un nombre. Por ejemplo, al grabar un programa es preciso otorgar a éste un nombre que facilite su identificación. Este nombre sirve para no confundir unos archivos con otros, dentro del mismo soporte. Así pues, cada archivo ha de tener su nombre identificativo.

Como sabemos, existen dos tipos de archivos: secuenciales y aleatorios. Ambos tipos son mutuamente incompatibles, por ello es preciso especificar el tipo al acceder a un archivo. En el caso de los archivos secuenciales no es posible leer y escribir al tiempo. Ello hace que se deba optar forzosamente por una de ambas acciones.

En algunas ocasiones puede ser necesario tener abierto más de un archivo al tiempo; por ejemplo, a la hora de traspasar información de un archivo a otro, o comparar dos archivos entre sí. Si esto ocurre, será preciso tener bien diferenciados los distintos archivos abiertos.

Toda esta información debe ser indicada en la apertura del archivo. Así pues, la instrucción OPEN presenta, por lo general, el siguiente formato:

(número de línea) OPEN <modo>,# <número>, <nombre>

A continuación de la palabra reservada OPEN se indica, en la zona <modo>, el tipo de archivo.

En el caso de los archivos secuenciales se especificará también si van a utilizarse como archivos de salida o de entrada, escribiendo una "O" (de Output=salida) o una "I" (de Input=entrada).

Tras el signo de número (#) se adjunta un dato o expresión numérica. Este dato identifica al archivo en cuestión entre todos los que se encuentren abiertos al mismo tiempo.

Por último, ha de figurar el nombre otorgado al fichero. Junto al nombre es corriente indicar el dispositivo en el que se encuentra el archivo.

Este formato no es ni mucho menos generalizable, aunque sí es de los más utilizados.

Existe otro formato, también muy frecuente, el cual ofrece el siguiente aspecto:

(número de línea) OPEN <nombre> FOR <modo> AS # <número>

En el que <modo> admite las palabras clave INPUT o OUTPUT.

A lo largo de este capítulo se utilizará el primero de los formatos descritos. Como ya se ha mencionado, la estandarización en este tema brilla por su ausencia. De todos modos este capítulo sólo pretende aportar una introducción; más adelante se completará la información relativa al tema de manejo de archivos.

Con las indicaciones señaladas, la apertura de un archivo no reviste mayor complicación. Por ejemplo, para abrir un archivo de nombre DATOS, almacenado en casete, para realizar en él operaciones de lectura, bastará con introducir lo siguiente:

```
OPEN "I",#1,"C:DATOS"
```

La instrucción especifica que se realizarán operaciones de lectura sobre el archivo (entrada al ordenador) con el dato "I", y que se trata del archivo DATOS residente en casete, con el indicativo "C:DATOS". El periférico suele especificarse con algunos caracteres que preceden al nombre del archivo y separados de éste por el signo dos puntos (:). En nuestro caso hemos tomado "C:" como indicativo de la unidad de casete. Algunos equipos consideran al casete como dispositivo por defecto; esto es, si no se especifica otra cosa se supone que el archivo se encuentra en casete.

OPEN es el comando BASIC especializado en la apertura de archivos.

Si se desea abrir un nuevo archivo para salida de datos puede utilizarse la siguiente expresión:

```
OPEN "O",#2,"C:RESULT"
```

En este segundo caso se ha utilizado la opción "O" que permite la salida de datos. El número 2 diferencia a este nuevo archivo del abierto con anterioridad. Por lo demás, se especifica que el archivo se encuentra en disco por medio del prefijo **D:**. Tras abrir el archivo puede ya procederse a su lectura o actualización.

ESCRITURA EN UN ARCHIVO SECUENCIAL

Para escribir datos en un archivo secuencial es imprescindible que éste haya sido abierto con anterioridad. En el comando de apertura se habrá otorgado al archivo un número identificador que se utilizará, posteriormente, como referencia para el acceso al archivo en cuestión. Además, el archivo se habrá abierto en modo escritura (con el carácter "O").

Como ya se mencionó anteriormente, la escritura en un archivo secuencial se asemeja a la escritura en pantalla; hasta el punto de que incluso el comando al efecto coincide: PRINT. Al actuar sobre archivos, el comando PRINT se acompaña del signo # seguido por el número del archivo a tratar.

Volviendo al ejemplo anterior, para escribir la palabra "SALIDA" en el archivo RESULT habría que teclear lo siguiente:

```
PRINT#2,"SALIDA"
```

En todo caso, lo más interesante será guardar una cantidad grande de datos, por ejemplo un "array" o matriz de datos. Para almacenar una matriz en un dispositivo externo es aconsejable utilizar un bucle FOR. El siguiente ejemplo muestra cómo realizar dicha operación.

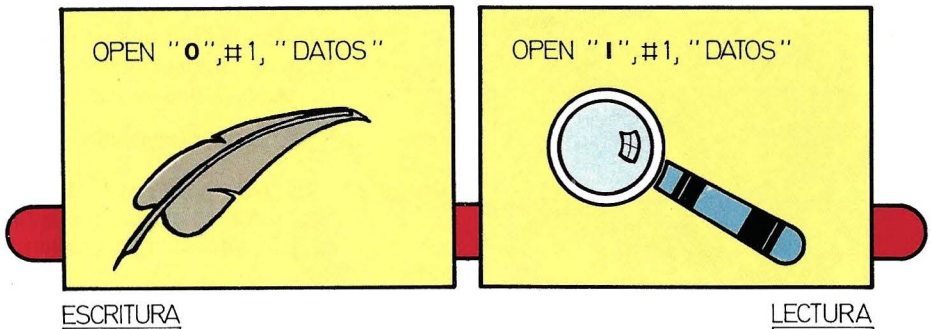
```
100 OPEN "O",# 1,"C:MATRIZA"
110 FOR I=1 TO 100
120 PRINT#1,A(I)
130 NEXT I
```

Estas cuatro líneas de programa almacenarán automáticamente los cien elementos de la matriz A en el archivo en casete llamado MATRIZA, datos que quedarán almacenados uno tras de otro. A la hora de proceder a su lectura éstos aparecerán en el mismo orden en el que se almacenaron, y en cualquier caso el primero en leerse será el elemento A(1).

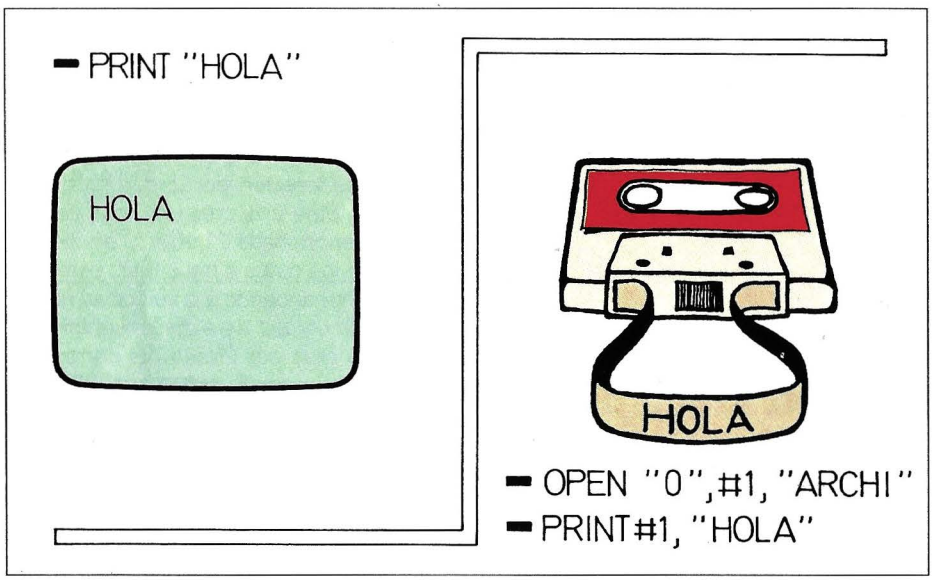
Si se desean almacenar distintos tipos de datos en un mismo archivo será preciso utilizar varios comandos PRINT#. Un ejemplo de esta posibilidad es el que figura a continuación.

```
200 OPEN "O",#1, "LISTA"
210 PRINT#1,"LISTA DE DATOS"
220 PRINT#1,"NUMERO"
230 PRINT#1,N
240 PRINT#1,"FECHA"
250 PRINT#1,F$
260 PRINT#1,"CANTIDAD"
270 PRINT#1,TOTAL+1200
```

Los datos enviados a un archivo pueden formatearse utilizando la opción USING, al igual que en el caso de la presentación en pantalla. En este último ejemplo se podría limitar el espacio ocupado en el archivo de la siguiente forma.



A la hora de abrir un archivo secuencial, es preciso indicar el modo en el que éste se va a utilizar; éste puede coincidir con el modo escritura/salida ("O") o lectura/entrada ("I").



Para escribir en un archivo secuencial se actúa de forma parecida a la escritura en pantalla. El comando utilizado al efecto es el conocido PRINT.

OPEN

Realiza la apertura de un archivo en un dispositivo de almacenamiento externo.

Formato: OPEN "<modo>",[#]<número>,<nombre>

Ejemplos:

```
10 OPEN "I",#1,"DATOS"
50 OPEN "O",#2,"C:ARCHIVO"
```

CLOSE

Cierra el archivo cuyo número figura en su argumento.

Formato: CLOSE [#] <número>

Ejemplos:

```
20 CLOSE#1
70 CLOSE 2
```

PRINT#

Escribe un dato en el archivo especificado.

Formato: PRINT#<número>,<dato>[;<dato>...]

Ejemplos:

```
20 PRINT#1,"PERICO"
70 PRINT#2, A+B
```

INPUT#

Lee el siguiente dato almacenado en el archivo que se indica en su argumento.

Formato: INPUT#<número>,<var>[;<var>...]

Ejemplos:

```
20 INPUT#1,A$
70 INPUT#3,DATO
```

```
200 OPEN "O",#1, "LISTA"
210 PRINT#1,"LISTA DE DATOS"
220 PRINT#1,"NUMERO"
230 PRINT#1,USING "###";N
240 PRINT#1,"FECHA"
250 PRINT#1,USING "    ";F$
260 PRINT#1,"CANTIDAD"
270 PRINT#1,USING "######.#";
    TOTAL+1200
```

CIERRE DEL ARCHIVO

Una vez realizadas las operaciones pertinentes, es conveniente cerrar de nuevo el archivo. El cierre de un archivo deja libre el número de archivo ocupado por éste. En consecuencia, se pueden abrir dos archivos con el mismo número, uno después de cerrar el otro. No obstante, el principal cometido del cierre de un archivo secuencial es marcar el final del mismo.

Cuando se graba un archivo en casete se añaden unos identificadores a la información propia del archivo. Entre estos indicadores está la cabecera, situada al principio, que guarda el nombre del archivo. La cabecera se genera a partir de la ejecución del comando OPEN.

Otro indicador es la marca de fin de archivo. Esta marca indica que el archivo no contiene más información. Con ella se evita el seguir leyendo más allá de los límites de un archivo.

El comando de cierre de un archivo es CLOSE. Para ordenar el cierre sólo es necesario conocer el número asociado al archivo. Como ya se sabe, en la apertura se aportan todos los datos identificativos del archivo y se otorga a éste un número. Es precisamente ese número el que definirá el archivo a cerrar. La formulación de este comando es, pues, la que se indica.

(número de línea) CLOSE#<número>

De esta forma, para cerrar cualquiera de los archivos creados en el apartado anterior bastaría con introducir la siguiente orden:

CLOSE#1

El tratamiento de archivos por parte del ordenador no es tan sencillo como aparece a ojos del usuario. El ordenador ha de mandar la información a la velocidad adecuada para su grabación en la unidad de almacenamiento. Esta velocidad es generalmente inferior a la velocidad de operación del ordenador, por lo que ha de ser adaptada. Para ello, crea una zona de memoria, denominada "buffer", en la que almacena los datos a transmitir; estos datos se van mandando a la velocidad requerida por la unidad de almacenamiento. El buffer se crea por efecto del correspondiente comando de apertura OPEN. En realidad, el número asignado a cada archivo abierto no es más que el identificador del buffer asociado al mismo.

Cuando se ejecuta un comando CLOSE, al tiempo de cerrar el archivo correspondiente, se libera la zona de memoria reservada a su buffer. Por este motivo es importante cerrar los archivos una vez que

han terminado las operaciones con los mismos.

LECTURA DEL ARCHIVO

Hemos visto que la escritura en un archivo secuencial es análoga a la escritura en pantalla. Consecuentemente, la lectura se asemeja en gran medida a la recogida de información a partir del teclado.
La lectura de datos almacenados en un archivo secuencial se realiza por medio del comando INPUT#. Su formulación es la misma que la del INPUT normal, con la adición del número de archivo a leer.

(número de línea) INPUT#<número>, <var.1> [,<var.2>...]

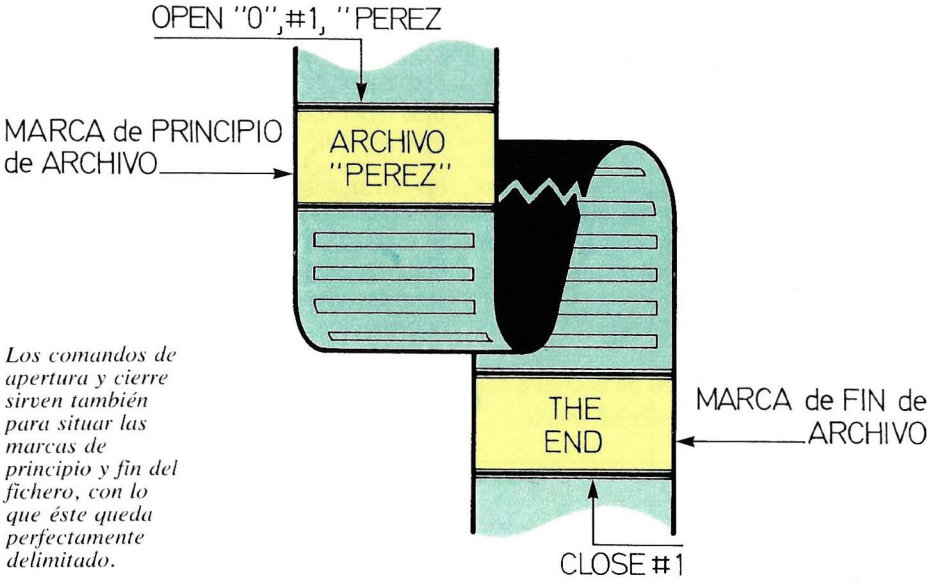
Como es natural, el archivo ha de estar abierto previamente y con la opción de lectura ("I"). Dadas las características de las cintas de casete, es imposible abrir en ellas más de un archivo al tiempo (la cabeza no puede posicionarse en dos lugares a la vez). Tampoco es posible leer y escribir en un mismo archivo. Estas posibilidades quedan reservadas para cuando se dispone de varias unidades de casete o de una unidad de disco.
Para la lectura de un archivo en casete hay que situar la cinta en el punto adecuado (en cualquier posición anterior al comienzo del archivo). Hecho esto, se debe abrir el archivo y proceder a su lectura. A título de ejemplo, veamos los comandos capaces de leer la matriz almacenada en el apartado anterior.

```
1000 OPEN "I",#1,"C:MATRIZA"
1010 FOR I=1 TO 100
1020 INPUT#1,B(I)
1030 NEXT I
1040 CLOSE#1
```

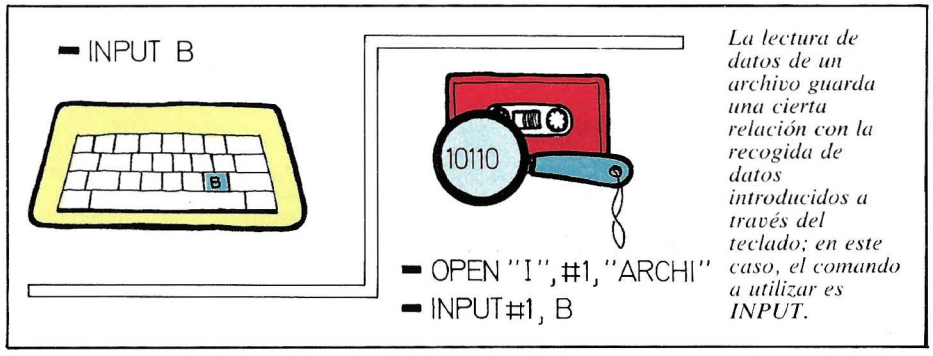
Cuando se trata de archivos, el comando INPUT no permite la opción de presentación de un mensaje. Así pues, la orden INPUT#1 "DATO", D produciría un error de tipo sintáctico.
Como se ha visto en el anterior ejemplo, el empleo de INPUT# es tan sencillo como el de PRINT#, el único cuidado está en no olvidar la apertura y cierre del archivo. Para completar la serie de ejemplos veamos cómo se realizaría la lectura del otro archivo creado en un apartado precedente:

```
2000 OPEN "I",#1,"LISTA"
2010 INPUT#1,CABE$
2020 INPUT#1,A$
2030 INPUT#1,NUM
2040 INPUT#1,B$
2050 INPUT#1,DIA$
2060 INPUT#1,C$
2070 INPUT#1,TOTAL
```

Como se observa, no es preciso que los nombres de las variables coincidan con los que se utilizaron al crear el archivo. Sin embargo, los tipos de las variables sí deben estar en correspondencia.



Los comandos de apertura y cierre sirven también para situar las marcas de principio y fin del fichero, con lo que éste queda perfectamente delimitado.



La lectura de datos de un archivo guarda una cierta relación con la recogida de datos introducidos a través del teclado; en este caso, el comando a utilizar es INPUT.

ARCHIVOS COMO BASES DE DATOS

En el capítulo anterior se describió un ejemplo consistente en el diseño de una base de datos. El programa así creado puede resultar de gran utilidad, sin embargo es preciso introducir de nuevo los datos cada vez que se desee utilizar el programa. Para solventar esta situación puede hacerse uso de los archivos recién presentados.
En aquel ejemplo se utilizaba una matriz de 100×10 para almacenar los datos. El paso de esta matriz desde la memoria central al casete es muy sencillo. Al tratarse de una matriz de dos dimensiones, son necesarios dos bucles anidados. Una primera aproximación a la rutina de almacenamiento la constituyen las siguientes líneas.

```
4110 OPEN "O",#1,"BASE"
4120 FOR REG=1 TO 100
4130 FOR CAM=1 TO 10
4140 PRINT#1,A(REG,CAM)
4150 NEXT CAM
4160 NEXT REG
4170 CLOSE#1
```

Los dos bucles harán que la matriz se grabe registro a registro. Dentro de cada registro se almacenan los campos por su orden. Aunque también es posible grabar la matriz en otro orden: por campos dentro de los cuales los registros se mantengan ordenados. Para utilizar esta otra distribución basta con cambiar de sitio los bucles.

```
4110 OPEN "O",#1,"BASE"
4120 FOR CAM=1 TO 10
4130 FOR REG=1 TO 100
4140 PRINT#1,A(REG,CAM)
4150 NEXT REG
4160 NEXT CAM
4170 CLOSE#1
```

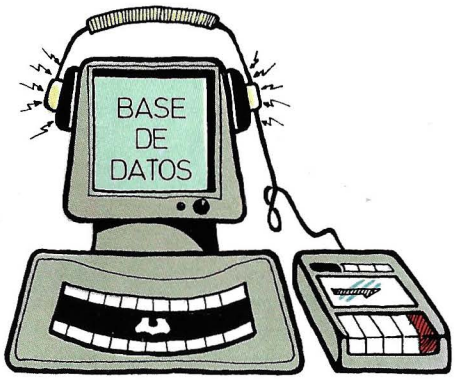
El orden en que se almacenen los datos no importa, siempre que se emplee ese mismo orden a la hora de proceder a su lectura. En todo caso, y dado que es el más intuitivo, vamos a poner en práctica el primer método.

En las líneas escritas hasta ahora se realiza correctamente el almacenamiento de datos. No obstante, el archivo en el que se guardan es siempre el mismo: "BASE". Lo más lógico sería poder diferenciar distintos conjuntos de datos dando diferentes nombres a los archivos. Esto es perfectamente posible ya que el comando OPEN admite variables en su argumento. De esta forma se puede elegir el nombre del archivo con entera libertad.

```
4010 INPUT "NOMBRE DEL ARCHIVO";N$
4110 OPEN "O",#1,N$
.....
.....
```

En estas líneas se recoge el nombre y se abre el archivo cuyo nombre coincide con el indicado. La longitud del nombre vendrá limitada por el equipo en cuestión, y más concretamente por el sistema operativo. La longitud de los nombres de archivo suele situarse alrededor de los ocho caracteres. Cuando se trabaja con unidad de disco depende del sistema operativo empleado, los habituales CP/M y MS-DOS utilizan ocho caracteres más tres de "extensión".

Con la posibilidad de elegir el nombre del archivo se abren otras ventajas interesantes. Por ejemplo, si usted guarda los archivos de la base de datos en una cinta donde residan más tipos de archivos, querrá poder distinguirlos. Para diferenciar los archivos creados por la base de datos



En una base de datos —como es el caso del ejemplo propuesto en el capítulo anterior—, resulta fundamental la posibilidad de emplear archivos residentes en una unidad de almacenamiento externo.

puede concluir su nombre con algunas letras características, por ejemplo BD (Base de Datos). Ello se consigue con el tratamiento del dato de cadena introducido en la línea 4010.

```
4010 INPUT "NOMBRE DEL ARCHIVO";N$
4020 LET N$=N$+"/BD"
4110 OPEN "O",#1,N$
.....
.....
```

En el proceso de la línea 4020 (adición de los caracteres "/", "B" y "D") puede rebasarse la longitud máxima especificada para el nombre. Si el límite está en ocho caracteres, habrá que elegir cinco de los aportados por el usuario, que sumados a los tres añadidos, hacen en total ocho. Esta sería la nueva línea:

```
4020 LET N$=LEFT$(N$,5)+"/BD"
```

Con lo expuesto, la rutina de escritura del archivo presentará un nuevo aspecto:

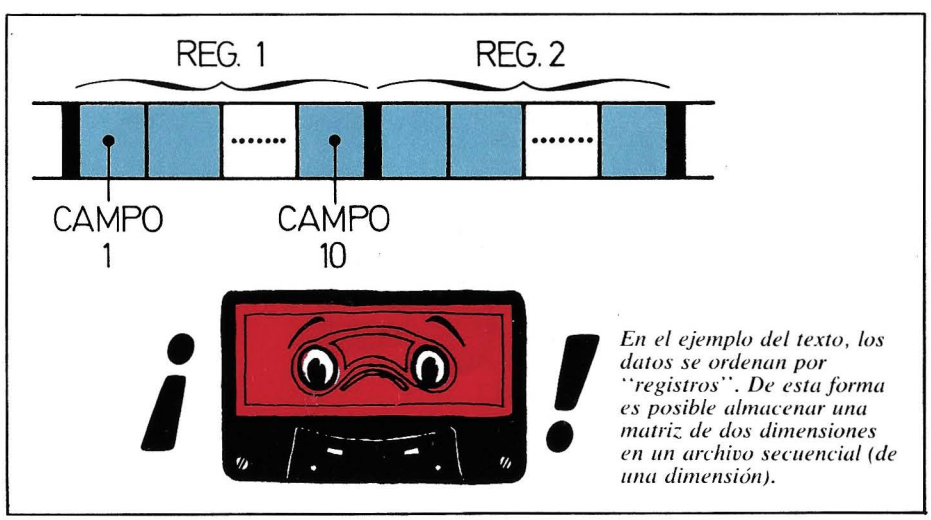
```
4000 REM      ------------
4001 REM      --- GRABACION ---
4002 REM      ------------
4010 INPUT "NOMBRE DEL ARCHIVO";N$
4020 LET N$=LEFT$(N$,5)+"/BD"
4110 OPEN "O",#1,N$
4120 FOR REG=1 TO 100
4130 FOR CAM=1 TO 10
4140 PRINT#1,A(REG,CAM)
4150 NEXT CAM
4160 NEXT REG
4170 CLOSE#1
4180 RETURN
```

LECTURA DE LOS ARCHIVOS DE DATOS

La contrapartida a la grabación es la lectura de los archivos. La zona de lectura de los datos es idéntica a la de la escritura, sin más que cambiar las órdenes PRINT por INPUT. Esto es:

```
5120 FOR REG=1 TO 100
5130 FOR CAM=1 TO 10
5140 INPUT#1,A(REG,CAM)
5150 NEXT CAM
5160 NEXT REG
```

No hay que olvidar que el orden de lectura debe ser congruente con el de escritura. Si usted lo alteró en el apartado anterior debe tomar la misma medida en éste. La zona de apertura tendrá que construir el nombre de la misma forma que lo hacía la de escritura. La única diferencia reside en la presencia del carácter "I" que indica que se trata de un archivo de entrada.



En el ejemplo del texto, los datos se ordenan por "registros". De esta forma es posible almacenar una matriz de dos dimensiones en un archivo secuencial (de una dimensión).

TABLA DE CONVERSION

ORDENADOR	APERTURA		CIERRE	ESCRITURA	LECTURA	SOPORTE DE MEMORIA
	OPEN "<modo>",#<n.º>,<nom>	OPEN <nom> FOR <m>AS<n.º>	CLOSE#<n.º>	PRINT#<n.º>, <dato>	INPUT#<n.º>, <var>	
APPLE II (APPLESOFT)	OPEN <nombre> (1)	—	CLOSE <nombre>	WRITE <nombre> (2)	READ <nombre> (2)	Disco
APRICOT (M-BASIC)	OPEN "<modo>",#<n.º>,<nom>	OPEN <nom> FOR <m>AS<n.º>	CLOSE#<n.º>	PRINT#<n.º>, <dato>	INPUT#<n.º>, <var>	Disco
ATARI	OPEN#<n.º>,<modo>, ,<nom> (3)	—	CLOSE#<n.º>	PRINT#<n.º>, <dato>	INPUT#<n.º>, <var>	Casete
CBM 64	OPEN <n.º>,1,<modo> <nom>	—	CLOSE#<n.º>	PRINT#<n.º>, <dato>	INPUT#<n.º>, <var>	Casete
DRAGON	OPEN "<modo>",#−1, <nom>	—	CLOSE#−1	PRINT#−1, <dato>	INPUT#−1, <var>	Casete
EQUIPOS MSX	—	OPEN <nom> FOR <m>AS<n.º>	CLOSE#<n.º>	PRINT#<n.º>, <dato>	INPUT#<n.º>, <var>	Casete
HP-150	OPEN "<modo>",#<n.º>,<nom>	OPEN <nom> FOR <m>AS<n.º>	CLOSE#<n.º>	PRINT#<n.º>, <dato>	INPUT#<n.º>, <var>	Disco
IBM PC	OPEN "<modo>",#<n.º>,<nom>	OPEN <nom> FOR <m>AS<n.º>	CLOSE#<n.º>	PRINT#<n.º>, <dato>	INPUT#<n.º>, <var>	Disco
MPF	—	—	—	—	—	—
NCR DM-V (MS-BASIC)	OPEN "<modo>",#<n.º>,<nom>	—	CLOSE#<n.º>	PRINT#<n.º>, <dato>	INPUT#<n.º>, <var>	Disco
NEW BRAIN	OPEN <modo>,#<n.º>,1,<nom>	—	CLOSE#<n.º>	PRINT#<n.º>, <dato>	INPUT#<n.º>, <var>	Casete
ORIC	—	—	—	—	—	—
SHARP MZ-700 (MZ-BASIC)	—	—	—	—	—	—
SINCLAIR QL	OPEN#<n.º>, <nom> (6)	—	CLOSE#<n.º>	PRINT#<n.º>, <dato>	INPUT#<n.º>, <var>	Microdrive
SPECTRAVIDEO	—	OPEN <nom> FOR <m>AS<n.º>	CLOSE#<n.º>	PRINT#<n.º>, <dato>	INPUT#<n.º>, <var>	Casete
ZX-SPECTRUM	—	—	—	SAVE <nom> DATA <Array> (7)	LOAD <nom> DATA <Array> (7)	Casete

Las formulaciones reflejadas en la tabla para cada ordenador, corresponden al caso del soporte de memoria que se especifica en la columna correspondiente.

(1) En Apple, los comandos de manejo de archivos se introducen como cadenas de caracteres, precedidas de un CTRL-D, dentro de instrucciones PRINT. Por ejemplo:

```
10 D$=" ":REM CTRL-D
20 PRINT D$;"OPEN DATOS"
```

(2) Los comandos WRITE y READ sirven para que los siguientes PRINT e INPUT no actúen con la pantalla y el teclado, sino con el archivo indicado.

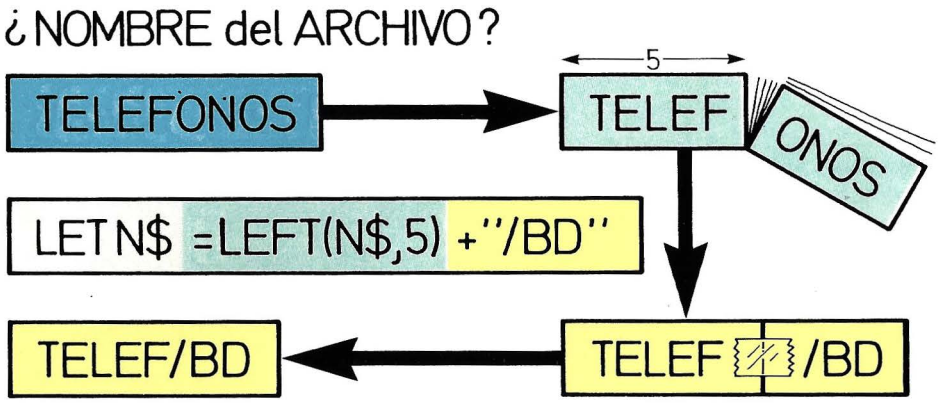
(3) En la zona <modo> se utilizan los números 4 para entrada y 8 para salida de datos.

(4) El modo se especifica como 0 (lectura) ó 1 (escritura).

(5) En <modo> se emplea IN en lugar de "I" y OUT en lugar de "O". El número 1 indica el primer conector de casete; se puede utilizar el segundo poniendo un 2.

(6) El nombre de archivo en microdrive debe tener la forma MDV<n.º>−<nombre>, siendo <n.º> el número de la unidad de microdrive. Por ejemplo, MDV1−ARCHI.

(7) Se utilizan para la grabación y lectura de ARRAYS en casete.



El nombre del archivo es modificado, haciendo que sus tres últimos caracteres sean /BD. Con ello, se ve facilitada la identificación de los archivos creados por medio de éste programa.

Las dos nuevas opciones introducidas han de verse contempladas en el menú. Unos ligeros retoques en la correspondiente rutina servirán a tal propósito.

```
5000 REM ----------
5001 REM ---LECTURA---
5002 REM ----------
5010 INPUT "NOMBRE DEL ARCHIVO";N$
5020 LET N$=LEFT$(N$,5)+"/BD"
5110 OPEN "I",#1,N$
5120 FOR REG=1 TO 100
5130 FOR CAM=1 TO 10
5140 INPUT#1,A(REG,CAM)
5150 NEXT CAM
5160 NEXT REG
5170 CLOSE#1
5180 RETURN
```

EL ULTIMO DETALLE

```
5010 INPUT "NOMBRE DEL ARCHIVO";N$
5020 LET N$=LEFT$(N$,5)+"/BD"
5110 OPEN "I",#1,N$
```

Por último, han de introducirse las líneas de cierre del archivo y retorno al programa principal. Estas dos líneas sí son exactamente iguales a las empleadas en la rutina anterior.

```
5170 CLOSE#1
5180 RETURN
```

Así pues, siguiendo los pasos comentados, se llega a la consecución de la rutina de lectura. Esta rutina completa y con los REM de identificación es la que sigue:

Para el correcto funcionamiento de estas rutinas con el programa analizado en el capítulo anterior, quedan por añadir unos retoques al programa original. En primer lugar, las opciones de lectura y escritura del archivo deben estar contempladas en el menú inicial. Para ello, habrá que introducir nuevas líneas que escriban sus nombres en pantalla.

```
285 LOCATE 16,5
288 PRINT "4.....GRABACION DEL ARCHIVO"
290 LOCATE 18,5
295 PRINT "5.....LECTURA DE ARCHIVO"
```

Con estas líneas se obtiene la presentación en pantalla de ambas posibilidades. Pero con esto no basta, también es necesario contemplar dichas opciones en el salto a las subrutinas; esto es: hay que cambiar la línea del ON/GOSUB. Como las nuevas rutinas se encuentran localizadas a partir de las líneas 4000 y 5000, la modificación es la siguiente:

```
330 ON N GOSUB 1000,2000,3000,4000,5000
```

Como último retoque hay que modificar ligeramente la línea 310, que contiene el comando INPUT. La línea original muestra el siguiente aspecto.

```
310 INPUT "ELJA OPCION (1-3)",N$
```

Esta línea, después de la modificación, debe indicar que existen cinco posibles opciones. Así pues, la nueva línea quedará como figura a continuación.

```
310 INPUT "ELJA OPCION (1-5)",N$
```

Una vez incluidas las modificaciones señaladas nos encontraremos ante la siguiente rutina:

```
199 REM ----------
200 REM -- MENU ---
201 REM ----------
210 LOCATE 5,10
220 PRINT "MENU"
230 LOCATE 10,5
240 PRINT "1.....ENTRADA DE DATOS"
250 LOCATE 12,5
260 PRINT "2.....EDICION DE DATOS"
270 LOCATE 14,5
280 PRINT "3.....VISUALIZACION DE DATOS"
285 LOCATE 16,5
288 PRINT "4.....GRABACION DEL ARCHIVO"
290 LOCATE 18,5
295 PRINT "5.....LECTURA DEL ARCHIVO"
300 LOCATE 20,5
310 INPUT "ELJA OPCION (1-5)",N$
320 N=VAL(N$)
330 ON N GOSUB 1000,2000,3000,4000,5000
340 GOTO 200
```

Introduciendo los cambios, que se resumen en las rutinas de lectura y escritura del archivo y la de menú, el programa del capítulo anterior resulta verdaderamente operativo. En realidad, en una base de datos es fundamental mantener los archivos en un dispositivo de almacenamiento externo.

En la apertura de los archivos se ha prescindido de la especificación de dispositivo. Tal como se ha programado se utilizará el dispositivo que el ordenador tome "por defecto"; en la mayor parte de los casos, éste corresponde a la unidad de casete. Para cambiar de dispositivo bastaría con indicarlo en la apertura de los archivos.

Forth (2)

Operadores aritméticos y manipulación de la pila

En este segundo capítulo avanzamos en la exposición de los conceptos básicos relativos al lenguaje FORTH. En primer lugar, se proseguirá con el estudio de las operaciones aritméticas y su peculiar actuación en este lenguaje, para continuar presentando las operaciones cuyo objetivo es realizar manipulaciones de la pila.

MAS OPERADORES ARITMETICOS

El lenguaje FORTH dispone de dos palabras, destinadas a realizar incrementos; éstas son las dos siguientes:

1+ Suma una unidad al elemento situado en la cima de la pila, depositando en la cima el resultado de la operación.
Ejemplo:

```
2 1+  .<CR>
2 1+  .3 OK
```

2+ Incrementa en dos unidades el elemento situado en la cima de la pila, dejando en su lugar el resultado de la operación.
Ejemplo:

```
23 2+  .<CR>
23 2+  .25 OK
```

Los dos caracteres que componen estas palabras FORTH han de escribirse seguidos, sin dejar espacio en blanco entre ellos. De no hacerlo así, el resultado de la operación sería el mismo, si bien no coincidiría el proceso ejecutado y ello puede influir en el desarrollo del programa además de hacerlo más lento.

1– Esta nueva palabra permite restar una unidad al número situado en la cima de la pila, colocando en su lugar el resultado de la operación.
Ejemplo:

```
23 1-  .<CR>
23 1-  .22 OK
```

2– Su funcionamiento es semejante al de la palabra anterior, pero en lugar de restar una sola unidad resta dos.
Ejemplo:

```
34 2-  .<CR>
34 2-  .32 OK
```

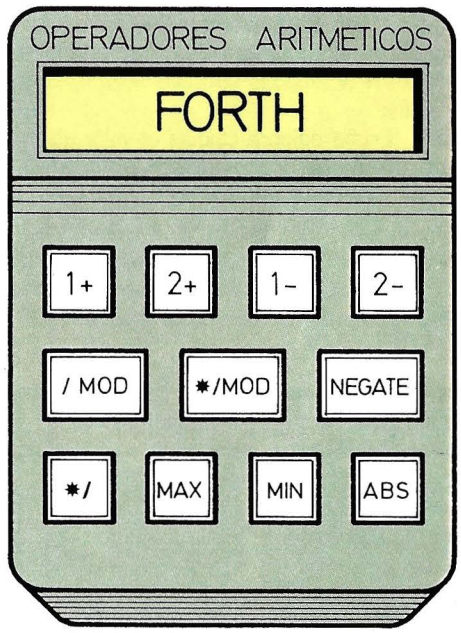
/MOD Divide el primer número introducido en la pila por el segundo, depositando en ella dos números resultantes: el de la cima, coincidente con el módulo de la división, y el siguiente que corresponde al resto de la división.
Ejemplo:

```
3 6 /MOD  . .<CR>
3 6 /MOD  .0  .3 OK
```

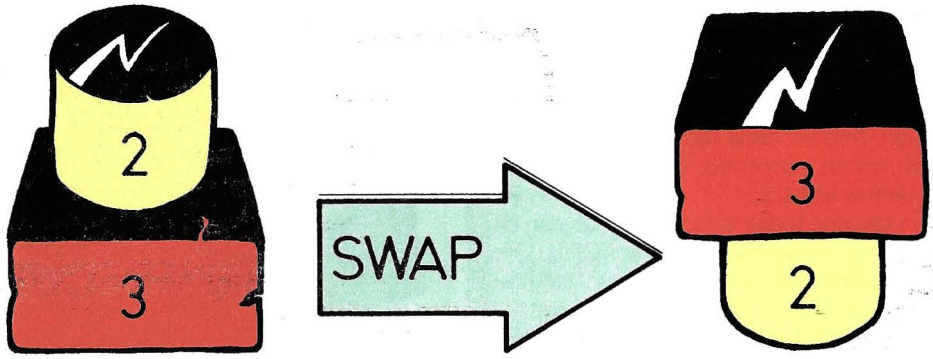
OPERADORES EVOLUCIONADOS

El lenguaje FORTH posee, además, una serie de operadores complejos que ejecutan más de una operación, empleando, para ello, dos o incluso más operandos.

*/ Esta palabra multiplica el segundo número situado en la pila por el tercero, dividiendo el resultado por el situado en el



El Forth es un lenguaje de programación con un amplio repertorio de palabras cuya finalidad es la de actuar a modo de operadores aritméticos.



SWAP es una palabra FORTH integrada en el grupo de órdenes para la manipulación de la pila. Concretamente, SWAP intercambia los dos números que ocupan las posiciones superiores de la pila.

primer lugar de la pila, esto es, en la cima.
Ejemplo:

```
2 4 5 */ . <CR>
2 4 5 */ . 1 OK
```

***/MOD** Ejecuta las dos operaciones descritas en el caso anterior, con la diferencia de que, además, proporciona el resto de la división.
Ejemplo:

```
2 4 5 */MOD . . <CR>
2 4 5 */MOD . 1 . 3 OK
```

Las siguientes palabras FORTH realizan algunas manipulaciones de datos íntimamente relacionadas con operaciones aritméticas.

ABS Dicha palabra calcula el valor absoluto del número que se encuentra en la parte superior de la pila, y coloca el resultado en esa misma posición.
Ejemplo:

```
-2 ABS . <CR>
-2 ABS . 2 OK
```

MAX Toma los dos números emplazados en las posiciones superiores de la pila, y deposita el mayor de ambos en la cima.
Ejemplo:

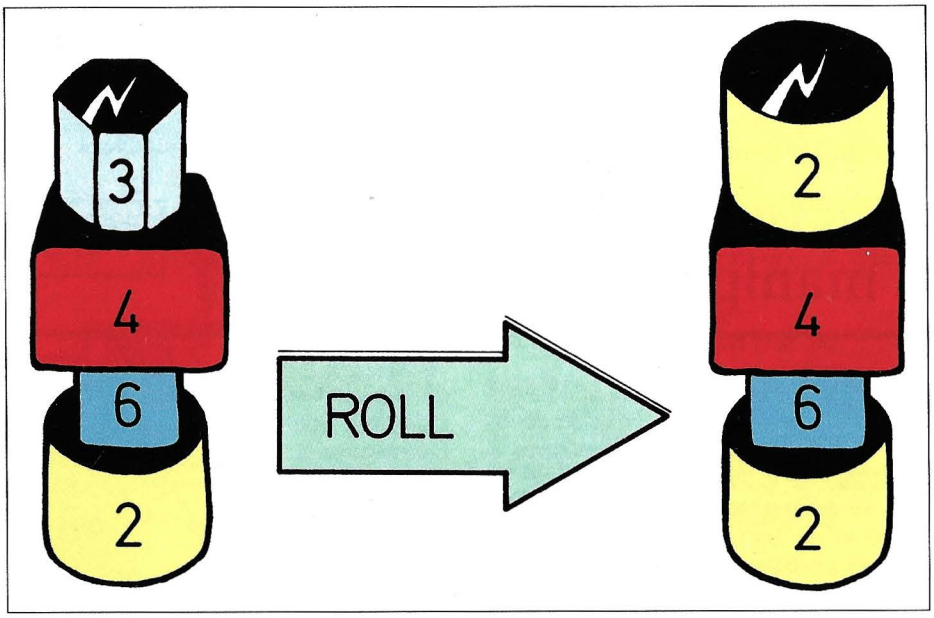
```
2 9 MAX . <CR>
2 9 MAX . 9 OK

5 3 8 MAX MAX . <CR>
5 3 8 MAX MAX . 8 OK
```

MIN Muy semejante al anterior; toma los dos elementos superiores de la pila y coloca el menor de ellos en la cima.

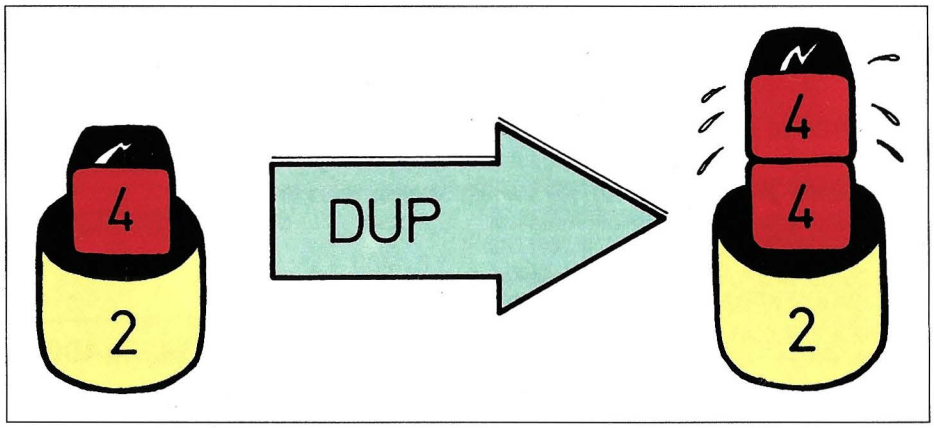
NEGATE Cambia el signo del número situado en la cima de la pila.
Ejemplo:

```
3 NEGATE . <CR>
3 NEGATE . -3 OK
```

Cabe observar que en algunos dialectos FORTH no existe la función NEGATE; si bien, existen otras funciones que realizan el mismo cometido, entre ellas se encuentra MINUS.
En general todas estas funciones son habituales en cualquier dialecto FORTH; hay que constatar que éste es un lenguaje muy uniforme. Por lo demás, existe la ventaja de que si nuestra versión de este lenguaje no posee una determinada palabra, es muy sencillo crearla, con lo cual el problema desaparece.



Al ejecutar la orden ROLL, el ordenador toma el número "n" situado en la cima de la pila (3 en nuestro caso); acto seguido, duplica el número situado "n" posiciones a partir de la cima de la pila y deposita la copia en la posición superior de la misma.



La palabra DUP duplica el dato localizado en la cima de la pila, siempre y cuando su valor no sea cero.

En nuestros días existen ya versiones del lenguaje FORTH adaptadas a múltiples microordenadores, incluso a equipos domésticos como es el caso del ZX- SPECTRUM.

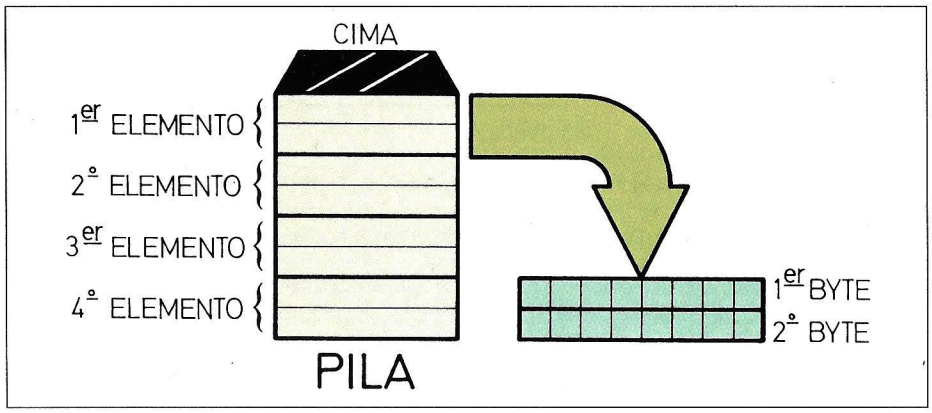
EL TRABAJO CON LA PILA

Una vez estudiadas las palabras FORTH que permiten la realización de operaciones aritméticas, pasamos a describir una serie de palabras FORTH especializadas en realizar diferentes manipulaciones de la pila.

Este nuevo tipo de instrucciones es fundamental en el lenguaje FORTH; no hay que perder de vista, como hemos indicado anteriormente, que la pila constituye la base de trabajo para todos los cálculos, ya que de ella se extraen los operandos.

Con una de estas nuevas palabras hemos trabajado ya anteriormente; se trata de la palabra "." que permite visualizar en la pantalla el número situado en la cima de la pila.

DROP Elimina el número que se encuentra encima de la pila, con lo cual éste ya no se tiene en cuenta.



En la mayor parte de las versiones FORTH, cada elemento de la pila consta de un total de 16 bits. Ello significa que la pila utiliza realmente dos posiciones de memoria para el almacenamiento de cada dato.

Ejemplo:

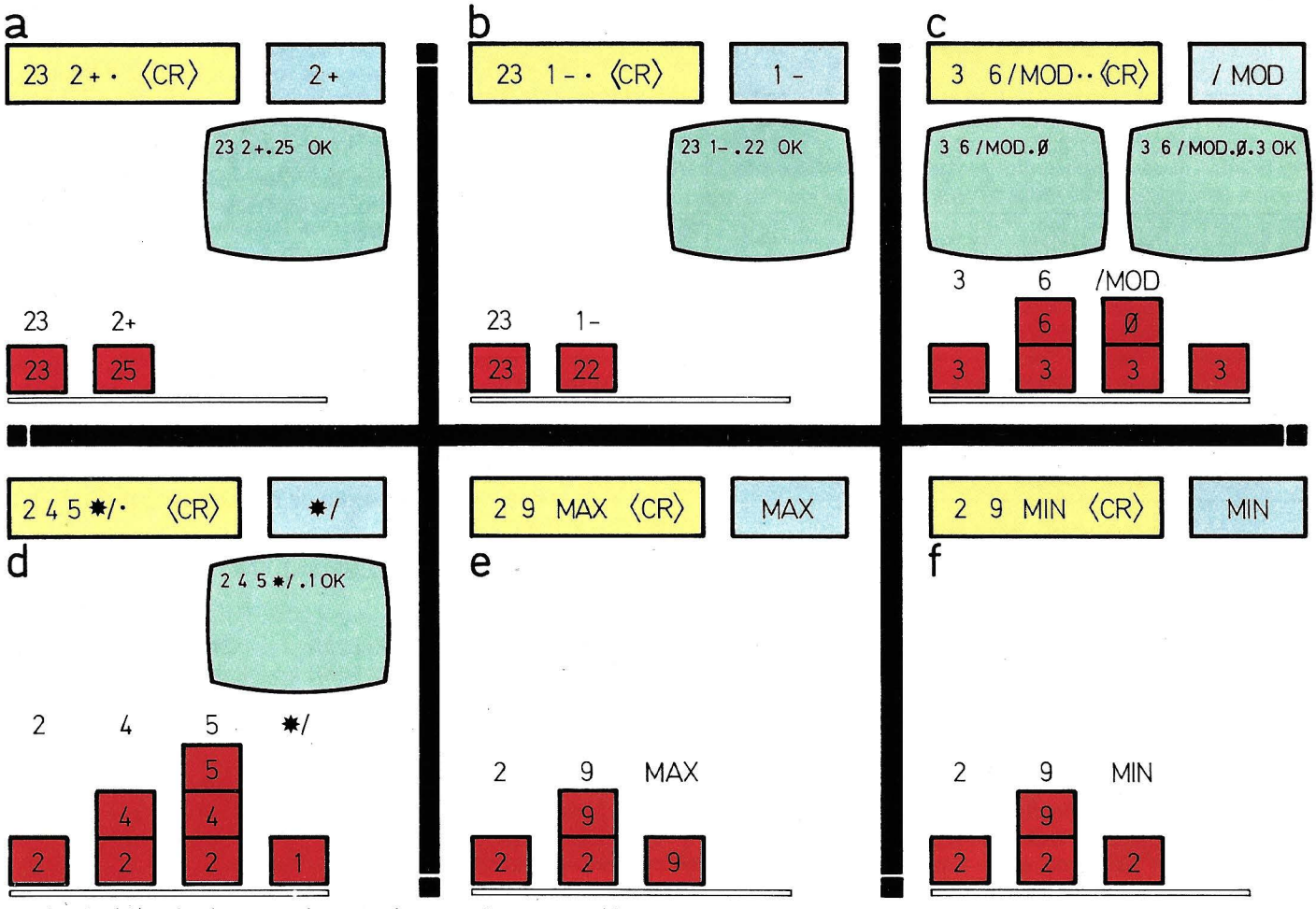
```
2 3 5 DROP . . <CR>
2 3 5 DROP . 3 . 2 OK
```

SWAP Intercambia los dos números situados en las posiciones superiores de la pila.

Ejemplo:

```
3 5 SWAP . . <CR>
3 5 SWAP . 3 . 5 OK
```

OVER Obtiene una copia del elemento situado en la segunda posición a partir de la cima de la pila, y coloca este duplicado



Evolución de la pila al ejecutar determinadas operaciones matemáticas.

TABLA DE ORDENES FORTH

PALABRA	DESCRIPCION	TIPO
1+	Suma uno al elemento situado en la cima de la pila.	Instrucción aritmética.
2+	Suma dos al elemento situado en la cima de la pila.	Instrucción aritmética.
1–	Resta uno al elemento situado en la cima de la pila.	Instrucción aritmética.
2–	Resta dos al elemento situado en la cima de la pila.	Instrucción aritmética.
/MOD	Divide dos números y da el cociente y el resto.	Instrucción aritmética.
*/	Divide y multiplica.	Instrucción aritmética.
*/MOD	Divide y multiplica dando además el resto.	Instrucción aritmética.
ABS	Entrega el valor absoluto del número situado en la cima de la pila.	Instrucción aritmética.
MAX	Deposita en la cima de la pila el mayor de los dos números situados más cerca de la cima.	Instrucción aritmética.
MIN	Deposita en la cima de la pila el menor de los dos números situados más cerca de la cima.	Instrucción aritmética.
NEGATE	Cambia el signo del número situado en la cima de la pila.	Instrucción aritmética.
DROP	Elimina el número situado en la cima de la pila.	Manipulación de la pila.
SWAP	Intercambia los dos elementos situados más cerca de la cima de la pila.	Manipulación de la pila.
OVER	Duplica el segundo elemento desde la cima de la pila y coloca este duplicado en la cima.	Manipulación de la pila.
ROT	Ejecuta una rotación de los tres elementos situados más cerca de la cima.	Manipulación de la pila.
ROLL	Toma un número N de la cima de la pila y busca el elemento situado en la posición enésima de la misma.	Manipulación de la pila.
DUP	Duplica la cima de la pila.	Manipulación de la pila.
CONSTANT	Se emplea para definir constantes.	Palabra de definición.

en la cima de la pila. Como resultado, la pila mostrará en la cima un número igual al tercer elemento de la misma.
Ejemplo:

```
2 3 OVER . . . <CR>
2 3 OVER . 2 . 3 . 2 OK
```

ROT Ejecuta una rotación de los tres elementos más próximos a la cima de la pila: coloca al tercer elemento en la cima y desplaza a los otros dos una posición hacia abajo.

Ejemplo:

```
1 2 3 ROT . . . <CR>
1 2 3 ROT . 1 . 3 . 2 OK

1 2 3 ROT ROT ROT . . . <CR>
1 2 3 ROT ROT ROT . 3 . 2 . 1 OK
```

ROLL Toma el número n situado en la cima de la pila y lo utiliza como referencia para coger el número situado n posiciones a partir de la cima de la pila y duplicarlo, depositando una copia en la posición superior de la pila.

Ejemplo:

```
1 2 3 3 ROLL . <CR>
1 2 3 3 ROLL . 1 OK
```

DUP Duplica la cima de la pila en el caso de que ésta no sea cero.
Ejemplo:

```
3 DUP . . <CR>
3 DUP . 3 . 3 OK
```

R Copia la cima de la pila de retorno en la pila de operación.

CONSTANTES

El lenguaje FORTH también permite definir constantes. Este concepto puede parecer un tanto extraño, especialmente para las personas acostumbradas a trabajar exclusivamente en BASIC, ya que tendrán la tendencia a asociar estas constantes con las variables
En esencia, este artificio permite otorgar un nombre a un valor específico, valor que no puede ser cambiado a lo largo del proceso. Tal posibilidad hace posible asignar un nombre a un número, lo que facilitará las referencias al mismo dentro del programa.
Para realizar una definición de esta índole, hay que empezar trasladando el número que se pretende definir a la cima de la pila; acto seguido hay que indicar al ordenador que se desea definir una constante utilizando para ello la palabra CONSTANT seguida por el nombre que se desea otorgar a la constante. Veamos un ejemplo ilustrativo, tanto de definición de constantes como de su uso práctico.

```
23 CONSTANT EDAD <CR>
56 CONSTANT BAR <CR>
```

```
23 EDAD + . <CR>
23 EDAD + . 46 OK
56 BAR _ . <CR>
56 BAR _ . ∅ OK
■
```

Apple ProDOS (2)

Comandos de volumen

El elemento base de almacenamiento de información, en el caso del sistema operativo ProDOS, es lo que Apple denomina el "volumen". En la práctica, este elemento se reduce a un disco rígido o a un disquete.

En el caso del ProDOS, los discos flexibles tienen una distribución distinta a la que puede encontrarse en otros sistemas operativos. Concretamente, están distribuidos en un total de 35 pistas con 16 sectores por pista; siendo el sector la división mínima a que accede la cabeza de la unidad de disco en una sola operación de lectura/escritura.

Para acceder a los comandos de volumen del sistema operativo ProDOS, hay que seleccionar la opción FILER del menú principal del sistema. Esta dará paso a un nuevo menú, en el cual se seleccionará la opción Volume Commands; acto seguido aparecerá un tercer menú, dentro del que será posible elegir el comando adecuado para llevar a cabo la acción deseada. Dando por sentado que nos encontramos en dicho menú, pasaremos al apartado siguiente.

DESCRIPCION DE LOS COMANDOS

A continuación se relacionan los diferentes comandos de volumen que el sistema operativo ProDOS brinda al usuario, así como una breve descripción de los mismos.

- FORMAT

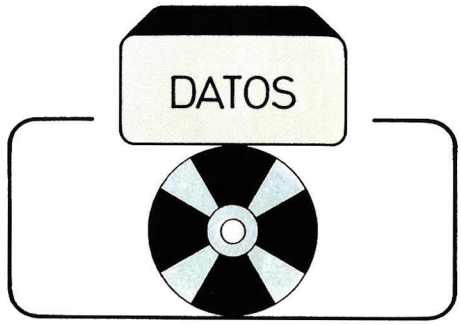
La función del comando FORMAT es la de preparar el disquete para su empleo por parte del sistema operativo, escribiendo sobre el mismo lo que podríamos denominar guías electrónicas, y dividiéndolo en una serie de bloques de tamaño estándar en los cuales se almacenará la información.

Puede parecer sorprendente el hecho de que los disquetes o discos rígidos no vengan ya formateados por el propio fabricante. El motivo es que éstos pueden ser empleados por una pléyade de ordenadores, y cada uno de estos distribuye el espacio de una forma distinta; en consecuencia, un formateado de origen no parece muy práctico.

Una vez formateado, el disquete es reconocido por el sistema operativo y la información puede ser almacenada sobre él sin problema alguno.

Una característica del comando FORMAT reside en el hecho de que elimina cualquier tipo de información contenida en el disquete, sin que sea posible recuperarla. Por ello, antes de activar tal operación, es conveniente consultar el directorio de los ficheros contenidos en el disquete, de forma que se sepa cuál es la información contenida en el mismo y pueda verificarse si hay algún fichero que no deba ser destruido.

El proceso a seguir para formatear un disquete con el sistema operativo ProDOS es el que se indica a continuación. Partiendo del submenú de comandos de volumen, se accionará la tecla F (Format). Hecho esto, aparecerá una pantalla en la cual se solicita la información correspondiente a la unidad que aloja el disco a



El sistema operativo ProDOS está especialmente capacitado para manipular la información residente en discos magnéticos de tipo rígido.

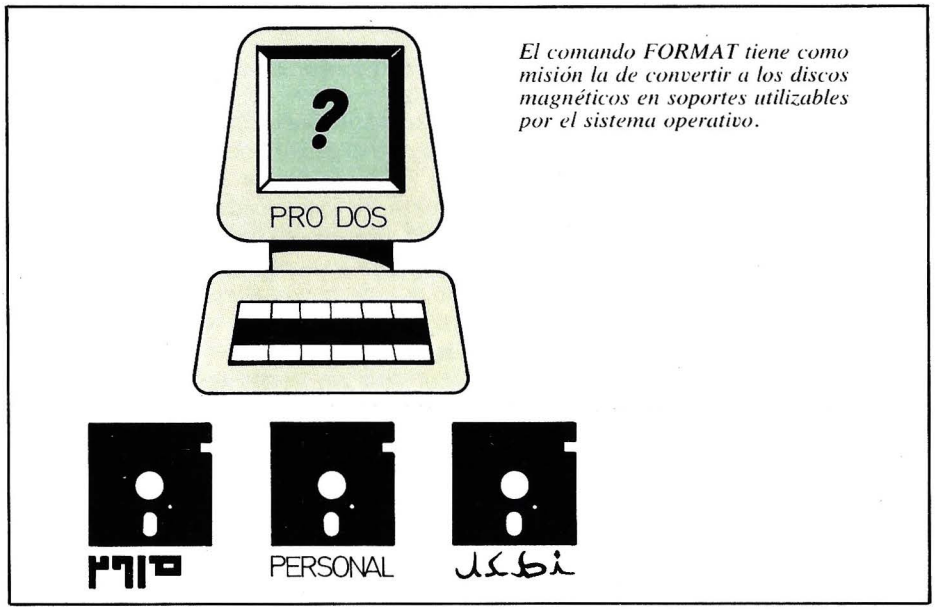
formatear, así como el nombre que se le va a dar al volumen. En ambos casos, el sistema genera estos valores por defecto, aunque el usuario siempre puede introducir los datos que desee. La unidad de disquete es identificada por el slot al que está asociado su controlador, y dentro de éste, por su número de unidad (1 ó 2). Por lo que se refiere al nombre del volumen, éste puede ser indicado por el usuario o bien puede tomarlo el sistema operativo por defecto, siendo este último un nombre de la forma:

BLANKXX

donde XX representan un número como, por ejemplo: 12, 20, etc. Una característica notable, aunque peligrosa, del método de formateo empleado, reside en el hecho de que si se efectúa el formateado

Selección de los comandos de volumen a través de los sucesivos menús del sistema operativo ProDOS.

Bajo la perspectiva del ProDOS, el elemento base de información es el denominado "volumen". En la práctica, este elemento se reduce a un disco rígido o a un disquete.



El comando FORMAT tiene como misión la de convertir a los discos magnéticos en soportes utilizables por el sistema operativo.

consecutivo de una serie de disquetes, y se elige la modalidad de dar nombres por defecto, los nombres asignados resultan de la forma BLANK 12, BLANK 13, BLANK 14, esto es, siguen una secuencia. Ello presenta el peligro de que si, al cabo de un cierto tiempo, se vuelve a realizar un proceso de formateado, pueden darse idénticos nombres a otros disquetes, con el consiguiente perjuicio para el control de la información. No hay que olvidar que es sumamente importante para el usuario de cualquier ordenador de sus disquetes estén etiquetados de forma clara y den una idea del contenido de la información que transportan.

Otra característica incorporada al comando FORMAT es la existencia de una protección contra el formateado de un disquete que contenga datos; en tal caso, la pantalla del ordenador mostrará un mensaje del tipo:

DESTROY "XXX" ? (Y/N)

siendo XXX el nombre del volumen.

Esta solicitud de confirmación reducirá el posible riesgo de confusión de un disquete virgen con otro ya formateado; obviamente, en su aspecto externo ambos disquetes son iguales. El referido mensaje recordará al usuario la necesidad de examinar el directorio de ficheros para comprobar si el disco contiene información de interés.

Una vez completado el proceso de formateo, el ordenador presentará el siguiente mensaje:

FORMAT COMPLETE

Si se quiere formatear un nuevo volumen bastará con repetir de nuevo el proceso. Si no es así, y deseamos volver al menú de comandos de volumen, habrá que pulsar la tecla ESCAPE.

- COPY

La misión de este nuevo comando es obtener copias completas de un disquete o disco rígido, transfiriendo todos los ficheros residentes en el disco original a la copia. Este proceso es sumamente importante, dado que permite duplicar la información y garantizar así que ésta no se perderá por cualquier error o pérdida accidental de un disquete. La seguridad es un aspecto fundamental. Imagine, por ejemplo, que el disquete que contiene los datos de las facturas a cobrar por un médico es destruido por el fuego, sin que exista un duplicado del mismo. La simple consideración de este hecho nos permite darnos cuenta de la importancia del referido comando.

La información necesaria para llevar a cabo este proceso es solicitada por el sistema operativo ProDOS, a través de un panel que aparece tras pulsar C en el menú de comandos de volumen. Este panel solicita tres datos:

— Unidad de disquete en que se encuentra el original.
— Unidad de disquete en que se encuentra el disco que va a recibir la copia.
— Nombre que se va a dar a la copia.

En cualquiera de estos tres datos, el sistema operativo ProDOS admite opciones por defecto; estas se activan pulsando la tecla RETURN como respuesta a su solicitud.

La opción por defecto correspondiente al nombre de la copia coincide con el propio

nombre del original, aunque es conveniente distinguirlo del mismo ya que, de no ser así, nunca se sabría cuál de los dos disquetes contiene la última revisión de los datos, con la consiguiente pérdida de consistencia en los resultados obtenidos.
Este comando representa como característica interesante el hecho de que, antes de iniciar el proceso, formatea el disquete sobre el que se va a obtener la copia. De ahí que si el disquete de destino está ya formateado, el ordenador preguntará:

DESTROY "XXX" Y/N ?

siguiendo el proceso que se indicó para el caso del comando FORMAT.
El comando COPY indica en todo momento cuál es la operación que se está llevando a cabo, por medio de una serie de mensajes, que aparecen sucesivamente en la pantalla; estos son:

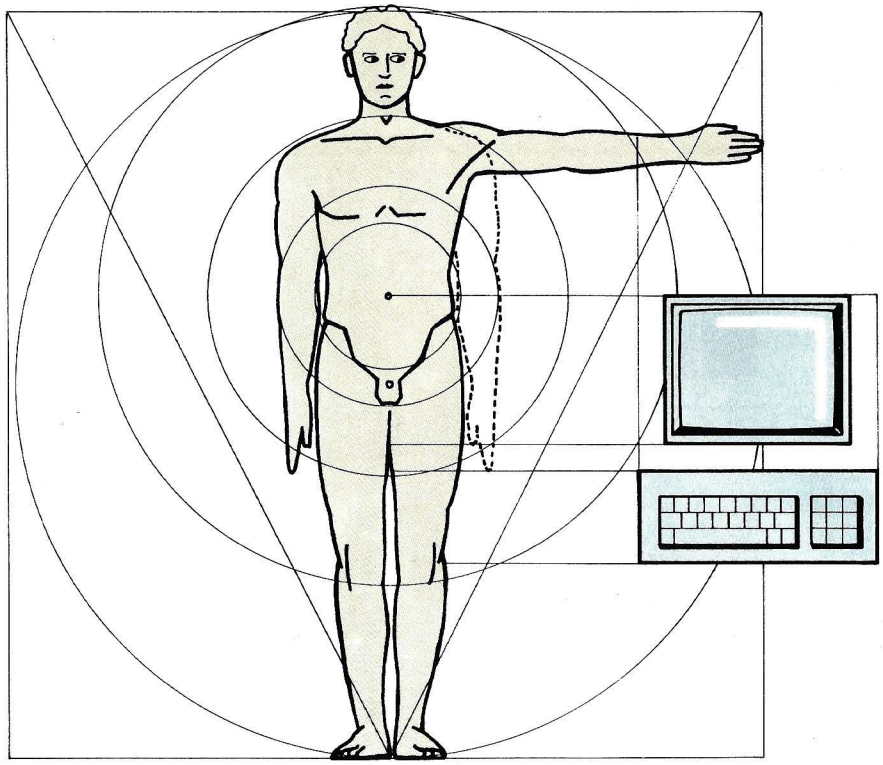
FORMATTING
READING
WRITING

Una vez finalizado el proceso, aparecerá también el mensaje al efecto:

COPY COMPLETE

La importancia de la ergonomía

En los últimos años se han realizado abundantes estudios enfocados a que el material informático cumpla una serie de requisitos mínimos que contribuyan a mejorar la comodidad del usuario. Las conclusiones derivadas de esta investigación están siendo contempladas, progresivamente, por la mayor parte de los fabricantes.
Estos estudios entran dentro de un campo más amplio denominado ergonomía: una disciplina que trata del estudio del trabajo y las herramientas necesarias para llevarlo a cabo, de manera que resulten más cómodas y confortables para el trabajador, tanto a nivel anatómico como fisiológico y psicológico.
Por lo tanto, la ergonomía no sólo se ocupa del trabajo en sí, sino también del entorno en que se mueve la persona, procurando crear una sensación de bienestar físico y mental por medio de una adecuada climatización, iluminación, eliminación de ruidos molestos y funcionalidad del mobiliario.
En el campo de la informática, la aplicación de principios ergonómicos a los terminales ha hecho que estos se adapten mejor al usuario y que este se encuentre más cómodo en su trabajo.
Aunque no hay ningún terminal perfecto, debido a que la necesidad de adaptarse a las características del usuario conlleva no una solución única, sino tantas como usuarios, sí existen una serie de reglas generalizables.
Respecto a las pantallas, éstas deben permitir su inclinación en sentido vertical, así como el giro en sentido horizontal para lograr su mejor adaptación al lugar de trabajo y al usuario. También es conveniente que dispongan de una rejilla antirreflectante o de una visera apropiada.
La mayoría de las pantallas están constituidas por un tubo de rayos catódicos, similar al existente en los televisores domésticos; estos deben cuidar sus características visuales para que no produzcan cansancio ni dolor de cabeza al operador. Para ello, los caracteres que aparecen en pantalla deben de estar formados por una matriz de puntos, preferiblemente cuadrados o redondos, de al menos 7 × 9 puntos. En general estos caracteres deben aparecer en color oscuro sobre fondo claro o de color amarillo ámbar sobre fondo marrón, como viene sucediendo últimamente ya que la visión se adapta mejor a ellos. En todo caso, hoy en día, las pantallas más utilizadas son las de fósforo verde.
En cuanto al teclado, éste debe ser independiente del mueble que aloja la pantalla, y debe reservar una zona especial de teclas numéricas para aplicaciones de gestión. Su posición no será totalmente horizontal, sino que ha de permitir una inclinación entre 5 y 11 grados, siendo la parte de perfil más bajo la más cercana al usuario.
Con respecto a las teclas, su tamaño debe estar comprendido entre los 12 × 12 y los 15 × 15 milímetros, y deben ser de color mate para evitar reflejos. También es conveniente que algunas teclas sean diferentes al tacto para ayudar al posicionamiento de los dedos, y al accionarlas debe emitirse una leve señal acústica que indique que se ha llevado a cabo su pulsación.
La ergonomía no sólo se ha instalado en las máquinas, sino que también ha entrado en el campo del Software. Cada vez los programas resultan más estéticos y "amigables", requieren una menor preparación técnica del operador y mejoran la comunicación hombre/máquina.



Ahora, el usuario puede elegir entre repetir el proceso una vez más o volver al menú de comandos de volumen. Esto último se consigue pulsando la tecla de ESCAPE.

- LIST

El comando LIST tiene una función eminentemente de ayuda al usuario, particularmente en el caso de que se trabaje con un sistema con múltiples unidades de disquete conectadas. Su ejecución efectúa un rastreo de las unidades de disco y señala, a continuación, qué volumen se encuentra localizado en cada unidad de disquete.
Para acceder a este comando, se pulsa la tecla L dentro del menú de comandos de volumen. Tras ello, aparece en la pantalla una lista de las unidades de disquete conectadas a cada slot indicando, para cada unidad, el nombre del volumen en ella contenido. La lista empieza con el slot y la unidad de disquete que contienen el sistema operativo; de tal forma que si este sistema se encuentra en el slot 3 unidad 1, esta será la primera indicada. Dicho proceso es controlado por el denominado programa Monitor, el cual, durante la fase de arranque, se encarga de localizar la unidad en la que está contenido el disquete con el sistema operativo.
La lista también señala las unidades que se encuentran vacías y aquellas en las que está alojado un disquete no reconocible por el sistema operativo ProDOS, en cuyo caso genera el siguiente mensaje:

NO DIRECTORY

- RENAME

A este comando se accede pulsando la tecla R dentro del menú de comandos de volumen. Su misión es la de modificar el nombre dado a un volumen determinado cuando se efectuó su formateo. Para ello, el sistema operativo presenta una pantalla en la cual solicita la identificación de la unidad de disquete en la que éste se encuentra, así como el nuevo nombre a otorgar volumen. Hecho esto, será ejecutado el cambio de nombre, indicándolo por medio del mensaje:

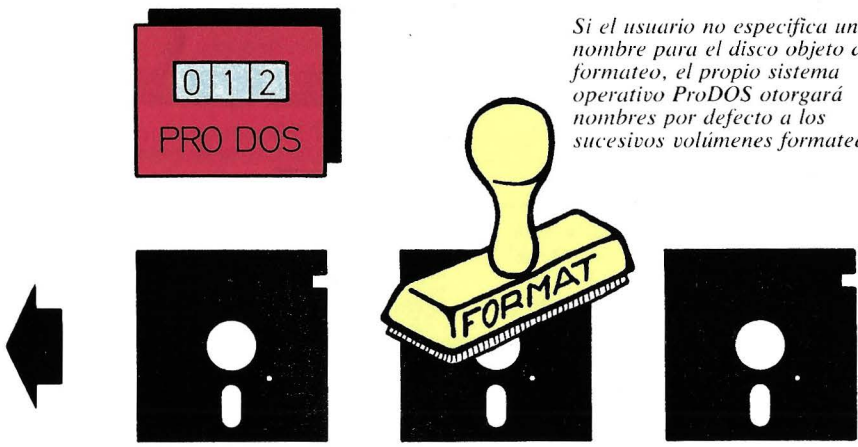
RENAME COMPLETE

- BAD BLOCKS y BLOCK ALLOCATION

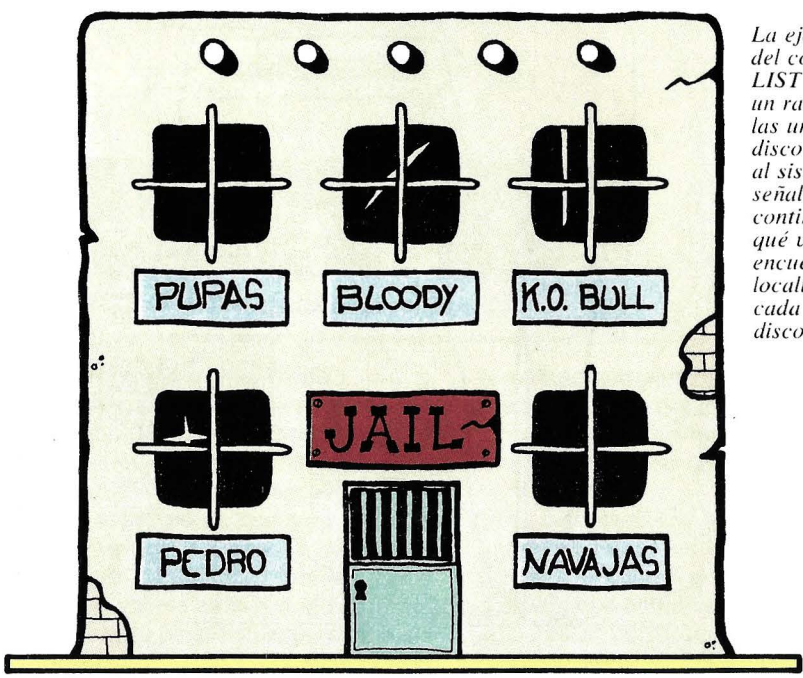
Estos dos camandos están relacionados, como su propio nombre indica, con los bloques en los que se divide un disquete. El primero, al que se accede pulsando la letra D del menú de comandos de volumen, tiene por misión señalar el número de identificación de los bloques defectuosos que pueda incluir el disquete alojado en la unidad que especifique el usuario.
El comando BLOCK ALLOCATION, al que se accede pulsando la tecla B una vez situados en el menú de comandos de volumen, muestra una lista que revela los bloques de un disquete que están ocupados por ficheros, cuántos están disponibles y cuál es el número total de bloques del disquete. De esta forma podemos conocer si en el disquete queda o no espacio suficiente para almacenar nuevos ficheros.
En ambos comandos, para volver al menú principal basta con pulsar la tecla de ESCAPE.

- COMPARE VOLUMES

A este comando se accede pulsando la tecla K del menú de comandos de volumen. Su función es la de efectuar una comparación, byte por byte, entre dos volúmenes y señalar las diferencias entre los mismos. Es una forma de averiguar si dos copias son exactas o si una se ha actualizado y la otra no.
El sistema operativo solicita, a través de los adecuados mensajes en la identificación de las unidades en los que se encuentran los volúmenes a comparar; tras recibir esta información genera una lista de los bloques que difieren en ambos volúmenes.



Si el usuario no especifica un nombre para el disco objeto de formateo, el propio sistema operativo ProDOS otorgará nombres por defecto a los sucesivos volúmenes formateados.



La ejecución del comando LIST efectúa un rastreo entre las unidades de disco asociadas al sistema y señala, a continuación, qué volumen se encuentra localizado en cada unidad de disco.

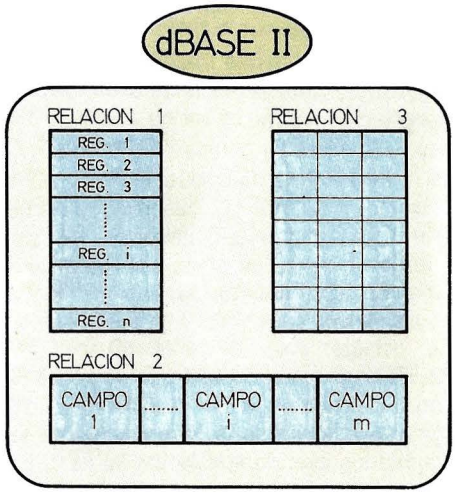
dBASE II (1)

Introducción a la base de datos dBASE II (1)

Las aplicaciones denominadas bases de datos para ordenadores personales no siempre son auténticas bases de datos. En algunos casos son programas que por su funcionamiento más bien debieran denominarse gestores de ficheros.
La aplicación dBASE II, puede considerarse como una auténtica base de datos relacional. A lo largo de éste y de los dos próximos capítulos, estudiaremos sus principales características. De su análisis podremos concluir que si no todas, dispone de casi todas las opciones que se exigen a una base de datos relacional.

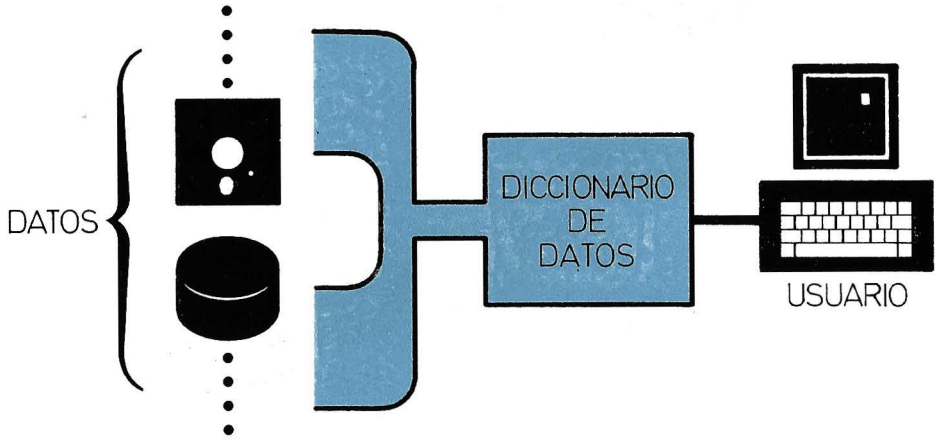
ALGUNAS CARACTERISTICAS DESTACABLES DEL dBASE II

En el mundo de los grandes ordenadores, el software más relevante de los años 80 se concreta en las aplicaciones de bases de datos con lenguajes de cuarta generación; esto es: una base de datos, que generalmente suele ser de tipo relacional, y un lenguaje de programación de muy alto nivel que permite simplificar enormemente la programación.
Desde luego, dBASE II no pertenece a este tipo de aplicaciones, en cambio sí cabe considerarla como un eficaz sustitutivo para ordenadores personales. Se ha mencionado ya que dBASE II cumple como base de datos relacional; en cuanto a la segunda propiedad —el lenguaje de cuarta generación—, dBASE II no dispone de él, pero sí incluye un vocabulario de comandos de programación estructurada que, aún no respondiendo al alto nivel exigido a un lenguaje de cuarta generación, sí



dBASE II es una base de datos relacional; por lo tanto, puede gestionar información tabular, organizando los datos en relaciones integradas por registros que, a su vez, están compuestos por campos.

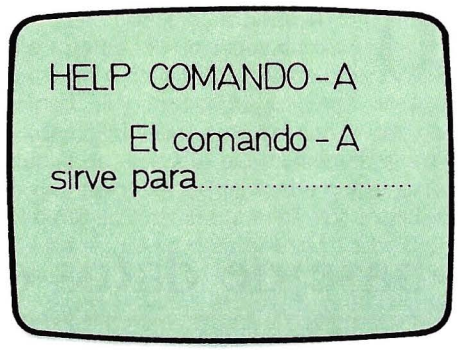
permite realizar programas bien estructurados que aumenten notoriamente las posibilidades de extracción de información de la base de datos.
Otro de los productos más novedosos para cualquier tipo de ordenador son los diccionarios de datos. Se trata de programas destinados a facilitar la administración de los datos existentes en una organización. Pues bien, dBASE II también incorpora comandos especializados en la administración de las distintas relaciones y, por lo tanto, de los distintos datos disponibles por el usuario. La misión de estos comandos es permitir la definición, uno por uno, de todos los campos de cada registro de una relación. Para ello se debe especificar el nombre, el tipo y otras características de los campos. De esta forma el usuario podrá, en cualquier momento, comprobar la estructura de información que tiene disponible y, de esta forma, optimizar tanto la explotación de dicha estructura como la calidad de la información, eliminando redundancias e inconsistencias.
Otra característica importante del dBASE II reside en su poderoso comando de ayuda HELP. Para invocarlo basta, simplemente, con teclearlo seguido por el nombre de un comando u otra entrada; inmediatamente, el programa responderá con



Un diccionario de datos permite al usuario tener una visión clara de la estructura que tiene la información en su base de datos.

claridad a la cuestión que le haya sido planteada. De esta forma, el usuario, en caso de duda, puede consultar al propio programa sin necesidad de desplazarse en busca de otra documentación.

OPERACIONES ELEMENTALES



*Mediante el comando **HELP**, el programa dBASE II resuelve las posibles dudas que el usuario pueda tener sobre cualquier concepto relativo a otros comandos.*

- *Creación de relación*

Evidentemente, la más elemental de las operaciones permitidas por dBASE II es la creación de una relación (fichero), para lo cual dispone del comando CREATE. Después de haber sido invocado por el usuario, el ordenador responderá preguntando: "ENTER FILENAME". El usuario podrá, en este momento, asignar cualquier nombre que comience por una letra y no supere los 8 caracteres.
A continuación, el programa solicitará la estructura de la relación mediante la siguiente frase: "ENTER RECORD STRUCTURE AS FOLLOWS". Para ello irán apareciendo, numerados desde el uno en adelante, los distintos campos que contendrá la relación. En cada uno de ellos, el usuario debe especificar:

1. Un nombre de hasta 10 caracteres de longitud.

2. El tipo de dato que almacenará el campo, indicando: C, cuando el dato sea de tipo carácter (alfanumérico), N, para datos de tipo numérico, o L para datos lógicos (sólo pueden adoptar los valores verdadero o falso).

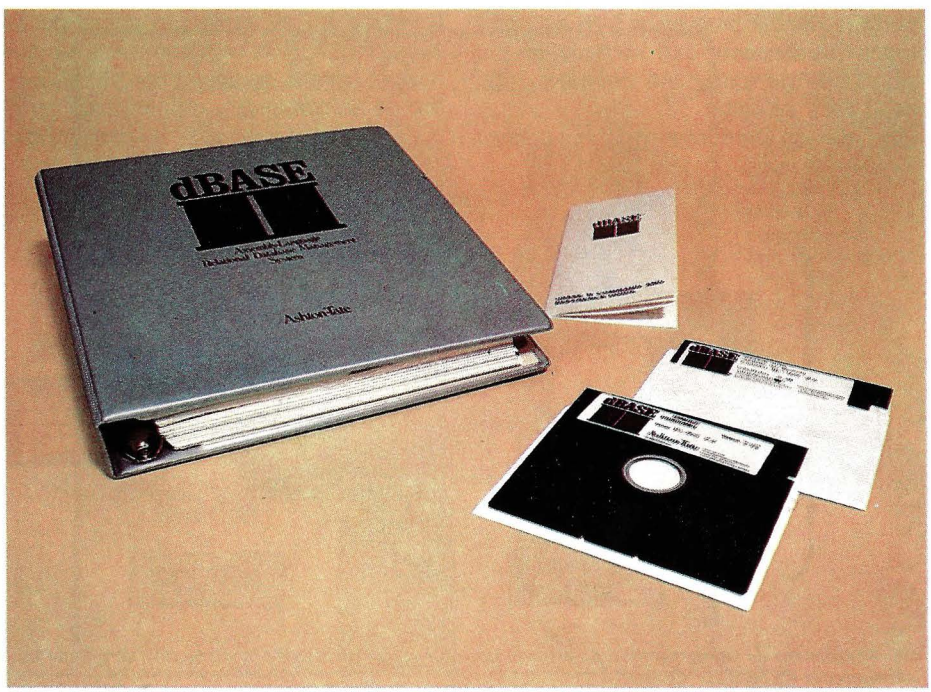
3. Longitud del campo, que no podrá sobrepasar nunca de 254 posiciones.
Una vez finalizada la introducción de las características de la relación, el dBASE II creará un fichero denominado: Nombre.DBF, en donde el parámetro Nombre coincide con el indicado por el propio usuario.

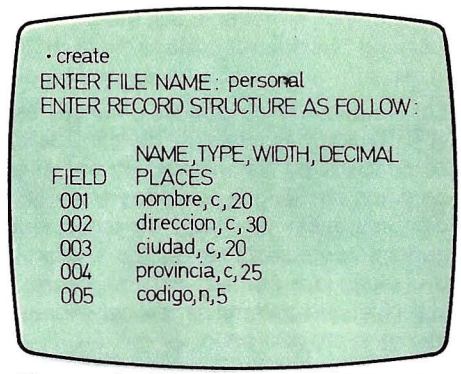
- *Introducción de información*

La segunda operación elemental sobre una relación suele consistir en introducir la información concreta. En el dBASE II, dicha introducción se realizará de dos formas distintas, según se disponga o no de editor de pantalla completa (full screen editor):

— Con editor: en la parte superior izquierda de la pantalla aparecerá el literal "RECORD NNNNN", donde NNNNN representa el número de orden del registro que se va a introducir; e inmediatamente debajo, los nombres de los distintos campos, con el espacio disponible para la introducción de datos en cada uno de ellos (espacio encerrado entre un par de símbolos dos puntos (:)).
En este caso, el cursor quedará situado en el primer carácter del primer campo del registro. El usuario podrá introducir la información en los distintos campos recurriendo al teclado del ordenador.

— Sin editor: si no se dispone de un editor de pantalla completa, el dBASE II irá mostrando, uno a uno, los nombres de los campos a introducir, y sólo después de haber tecleado el valor de dicho campo aparecerá una nueva línea para el siguiente. Es obvio que el primer sistema resulta mucho más cómodo para la carga de datos sobre una relación.





*El comando **CREATE** sirve para definir el contenido de los registros de una nueva relación; también, de forma implícita, este comando define un menú de visualización y actualización.*

- *Visualización de registros*

Siguiendo con nuestra exposición de operaciones elementales, la siguiente consiste en la edición de la información previamente almacenada. Para ello, hay que utilizar dos comandos consecutivamente:

1. USE nombre
para indicar al programa el nombre de la relación en la que se desea localizar un registro.

2. EDIT número
para indicar el número de orden del registro que se desea editar.

Tras haber ejecutado estos dos pasos, aparecerá en la pantalla la información asociada al registro solicitado, y el usuario podrá modificar cualquiera de los datos si así lo desea.
Evidentemente, este método para visualizar la información resulta algo "pobre", ya que en la mayoría de los casos el usuario no recordará el número de orden del registro que desea localizar. Sí sabrá, en cambio, el valor concreto o aproximado de alguno de sus campos; en este caso, la operación a realizar será la siguiente:

- *Listado de registros*

El comando utilizado en este caso se denomina LIST; uno de los más sencillos y poderosos del dBASE II. Su sintaxis es la siguiente:

LIST [OFF] [FOR <expresion>]

Antes de invocarlo es necesario utilizar el

Cuando el usuario desee introducir información en la pantalla, aparecerán en la pantalla el número de registro y los nombres de todos los campos que lo componen...

... A continuación, éste tecleará valores concretos para cada uno de los campos y, al pulsar la tecla de actualización, el registro cumplimentado se incorporará a la relación.

comando USE para especificar el nombre de la relación que se empleará para producir el listado de los registros.
A continuación, vamos a detallar las posibilidades aportadas por los argumentos de este comando:

a) OFF
De utilizarse el argumento OFF, los datos que aparecerán en el listado no incluirán el número de orden de cada registro; en cambio, si se omite el parámetro OFF, la lista incluirá también dicho número.

b) FOR <expresion>
Mediante este argumento, el usuario puede imponer ciertas restricciones que deben ser cumplidas por los registros para aparecer en la lista. Dichas restricciones se especificarán mediante una expresión que incluirá nombre de campos, constantes y los siguientes operadores de relación:

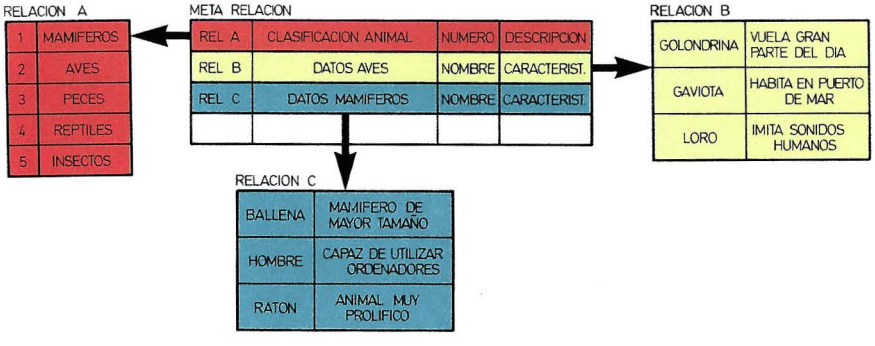
. Menor que (<)
. Mayor que (>)
. Igual a (=)
. Menor o igual que (<=)
. Mayor o igual que (>=)

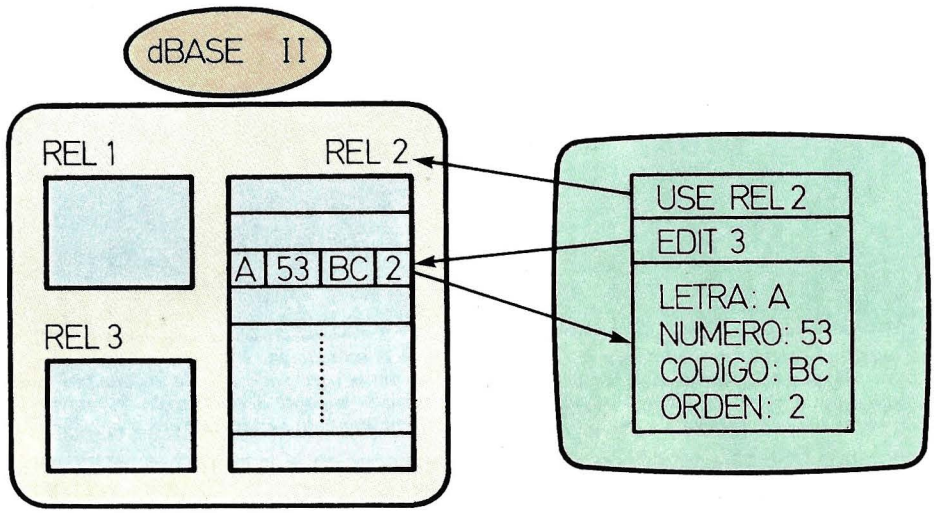
c) Ausencia de argumentos
Si no se incluye ningún argumento en el instante de invocar al comando LIST, este producirá una lista de todos los registros incluidos en la relación previamente espe-

Meta-relaciones (relaciones de relaciones)

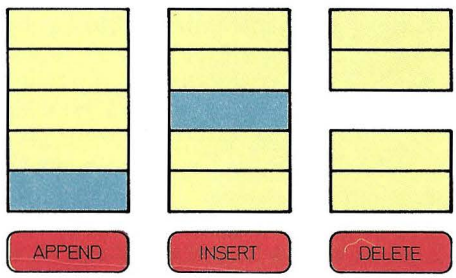
Cuando un usuario de informática dispone de un número considerable de datos, la estructura de información manejada por él puede llegar a tener una considerable complejidad. Esto le sucederá en cualquier caso: tanto si utiliza un sistema de base de datos, como si emplea un gestor de ficheros, e incluso si los ficheros los mantiene con programas específicamente diseñados para ese cometido.
Los grandes soportes de almacenamiento que equipan a los ordenadores presentes en los centros de cálculo se organizan mediante unos ficheros especiales, denominados VTOC (VIRTUAL TABLE OF CONTENS). Este tipo de ficheros se utiliza también en algunos ordenadores personales para que el sistema operativo organice la información. En ellos se almacena la ubicación física y otras características de los distintos ficheros almacenados en el soporte.
Si el sistema operativo es capaz de organizar eficientemente los ficheros de un soporte mediante ficheros VTOC, ¿por qué no puede hacer lo mismo un usuario para organizar su estructura de información?

Evidentemente la idea es buena, y en el caso de las bases de datos relacionales esta técnica se plasma en las metarrelaciones, que están compuestas por registros cuyos campos contienen información sobre otras relaciones.





Para editar los valores de un registro es necesario, en primer lugar, identificar cuál de las relaciones contiene al registro y, a continuación, el número de orden de dicho registro.



Los comandos fundamentales de actualización permiten añadir un nuevo registro al final de la relación (APPEND), en una posición intermedia (INSERT) o eliminarlo (DELETE).

cificada mediante el comando USE, figurando también sus números de orden.
Otro comando alternativo para recuperar información de una relación es el denominado DISPLAY. Este puede ser invocado de la siguiente forma:

DISPLAY [ALL / RECORD N / NEXT M] [OFF] [FOR <expresion>]

A simple vista se observa que el segundo y tercer argumentos del comando DISPLAY tienen el mismo cometido que en el caso del comando LIST. La novedad la aporta el primer argumento, el cual puede tomar los siguientes valores:

a) ALL
Implica que el comando DISPLAY actuará sobre todos los registros de la relación.

b) RECORD N
Si el primer argumento es de este tipo, el comando DISPLAY se ejecutará sobre el registro número N de la relación.

c) NEXT M
Se utiliza para que el comando DISPLAY actúe sobre los M registros de la relación situados a continuación del registro activo.

- *Comandos de posicionamiento*

Como ya hemos señalado en el caso del comando DISPLAY, existen determinadas operaciones que se ejecutarán sobre un registro específico de la relación, al que denominaremos registro activo. El dBASE II incorpora dos comandos básicos para posicionar el registro activo; por supuesto, en cualquiera de ambos casos, antes de utilizarlos el usuario debe especificar en qué relación se efectuará el posicionamiento mediante el comando USE.
El primer comando de posicionamiento se puede ejecutar tecleando indistintamente las palabras GO o GO TO, seguidas por un argumento que podrá tomar tres valores distintos:

a) TOP
Implica que el registro activo debe coincidir con el primero de la relación.

b) BOTTON
El registro activo se situará sobre el último de la relación.

c) N
Si después del comando GO se incluye un número N, el registro activo pasa a ser precisamente el N-esimo de la relación.
El segundo comando de posicionamiento del dBASE II se identifica mediante SKIP. Siempre actúa operando un desplazamiento del registro activo; esto es: si el registro activo es, inicialmente, el número 10 y se ejecuta SKIP 3, el nuevo registro activo será el 13; en cambio, si se ejecuta SKIP −3, el nuevo registro activo será el 7.

- *Actualización de datos*

Aunque hasta ahora tan sólo hemos descrito una pequeña parte de los comandos y operaciones que pueden realizarse con el dBASE II, contando con los tres nuevos comandos de actualización que se describen seguidamente, estaremos ya en disposición de utilizar con eficacia la base de datos.

1. APPEND
Mediante el comando APPEND, el usuario puede introducir nuevos registros en la relación que previamente haya sido activada mediante la orden USE. El orden en el que quedará almacenado el nuevo registro será inmediatamente superior al último existente.

2. INSERT
Su misión es análoga a la del comando anterior; es decir, permite introducir nuevos registros en una relación. La única diferencia radica en que con el comando APPEND el nuevo registro se incluirá después del último, mientras que con el comando INSERT la inclusión se realizará después del registro activo y, en consecuencia, el número de orden de todos los posteriores se incrementará en una unidad.

3. DELETE
El comando DELETE permite eliminar información de una relación; su sintaxis es la siguiente:

DELETE [ALL / RECORD N / NEXT M] [FOR <EXPRESION>]

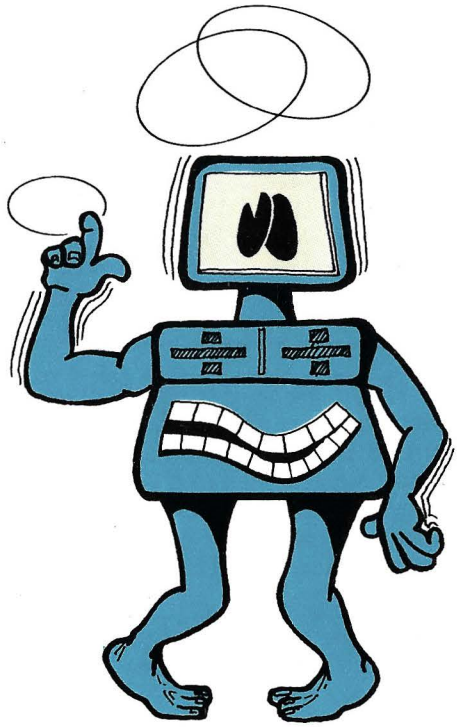
El nivel de eliminación puede afectar a un único registro, a un grupo de registros, o incluso a toda la relación.
El significado de los argumentos del comando DELETE es análogo al de los argumentos del comando DISPLAY descrito anteriormente.

Tratamiento de errores

Detección y supresión de errores en los programas BASIC

Tarde o temprano, todo programador o usuario tropieza con un programa que no funciona debidamente. ¿Qué le ocurre al programa? En principio, cabe pensar que debe tratarse de un error cometido durante la introducción del mismo. Así pues, hay que localizar y subsanar ese error. Esta tarea no siempre resulta fácil. Habitualmente, es necesario disponer de un listado por impresora del programa, y repasarlo a conciencia. La dificultad se ve incrementada cuando el programa ha sido escrito por otra persona. En tal caso, es preciso comprender previamente el funcionamiento del mismo. A lo largo de este capítulo se comentan algunos métodos encaminados a la detección y eliminación de errores, así como la forma de producir y utilizar errores a título voluntario.

INTRODUCCION DE PROGRAMAS



Al igual que cualquier otra tarea realizada por el hombre, la confección de programas para ordenador es también una actividad propensa a acoger errores. En tal caso, el ordenador manifestará un funcionamiento distinto al esperado.

Al introducir los programas por medio del teclado resulta inevitable que se escape algún error de tipo mecanográfico. Cuando se trata de un programa largo, las posibilidades de error aumentan.

Una forma habitual de hacerse con programas consiste en teclear aquellos cuyos listados aparecen en revistas especializadas. Son muchas las publicaciones de esta índole que se encuentran en los quioscos; sin embargo, la calidad de los programas no es siempre la esperada. Ante todo, es recomendable elegir una revista que garantice un buen nivel de calidad y seriedad. De esta forma se evitarán los problemas derivados de un listado defectuoso y/o con errores. Una vez seguida esta precaución, ya se está en disposición de introducir el programa.

En los equipos domésticos, el empleo de cintas en casete de una cierta calidad contribuye a asegurar la grabación, evitando posibles errores.

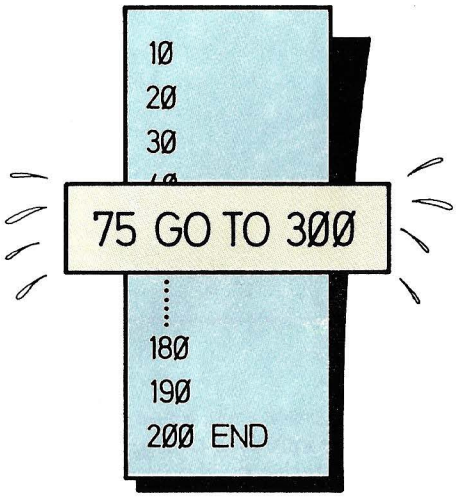
Cuando se teclea en el ordenador un programa de longitud considerable, conviene ir grabando los sucesivos fragmentos en una unidad de almacenamiento externo. Con ello se elimina la necesidad de teclearlo de nuevo si ocurre algún percance. Las unidades más empleadas al respecto son las de disco y casete. En cualquiera de los casos es interesante seguir una serie de normas.

Al realizar una copia de la última versión del programa, conviene no borrar la versión anterior. También es recomendable guardar copias por duplicado y en soportes separados. Con ello se evita la pérdida ocasionada por una grabación o el uso de un soporte defectuosos. Si el ordenador dispone de la posibilidad de verificar la grabación, ésta ha de utilizarse en el proceso.

Antes de comenzar la sesión de trabajo, es conveniente asegurarse de que la unidad de almacenamiento se encuentra debidamente conectada. En algunos aparatos no es posible realizar la conexión cuando el ordenador está en funcionamiento, y el apagado del mismo produce la pérdida de toda la información contenida en su memoria.

Ni que decir tiene que el material de buena calidad evita infinidad de problemas. Es aconsejable observar esta norma tanto con los soportes (discos o casetes) como con las propias unidades de grabación (unidad de disco o magnetófono). Así mismo, hay que mantener el material en buenas condiciones, no exponiéndolo a situaciones que puedan destruirlo o inutilizarlo.

La obtención de copias de seguridad es especialmente importante cuando el programa contiene rutinas en código máquina. Un error en una de esas rutinas puede desembocar en el bloqueo del or-



Es necesario revisar los programas para evitar errores de codificación. Por ejemplo, un salto a una línea que no existe.

denador, del cual sólo se podrá salir desconectando la alimentación del aparato (con la siguiente pérdida del programa).
Una vez completada la introducción de las líneas del listado, llega el momento más importante: la prueba de su funcionamiento. Para realizar dicha prueba es conveniente respetar estrictamente las instrucciones que se adjunten con el listado, y observar lo que va sucediendo en la pantalla. Si no ocurre nada sospechoso y el programa se ejecuta con aparente normalidad: ¡Enhorabuena!.. Aunque esto no suele ser lo más frecuente. Si algo falla, será cuestión de repasar el listado y comprobar si coincide con el original. En el caso de que no exista error mecanográfico, el problema habrá que achacarlo al listado original.
No sólo existen errores de tipo mecanográfico. Otro tipo de errores, harto frecuentes, se deben a la estructura del propio programa. La mayor parte de estos últimos proviene del empleo de saltos (GOTO,GOSUB...).

DEPURACION DE PROGRAMAS

En determinadas circunstancias el mal funcionamiento de un programa se debe a errores de codificación. Puede darse el caso de que la ejecución no pase nunca por una rutina determinada. En el otro extremo puede ocurrir que no se salga nunca de un bucle. En ambos casos, el problema se debe a errores en la desviación del flujo de ejecución.
En la etapa de creación de un programa es conveniente seguir una estructura definida. Delimitar los bloques de subrutinas y no mezclar las distintas partes del programa, proporcionará una mayor claridad al listado. Ha de evitarse la excesiva proliferación de instrucciones del tipo GOTO. Un abuso de estas instrucciones hace que un programa se vuelva difícilmente inteligible para cualquier persona, incluido el propio autor.
La forma más elemental de comprobar si un programa funciona, reside en el "seguimiento" de su ejecución "paso a paso" o "instrucción a instrucción". Este es el mejor procedimiento cuando se trata de comprobar el correcto funcionamiento de una rutina o de un programa corto.
La referida técnica consiste en actuar como lo haría el ordenador; esto es: ir leyendo las sucesivas instrucciones y ejecutarlas sobre el papel, una a una. Para ello es conveniente realizar una tabla de variables en la que se irán reflejando los sucesivos valores que adoptan. Cabe hablar de este método, pues, como de una simulación "a mano".
Realmente, se trata de una técnica muy adecuada; y no sólo para comprobar el buen funcionamiento de la rutina, sino también para comprender su modo de trabajo. El gran problema, no obstante, es que sólo es aplicable con plena eficacia cuando la rutina a estudiar es pequeña.
A título de ejemplo, veamos a continuación cómo se realizaría el seguimiento de una rutina corta, tal como la que sigue:

```
10 FOR I=1 TO 3
10 FOR J=1 TO 3
30 LET A(I,J)=I+J
40 NEXT J
50 NEXT I
```

Para evaluarla aplicando la técnica descrita, se construye una tabla con las variables utilizadas. En este caso, se tienen las dos variables de los bucles y las nueve de la matriz. Se supone que la matriz está inicializada con ceros.
A medida que se va pasando por la línea

El uso abusivo de la instrucción GOTO puede convertir el seguimiento del programa en una auténtica odisea.

30, se van asignando valores a los elemento de A. En definitiva, la referida tabla de seguimiento de variables adoptará el aspecto que se indica en el cuadro adjunto.

En cada ejecución del bucle interno se asigna un valor a uno de los elementos de la matriz. Los restantes elementos no se alteran y siguen conservando los valores que tenían en la pasada anterior.

Otro procedimiento de uso también generalizado, consiste en introducir señalizadores en el programa que permitan saber por dónde está avanzando la ejecución del programa; ello permitirá saber si determinada rutina se ejecuta o no. Como señalizadores pueden emplearse varias de las posibilidades del aparato. Desde señales sonoras hasta mensajes que se han de visualizar en pantalla. Entre estos últimos cabe destacar como especialmente interesantes los consistentes en visualizar en pantalla los valores de determinadas variables.

Acto seguido se muestra un ejemplo de aplicación de esta técnica. Se trata de un programa que calcula los números primos hasta un cierto valor N definido por el usuario.

```
10 INPUT N
20 FOR I=1 TO N
30 GOSUB 1000
40 NEXT I
50 END
1000 PRINT "P"
1010 FOR J=2 TO I/2
1020 IF INT(I/J)=I/J THEN GOTO 1050
1030 NEXT J
1040 PRINT I
1050 RETURN
```

La instrucción PRINT "P" de la línea 1000 se ha introducido con el único objeto de comprobar si se han realizado tantas llamadas a la rutina como se esperaba a priori. Por cada P que aparezca en la pantalla sabremos que se ha efectuado una llamada a la subrutina. De este modo, se puede controlar si el flujo de ejecución del programa es el adecuado.

Otra forma útil de determinar el correcto funcionamiento del programa consiste en detener el proceso en el momento adecuado y examinar los valores de las variables en ese preciso instante. Téngase en cuenta que, por lo general, será posible continuar la ejecución del programa mediante el empleo del comando CONT.

I	J	A(1,1)	A(1,2)	A(1,3)	A(2,1)	A(2,2)	A(2,3)	A(3,1)	A(3,2)	A(3,3)
1	1	2	0	0	0	0	0	0	0	0
1	2	2	3	0	0	0	0	0	0	0
1	3	2	3	4	0	0	0	0	0	0
2	1	2	3	4	3	0	0	0	0	0
2	2	2	3	4	3	4	0	0	0	0
2	3	2	3	4	3	4	5	0	0	0
3	1	2	3	4	3	4	5	4	0	0
3	2	2	3	4	3	4	5	4	5	0
3	3	2	3	4	3	4	5	4	5	6

Tabla de evolución de variables asociada al seguimiento de la rutina BASIC incluida en el texto.

También puede resultar de utilidad realizar ejecuciones parciales del programa dando unos valores determinados a ciertas variables. Al respecto, es conveniente recordar que una orden GOTO dirigida a un número de línea no borrará los valores previos de las variables, al contrario de lo que sucede al ejecutar un comando RUN.

LOS COMANDOS TRON/TROFF

Por último, dentro de la depuración de programas es interesante saber que muchos aparatos incorporan dos instrucciones muy útiles. Se trata de los comandos TRACE ON y TRACE OFF. El uso de estos comandos permite conocer en todo momento las líneas de programa que se están ejecutando. Su modo de funcionamiento es el siguiente: TRACE ON o TRON (dependiendo del equipo) activa un mecanismo por el que se visualiza en la pantalla los números correspondientes a las líneas que se van ejecutando. Por su parte, TRACE OFF o TROFF desactiva dicho mecanismo.

Las herramientas TRON/TROFF resultan muy útiles, especialmente en programas relativamente complicados en los que realizar el seguimiento "manual" resulta una ardua tarea. Da buenos resultados cuando el programa no funciona por quedarse estancado en un bucle sin salida. En tal caso, únicamente es necesario dejar que el programa se ejecute e ir tomando nota de los números de línea que aparecen. Hay que observar si éstos se repiten tras un determinado ciclo. Si ese ciclo es tal que impide la ejecución satisfactoria del pro-

TRON

Pone en marcha el mecanismo de trazado de la ejecución.

Formato: (número de línea) TRON

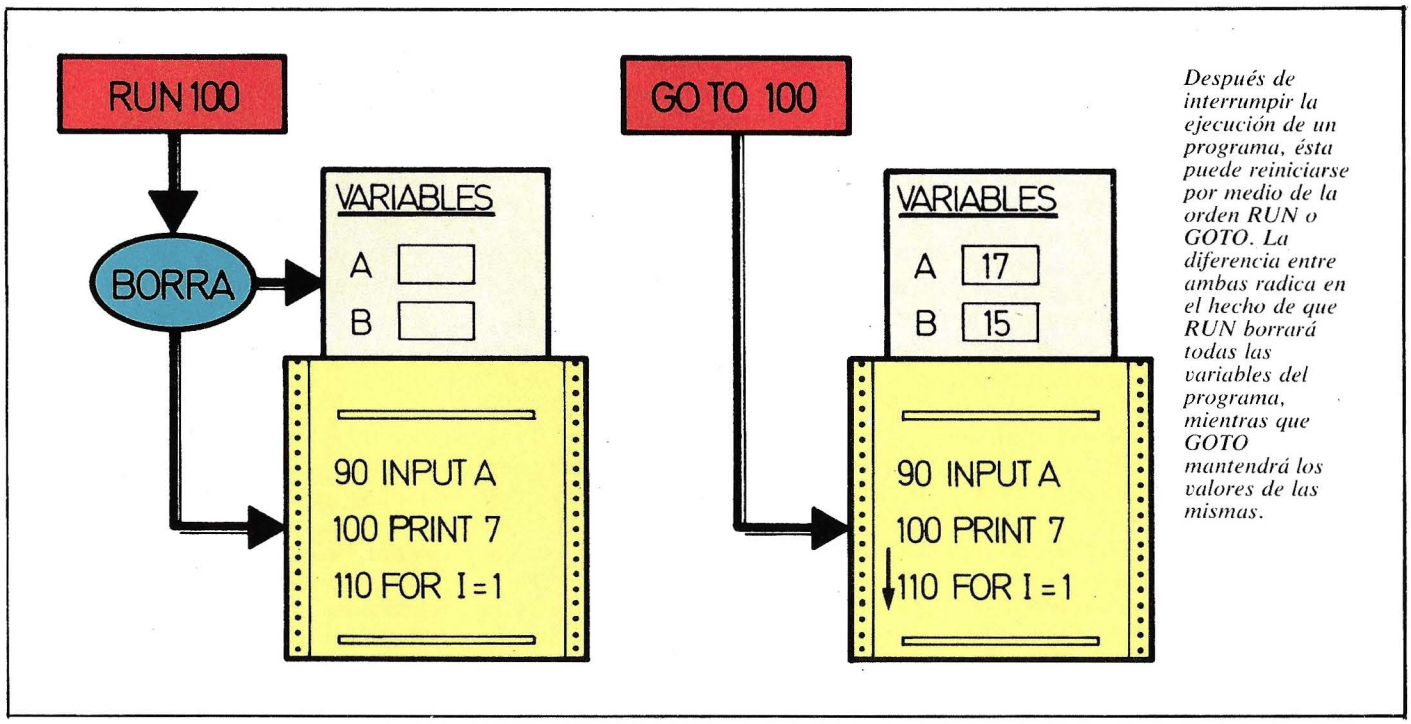
Ejemplos: 10 TRON
TRON

TROFF

Desactiva la ejecución paso a paso o trazado del programa.

Formato: (número de línea) TROFF

Ejemplos: TROFF
70 TROFF



Después de interrumpir la ejecución de un programa, ésta puede reiniciarse por medio de la orden RUN o GOTO. La diferencia entre ambas radica en el hecho de que RUN borrará todas las variables del programa, mientras que GOTO mantendrá los valores de las mismas.

grama, será ahí (en esos números de línea) donde se halla el error.

TRATAMIENTO DE ERRORES

En condiciones normales, cuando se produce un error, la máquina informa de ello mediante el correspondiente mensaje de error y, por supuesto, deteniendo la ejecución del programa.
Esta forma de actuar suele ser la más conveniente en la mayor parte de las ocasiones. Sin embargo, ello no es siempre así. De hecho, este modo de proceder es molesto y desaconsejable en muchas ocasiones. Por ejemplo, se producirá un error al introducir un carácter alfabético tras la ejecución de un comando INPUT correspondiente a una variable numérica. Este error producirá la detención del programa, siendo necesario reiniciarlo si se desea continuar.
Existe una primera solución al problema. Esta consiste en realizar la introducción del dato como variable de cadena para, más tarde, operar la necesaria conversión.
Existe otra solución, que es precisamente la que se pretende estudiar en este capítulo; solución que tiene como base el tratamiento del error por parte del programa. Para llevarlo a cabo es necesario indicar al ordenador que, en caso de producirse di cho error, se ejecute una determinada rutina. Rutina ésta que ha de aportar las instrucciones necesarias para el correcto tratamiento del error detectado. Esa rutina puede muy bien ser codificada en BASIC por el propio usuario. El comando necesario para indicar esta situación al ordenador es ON ERROR GOTO, cuyo formato es el que se muestra a continuación:

(Número de línea) ON ERROR GOTO <número de línea>

Donde <número de línea> corresponde al número de línea donde empieza la rutina que se ha de ejecutar en el caso de que se produzca el error en cuestión.
A partir de este momento es necesario conocer algunos detalles que intervienen en la rutina de tratamiento de error. Para empezar, existen dos variables: ERR y ERL que contienen información acerca del error que se ha producido. La primera de ellas es ERR (ERror Report) y contiene el código que identifica el error que ha tenido lugar. Por su parte ERL (ERror Line) contiene un número que corresponde al número de línea en el que se ha detectado dicho error. Con esta información ya es posible conocer unos parámetros bastante interesantes y que pueden ser empleados en la rutina.
Los códigos de error vienen asignados por el propio ordenador con el que se trabaje; aunque bien es cierto que también existen otros códigos no asignados que pueden ser empleados para definir errores de otra índole.
En cuanto al repertorio de errores que aporta el aparato y a sus códigos correspondientes, lo más efectivo es consultar

ON ERROR GOTO

Hace que, al producirse un error, se ejecute la rutina situada a partir de la línea que se indica.

Formato: ON ERROR GOTO <número de línea>

Ejemplos: 10 ON ERROR GOTO 100
50 ON ERROR GOTO 2500

el manual del propio equipo, ya que la equivalencia de esos códigos es específica de cada ordenador.

Las rutinas de tratamiento de error han de incluir, al final, una instrucción que continúe la ejecución del programa. Estas rutinas actúan de forma muy parecida a las subrutinas invocadas con el comando GOSUB. Al igual que aquellas, necesitan de una instrucción que, como RETURN, de-

RESUME

Reanuda la ejecución de la zona principal del programa en el punto indicado en su argumento.

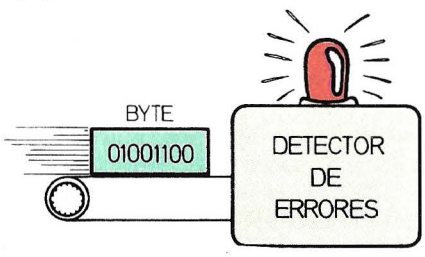
Formato: (número de línea) RESUME <opción>

Ejemplos: 20 RESUME
70 RESUME 50

Códigos detectores de error

Paridad par y paridad impar

En la transmisión de información de un emisor a un receptor pueden producirse errores. Esto ocurre en todo tipo de comunicaciones. En una conversación entre dos personas es posible que el ruido ambiente haga difícil el entendimiento entre éstas. Al comunicarse utilizando otro medio de transmisión también se dan interferencias que alteran la información transmitida. El ejemplo más palpable es el de las conferencias telefónicas.
En el interior del ordenador se produce un continuo trasiego de información. Esta se va desplazando del teclado a la unidad central, de la unidad central a la memoria, de la unidad central a la pantalla, de la memoria a la unidad central, etc. En este maremagnum de tráfico no es extraño que se produzcan alteraciones en la información.
Como es sabido, la información es tratada por el ordenador en forma de impulsos eléctricos. Dichos impulsos se transmiten a través de conductores y cuanto mayor es la longitud del conductor, mayor resistencia opone éste al paso de la corriente. Esto hará que algunos impulsos queden anulados, produciéndose errores en la transmisión.
Para paliar de algún modo esos errores de transmisión se hace uso de los sistemas de detección de error. Para ello se recurre al envío de información redundante que permite la confrontación de los datos. Si una información no coincide con la otra es que se ha producido un error. Los diferentes métodos detectan el error producido en la transmisión de un byte. La información adicional se codifica en algunos bits que se añaden a los ocho de cada byte.



El sistema más sencillo y más empleado es el método de la paridad. En él se comprueba si el número de unos transmitidos es par o impar. La paridad par consiste en añadir a la codificación de cada uno de los caracteres un bit; bit que se elige de forma que el número total de unos sea par. Con este convenio es posible detectar si se ha producido un error en uno de los bits. Veamos cómo.
Suponga que se desea transmitir la siguiente información:

01010101
11100000
11111110

A estos bytes es preciso añadirles un noveno bit; bit que hará que el número de unos sea par. Los nuevos datos serían los siguientes:

01010101 0
11100000 1
11111110 1

Tal es la información que se envía. En el extremo receptor se verifica la paridad de los datos; si el número de unos de alguno de los bytes es impar es que existe un error; a su vez, si no hay error se elimina ese noveno bit.
Con los datos anteriores se podría recibir la secuencia que se muestra a continuación.

00010101 0
11100000 1
11111110 1

Al contar el número de unos del primer byte de datos se ve que es impar. Esto permite asegurar que se ha producido un error en la transmisión de dicho dato. Este método de detección sólo es seguro cuando se produce un único error por byte. Si se

Los errores pueden convertir a una transmisión en inteligible. Para evitar esta situación se recurre al empleo de los bits de paridad, cuya misión es la de permitir la detección de posibles errores en la información transferida.

producen dos errores (o un número par de errores) éstos no serían detectados.
Veamos lo que ocurre cuando se introduce un segundo error en el dato anterior.

10010101 0

El dato no coincide con el enviado por el emisor. Sin embargo, el número de unos es par. Esto indica al receptor que no existe error, cuando sabemos que ello no es cierto.
Afortunadamente, en los sistemas modernos es cada vez más difícil que se produzca un error de transmisión. Si la probabilidad de que se introduzca un error es bastante pequeña, la probabilidad de que coincidan dos en el mismo byte es mucho más remota. Por ello este sistema es suficientemente eficaz.
El método de paridad impar es semejante al comentado. La única diferencia estriba en la elección del noveno bit. En el caso de paridad impar, el mencionado bit ha de hacer que el número de unos de cada byte de datos sea impar.

vuelva el control a la zona principal del programa. Esta instrucción es RESUME y su formato es el siguiente:

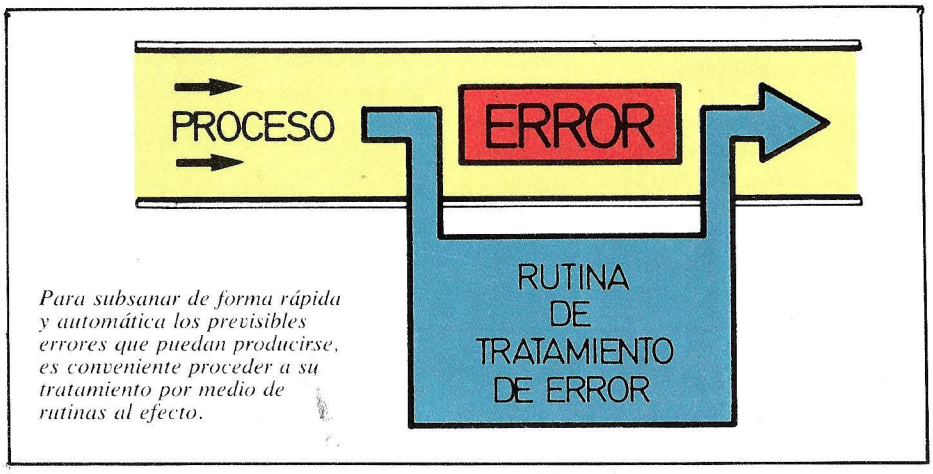
(Número de línea) RESUME <opción>

En el que la zona de <opción> hace referencia a una de las posibles alternativas que a continuación se comentan.

● RESUME sin argumento o RESUME 0
Una vez procesada la rutina de tratamiento, si el ordenador se encuentra con esta instrucción continuará la ejecución del programa principal en la misma línea en la que se produjo el error.

● RESUME NEXT
Esta opción hace que, tras el tratamiento del error, la ejecución del programa continúe en la línea siguiente a aquella en la que se produjo dicho error.

● RESUME <número de línea>
Cuando se incluye este argumento, el programa continúa ejecutándose en la línea correspondiente al número que se especifica.
De esta forma, después de tratar el error se puede volver al programa principal, prosiguiendo su ejecución desde el punto deseado.
Cuando se emplea una de estas instrucciones sin que previamente se haya producido un error, se producirá a su vez otro error: RESUME ERROR. Por lo tanto, es preciso separar la línea que contenga este comando del resto del programa. Ese error es similar al que se produce al intentar ejecutar un comando RETURN sin haber efectuado previamente una llamada a subrutina (GOSUB).
Y no sólo se pueden tratar los errores contemplados por el aparato. También el propio usuario puede crear errores que atiendan situaciones no previstas por el fabricante. Esto último se consigue utilizando los códigos de error que no estén asignados previamente, o tratando de forma distinta los errores menos habituales.



Para subsanar de forma rápida y automática los previsibles errores que puedan producirse, es conveniente proceder a su tratamiento por medio de rutinas al efecto.

ERRORES "A VOLUNTAD" DEL USUARIO

Para producir un error se puede emplear el comando ERROR, cuyo formato es el que se indica.

(Número de línea) ERROR <código de error>

Donde <código de error> es un dato numérico correspondiente al código que identifica al error deseado.
Con este comando se puede generar cualquiera de los errores contemplados en el aparato, así como otros que el usuario desee emplear. Estos últimos han de ser los correspondientes a los códigos que se encuentran sin asignar.
A continuación, se incluye un ejemplo relativo al uso de las instrucciones para el tratamiento de errores. En este caso se aplica al conocido juego de adivinar un número.

```
10 REM ADIVINA UN NUMERO
20 ON ERROR GOTO 80
30 LET A=1+INT(100*RND)
40 INPUT "NUMERO";B
50 IF A>B THEN ERROR 80 ELSE IF A<B
   THEN ERROR 81
60 PRINT "ACERTASTE"
70 END
80 IF ERR=81 THEN PRINT "DEMASIADO
   ALTO" ELSE PRINT "DEMASIADO BAJO"
90 RESUME 40
100 END
```

En la línea 20 se comunica al ordenador que, en el caso de producirse un error, debe desviar la ejecución a una rutina de tratamiento situada a partir de la línea 80.
A continuación, en la línea 30 se genera el número que el usuario debe acertar. Este número estará comprendido entre uno y cien.
En la línea 40 el aparato pide al operador que introduzca el número que constituye su intento; mientras que en la línea 50 se comprueba si el número introducido coincide con el que calculó la máquina. En ese preciso momento se puede generar un error, que será el 80 si el número introducido es mayor que el calculado por la máquina, o el 81 si es menor.
Si se produce uno de ambos errores, el ordenador efectuará un salto en la ejecución al número de línea indicado en la instrucción ON ERROR GOTO. El comando situado en la línea 60 es el que se ejecutará cuando se acierte el número; su misión es presentar en la pantalla un mensaje de felicitación. Por último, la instrucción de la línea 70 es la que finaliza la rutina principal.

ERROR

Produce el error identificado por el número de código indicado en su argumento.

Formato: (número de línea) ERROR <código de error>

Ejemplos: 20 ERROR 80
70 ERROR 2

TABLA DE CONVERSION

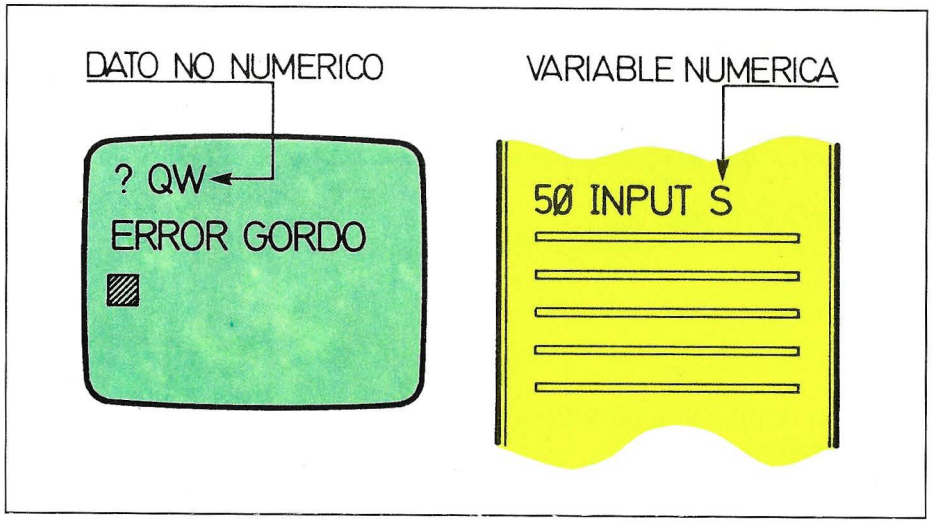
ORDENADOR	TRAZADO	TRATAMIENTO DE ERRORES			
	TRON/TROFF	ON ERROR GOTO n	RESUME	ERROR n	ERR y ERL
APPLE II (APPLESOFT)	TRACE/NOTRACE	ON ERR GOTO	RESUME	—	—
APRICOT (M-BASIC)	TRON/TROFF	ON ERROR GOTO n	RESUME	ERROR n	ERR y ERL
ATARI	—	TRAP	—	—	—
CBM 64	—	—	—	—	—
DRAGON	TRON/TROFF	—	—	—	—
EQUIPOS MSX	TRON/TROFF	ON ERROR GOTO n	RESUME	ERROR n	ERR y ERL
HP-150	TRON/TROFF	ON ERROR GOTO n	RESUME	ERROR n	ERR y ERL
IBM PC	TRON/TROFF	ON ERROR GOTO n	RESUME	ERROR n	ERR y ERL
MPF	TRACE/NOTRACE	ONERR GOTO	RESUME	—	—
NCR DM-V (MS-BASIC)	TRON/TROFF	ON ERROR GOTO n	RESUME	ERROR n	ERR y ERL
NEW BRAIN	—	ON ERROR GOTO n	RESUME	ERROR n	ERRNO/ERRLIN
ORIC	TRON/TROFF	—	—	—	—
SHARP MZ-700 (MZ-BASIC)	—	ON ERROR GOTO n	RESUME	—	ERR y ERL
SINCLAIR QL	—	—	—	—	—
SPECTRAVIDEO	TRON/TROFF	ON ERROR GOTO n	RESUME	ERROR n	ERR y ERL
ZX-SPECTRUM	—	—	—	—	—

Entre las líneas 80 y 100 está situada la rutina de tratamiento de errores. En ella se encuentra la línea 80, que genera un mensaje si el error que se está tratando es el 80, y otro si corresponde al 81. Por último, la línea 90 es la que pone fin a la rutina y produce el retorno a la zona principal. El retorno se efectúa, concretamente, a la línea 40.

Si el ordenador utilizado no dispone de la partícula ELSE dentro del comando IF, se puede sustituir la línea 50 por las dos siguientes:

```
50 IF A>B THEN ERROR 80
55 IF A<B THEN ERROR 81
```

De la misma forma, la línea 80 se verá sustituida por las que se indican a continuación:



El propio intérprete de lenguaje BASIC hace uso de un sistema de tratamiento de errores. Esto se pone de manifiesto, por ejemplo, al intentar asignar una cadena de caracteres a una variable numérica.

```
80 IF ERR=81 THEN PRINT "DEMASIADO
   ALTO"
85 IF ERR=80 THEN PRINT "DEMASIADO
   BAJO"
```

EL TRATAMIENTO DE ERRORES EN LA PRACTICA

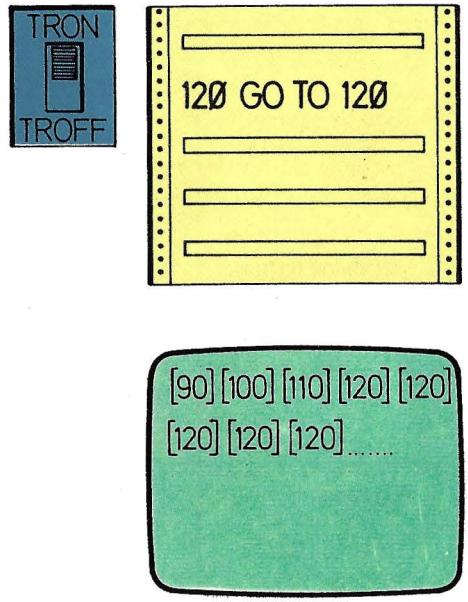
Como ya se ha indicado, los errores de usuario pueden ser empleados para producir y tratar errores no contemplados en el repertorio propio del aparato.
Por ejemplo, si como se comentaba más arriba, se desea que no se detenga la ejecución del programa cuando se produzca un error, se puede emplear la técnica que revela el siguiente ejemplo:

```
90 ON ERROR GOTO 100
100 INPUT A
110 PRINT 100/A
```

En este caso, al producirse un error se saltará a la línea 100. El error puede originarse al intentar asignar el valor cero a la variable numérica A. Al tener lugar ese error se salta a la línea 100 volviéndose a recoger un nuevo valor del teclado. Con ello se consigue la reiterada ejecución del comando INPUT. El programa sólo continuará si el dato introducido no conduce a error.
Realmente, en el ejemplo propuesto no existe un tratamiento del error propiamente dicho. Para efectuar dicho tratamiento habría que utilizar una rutina que podría ser semejante a la incluida en las líneas 1000 y 1010 del siguiente ejemplo:

```
90 ON ERROR GOTO 1000
100 INPUT A
110 PRINT 100/A
.....
.....
.....
900 END
1000 PRINT "DATO NO VALIDO,INTENTELO
     DE NUEVO"
1010 RESUME 100
```

Como se observa, el tratamiento del error consiste en mostrar un mensaje y repetir la introducción del dato. Hay que poner



Las instrucciones TRON y TROFF activan y desactivan el trazado del programa, o lo que es lo mismo, su ejecución paso a paso. Esta posibilidad resulta muy útil para seguir el flujo de ejecución a través de las diversas zonas del programa. Ello permitirá detectar fácilmente la existencia de bucles infinitos.

La instrucción RESUME indica al ordenador en qué punto del programa principal ha de continuar la ejecución tras procesar la rutina para el tratamiento del error.

cuidado en la elección del punto de retorno. Si la línea 1010 incluyera un comando RESUME sin argumento, se reanudaría la ejecución en la línea 110. Ello provocaría un retorno a la línea que da origen al error, sin subsanar el mismo.
En el caso de que se pueda conocer a priori el tipo de error originable, éste puede ser corregido en la rutina de tratamiento. En el ejemplo se sabe que el error se producirá cuando A valga cero; por lo tanto, la rutina de error puede utilizarse para cambiar el valor de dicha variable. Esto es precisamente lo que se hace en el siguiente ejemplo:

```
90 ON ERROR GOTO 1000
100 INPUT A
110 PRINT 100/A
.....
.....
.....
900 END
1000 LET A=1
1010 RESUME
```

Ahora, de producirse el error, será el mismo programa quien lo solucione, asignando un valor no nulo (en este caso 1) a la variable A.
Si se prevén distintos tipos de errores, se pueden separar sus rutinas de tratamiento de varias formas.
Si los diferentes errores surgen en zonas separadas del programa, lo más adecuado es intercalar comandos ON ERROR GOTO con distintos argumentos. Así, al principio de una determinada zona de instrucciones, se colocará una orden del tipo ON ERROR GOTO que salte a la rutina que atienda el error propio de esa parte del programa. Ello supone añadir tantas rutinas de tratamiento como errores se prevean.
En el caso de que el error pueda surgir en cualquier parte del programa y pueda ser de distinto tipo, lo anterior será inviable. Esto ocurrirá si en una misma línea se pueden producir errores distintos. La solución en tal caso es utilizar una rutina de tratamiento que "distinga" el tipo de error. Dicha distinción se puede realizar muy fácilmente utilizando la variable ERR para bifurcar la ejecución.
También es posible identificar de una manera más detallada el error. Esto último se consigue con el empleo conjunto de ERR y ERL. De esta forma se puede realizar un tratamiento individualizado de cada error que se produzca.

Forth (3)

Variables y manejo de pantalla

Al igual que ocurre con cualquier otro lenguaje informático, también en el FORTH es posible definir variables. Como ya conocerá todo aquel que se encuentre familiarizado con los temas de programación, una variable corresponde sencillamente a una posición de memoria cuyo valor se puede alterar a voluntad. El mecanismo para crear una variable FORTH pasa por utilizar la palabra VARIABLE. Para proceder a su definición, hay que empezar situando el valor inicial de la variable en la pila. A continuación, hay que indicar al ordenador que deseamos crear una variable, mediante el empleo de la palabra FORTH VARIABLE y, por último, hay que especificar el nombre que deseamos emplear como distintivo de dicha variable. Veamos algún ejemplo de definición de variables:

```
23 VARIABLE EDAD <CR>
23 VARIABLE EDAD OK

45 VARIABLE BASE <CR>
45 VARIABLE BASE OK
■
```

El siguiente paso dentro del proceso de trabajo con una variable, se concreta en la forma de trasladar a la pila el valor de la misma. Para ello, es necesario utilizar la palabra FORTH "@", seguida por el nombre identificador de la variable. Por ejemplo:

```
EDAD @  . <CR>
EDAD @  . 23
```

Una vez que el valor de la variable se encuentra depositado en la pila, es posible ya efectuar operaciones con el mismo.
La actuación de la palabra @ se resume en lo siguiente: busca en la pila una dirección de memoria, y recoge el contenido de dicha posición de memoria y de la siguiente, depositándolo en la pila. Cabe deducir, por lo tanto, que hace el uso de una palabra como identificador de una variable, equivale a asociar a dicha palabra un número representativo de una dirección de memoria. Veamos un ejemplo:

```
EDAD  . <CR>
EDAD  . 15452 OK
```

Como se observa, tras indicar el nombre de una variable, el ordenador muestra un número que se encuentra en la pila; número que coincide realmente con una dirección de memoria. Veamos qué sucede si colocamos directamente en la pila la referida dirección de memoria y empleamos la palabra @ .

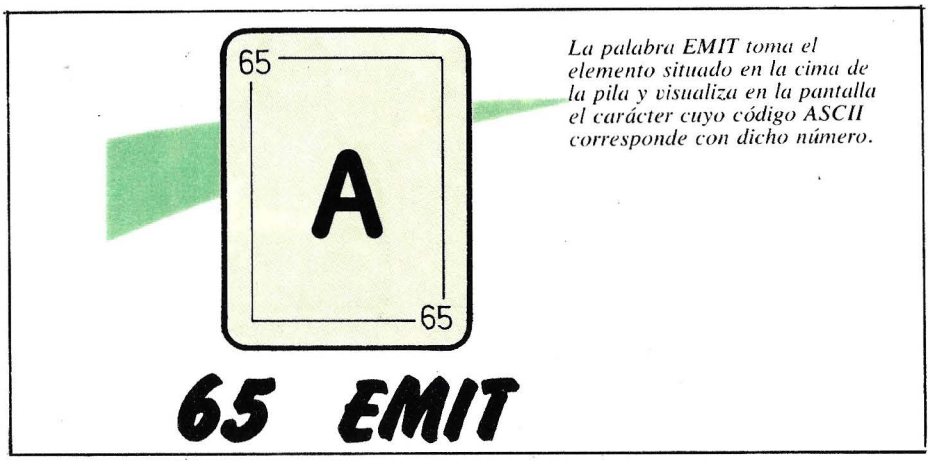
```
15452 @   . <CR>
15452 @   . 23 OK
```

En efecto, la pantalla muestra el valor almacenado en dicha posición. Ello significa que puede utilizarse la palabra @ para examinar el contenido de dos posiciones de memoria consecutivas; los que desee el programador.
Para cambiar el valor de una variable, hay que recurrir a la palabra "!", palabra que, a su vez, también es posible utilizarla para alterar el contenido de una dirección de memoria. Por ejemplo:

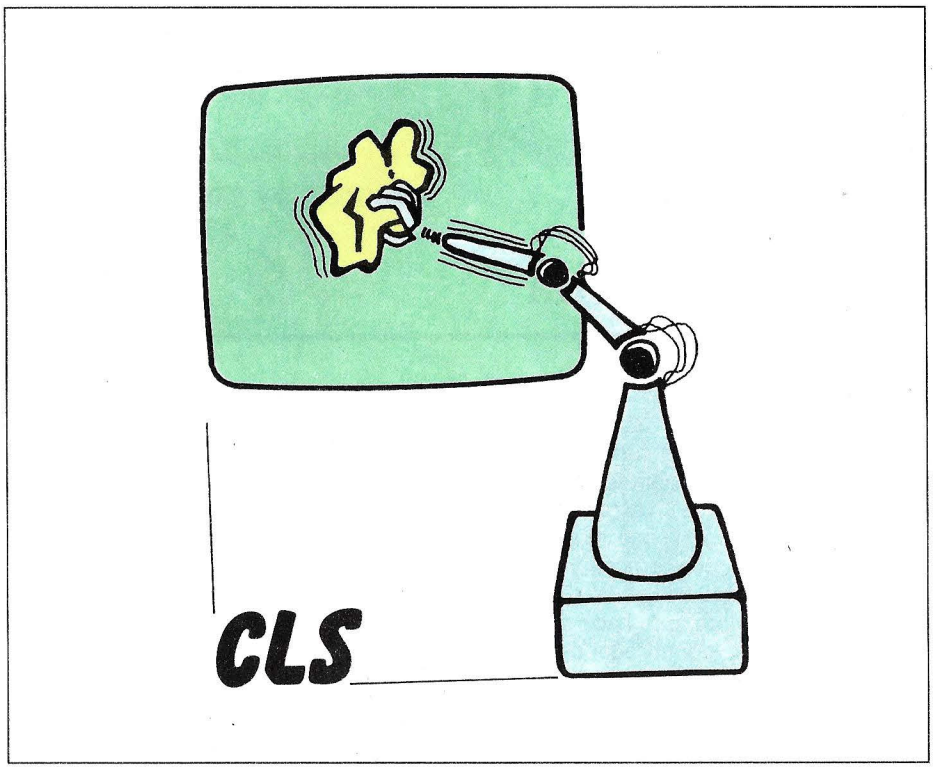
```
29 EDAD ! EDAD @  . <CR>
29 EDAD ! EDAD @  . 29 OK

29 15452 @ EDAD @  . <CR>
29 15452 @ EDAD @  . 29 OK
```

Otra posibilidad es la de modificar el valor de una constante, redefiniéndola mediante el empleo de la palabra asociada a la misma, aunque con la desventaja de que dicha palabra no podrá ser utilizada dentro de la definición de otra nueva palabra. Por ejemplo:

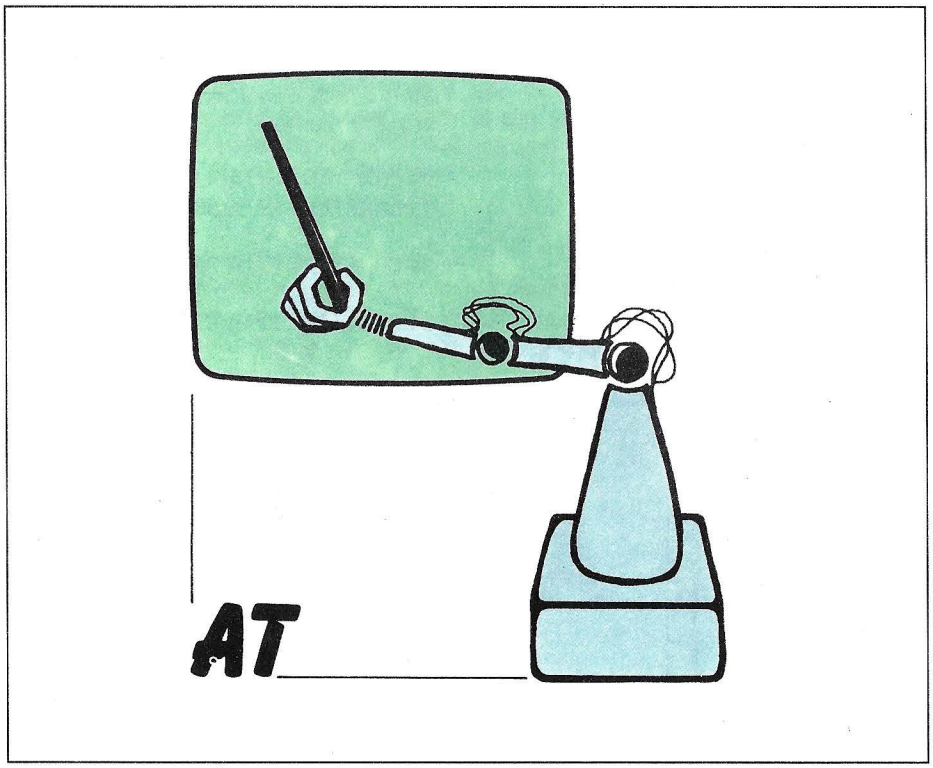
```
23 CONSTANT BALA <CR>
23 CONSTANT BALA OK

34 CONSTANT BALA REDEFINE BALA
   BALA  . <CR>
34 CONSTANT BALA REDEFINE BALA
   BALA  . 34 OK
```



La palabra EMIT toma el elemento situado en la cima de la pila y visualiza en la pantalla el carácter cuyo código ASCII corresponde con dicho número.



CLS es el comando FORTH especializado en el borrado del contenido de la pantalla.



La orden AT tiene encomendada la tarea de precisar el punto de la pantalla en el que se realizará la próxima visualización de datos.

IMPRIMIENDO EN PANTALLA

El lenguaje FORTH ofrece varios métodos para visualizar en pantalla la información puesta en juego. La forma más frecuente y sencilla consiste sencillamente, en el uso de la palabra FORTH " .". Esta palabra (.) presenta en pantalla el número que se encuentre, en ese preciso instante, en la cima de la pila. Por supuesto, si la pila está vacía la máquina responderá con un mensaje de error.

Para imprimir mensajes en pantalla podemos recurrir a la palabra FORTH ." . Esta hará que se visualice en la pantalla el mensaje que se escriba a continuación, hasta que el ordenador encuentre otras comillas.

En determinadas versiones de FORTH, la palabra que nos ocupa (.") sólo se puede utilizar dentro de la definición de otra palabra FORTH. Suponiendo que la versión con la que se trabaja admite su ejecución en modo inmediato, la actuación de dicha palabra es la que ilustran los siguientes ejemplos:

```
." HOLA" <CR>
." HOLA" HOLA OK
." ESTO ES UN MENSAJE" <CR>
." ESTO ES UN MENSAJE" ESTO ES UN
   MENSAJE OK
```

No sólo es posible visualizar en pantalla un mensaje entrecomillado, sino que también puede escribirse en la misma un carácter dado por su código ASCII. Para ello, el lenguaje FORTH cuenta con la palabra clave EMIT. Al ejecutarla, el ordenador escribirá en la pantalla el carácter cuyo código ASCII se encuentra en la cima de la pila. Por ejemplo:

```
65 EMIT <CR>
65 EMIT A OK

48 EMIT <CR>
48 EMIT 0 OK
```

EMIT permitirá, pues, visualizar en pantalla los caracteres que conforman el reper-

torio del ordenador en uso. Dado que cada equipo tiene un juego de caracteres distinto, en mayor o menor grado, al de los demás, no es posible dar una tabla de códigos generalizable. No obstante, en cualquier caso, las cifras decimales tendrán el código comprendido entre los valores 48 y 57, y las letras desde el 65 en adelante (generalmente para las mayúsculas).

Hasta ahora se ha descrito cómo obtener una visualización en pantalla, aunque sin realizar ningún tratamiento específico de esta. El FORTH brinda algunas palabras especializadas en la operación con la pantalla; veamos a continuación las más importantes:

CLS Permite borrar la pantalla, eliminando ella todo lo que anteriormente se encontraba escrito en la misma.

```
3 . CLS 5 . <CR>
```

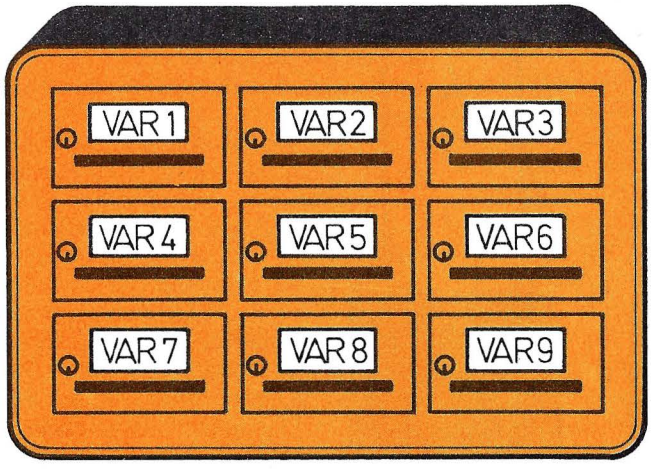
```
3 . 3
■
4 . 4 OK
■
```

AT Sitúa el cursor de impresión en el punto especificado por los dos números más próximos a la cima de la pila. El más cercano a la cima, indica la columna, y el siguiente la fila en la que ha de realizarse la impresión. Por ejemplo:

```
4 1 8 AT . <CR>
```

```
4 1 8 AT
        . 4 OK
■
```

La presencia del FORTH en el ámbito de los microordenadores es notable. Hace algún tiempo, incluso podía adquirirse un económico equipo doméstico —denominado Júpiter Ace— cuyo lenguaje residente era precisamente el FORTH, en lugar del tradicional BASIC.



Cabe imaginar a la memoria como un conjunto de buzones, cada uno de ellos constituido por dos celdas de memoria (bytes), en donde es posible almacenar datos. Mediante la definición de constantes y variables asociamos un identificador a cada uno de los buzones, lo cual permitirá un acceso más cómodo a los mismos.

TABLA DE ORDENES FORTH

PALABRA	DESCRIPCION	TIPO
VARIABLE	Define una variable.	Palabra de definición.
@	Busca el contenido de un elemento en memoria.	Palabra de definición.
!	Sitúa un elemento en memoria.	Palabra de definición.
REDEFINE	Cambia el valor de una costante.	Palabra de definición.
.	Imprime en la pantalla el valor de elemento situado en la cima de la pila.	Trabajo con la pantalla.
."	Imprime en pantalla un mensaje.	Trabajo con la pantalla.
EMIT	Presenta en pantalla un carácter dado por su correspondiente carácter ASCII.	Trabajo con la pantalla.
CLS	Borra la pantalla.	Trabajo con la pantalla.
AT	Sitúa el cursor en un punto dado de la pantalla.	Trabajo con la pantalla.
HOME	Coloca el cursor en la esquina superior izquierda de la pantalla.	Trabajo con la pantalla.
SPACE	Escribe en la pantalla un espacio en blanco.	Trabajo con la pantalla.
SPACES	Escribe en la pantalla los espacios en blanco que indique el número situado en la cima de la pila.	Trabajo con la pantalla.
AND	Realiza la operación AND.	Operadores lógicos.
OR	Realiza la operación OR.	Operadores lógicos.
XOR	Realiza la operación OR-exclusiva.	Operadores lógicos.

OPERADORES LOGICOS

Tal y como establece la teoría de cálculo en el sistema binario, las operaciones lógicas se realizan bit a bit. El FORTH cuenta, habitualmente, con tres operadores lógicos fundamentales: AND, OR y XOR.

AND: Corresponde a la operación de producto lógiço (AND).

OR: Corresponde a la operación de suma lógica (OR).

XOR: Corresponde a la operación de suma lógica exclusiva (OR-exclusiva).

Veamos algún ejemplo al respecto:

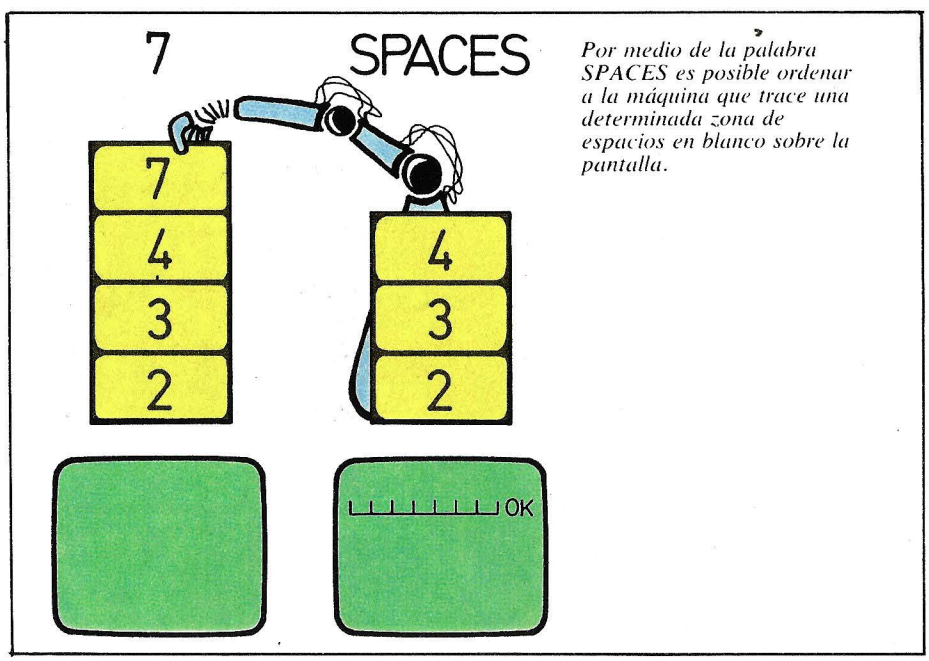
```
2 7 AND . <CR>
2 7 AND . 2 OK

2 7 OR . <CR>
2 7 OR . 7 OK

2 7 XOR . <CR>
2 7 XOR . 5 OK
```

HOME Traslada el cursor de impresión a la esquina superior izquierda de la pantalla.

SPACE Imprime en pantalla un espacio en blanco.

SPACES Visualiza una zona de espacios en blanco coincidente con el número situado en la cima de la pila:

```
9 SPACES <CR>
9 SPACES          OK
■
```

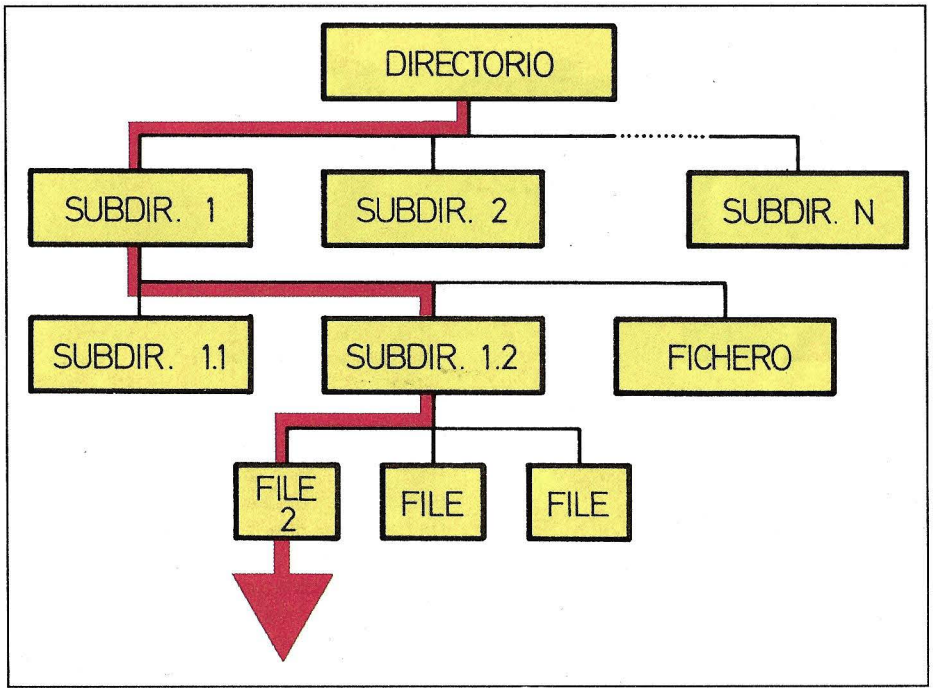


Por medio de la palabra SPACES es posible ordenar a la máquina que trace una determinada zona de espacios en blanco sobre la pantalla.

Apple ProDOS (y 3)

Comandos de fichero

Los comandos de Fichero son aquellos que permiten al sistema operativo actuar sobre estas unidades de información y realizar sobre ellas tareas de mantenimiento y modificación. Se tiene acceso a los mismos desde el menú principal seleccionando la opción FILER. Esta opción da entrada a un nuevo menú, cuya opción FILE COMMANDS da paso, finalmente, al conjunto de comandos que permitirán realizar las tareas anteriormente indicadas. Antes de profundizar en los diferentes comandos y sus funciones, señalaremos algunas características de la estructura de almacenamiento de datos, tal y como la gestiona el sistema operativo ProDOS.



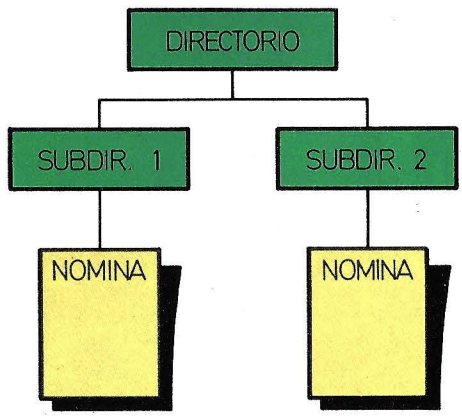
La estructura jerárquica que adoptan los archivos bajo el control del ProDOS, obliga al usuario a precisar el recorrido a seguir, a través del correspondiente directorio y subdirectorios, para definir la localización de un fichero.

DIRECTORIOS Y SUBDIRECTORIOS

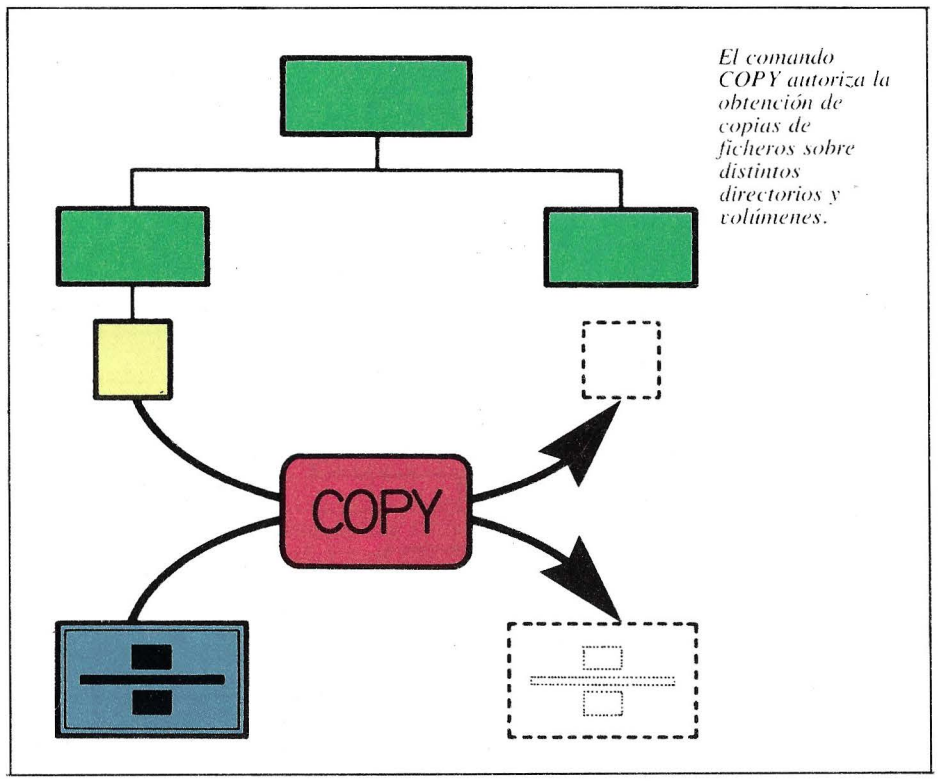
La estructura de almacenamiento de información en este sistema operativo es de tipo arborescente: parte de un elemento raíz y a partir del mismo se va dividiendo en una serie de ramas, las cuales, a su vez, se van subdividiendo sucesivamente hasta llegar a un elemento final. En esta estructura, el elemento raíz es el denominado directorio; los nudos en los que se va ramificando, los subdirectorios; y los elementos finales, donde ya no se producen divisiones, son precisamente los ficheros o elemento de almacenamiento de información.

Cuando un disco o disquete se formatea bajo el control del sistema operativo ProDOS, adquiere un nombre y un directorio. El directorio es como tal un fichero, aunque un fichero de una categoría muy especial, ya que mantiene un seguimiento de todos los ficheros contenidos en el disco o disquete.

Si no se diera entrada a la estructura arborescente y fuéramos creando ficheros con plena independencia, al cabo de un tiempo el disquete estaría lleno de un gran número de ellos no relacionados entre sí. Aparece, en este momento, la figura del subdirectorio, como un método para organizar lógicamente los ficheros, de forma que todos aquellos que se encuentren relacionados puedan almacenarse bajo un epígrafe común. El subdirectorio es, como su propio nombre indica, un directorio de nivel inferior, con una función idéntica, pero agrupando tan sólo aquella información que depende de él. Como elemento de clasificación lógica



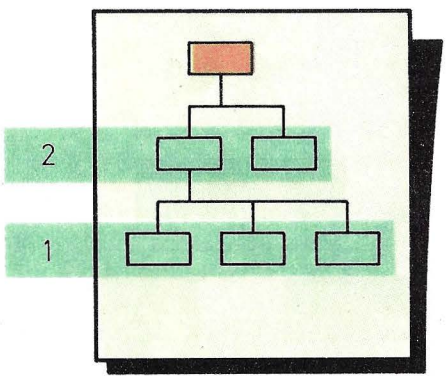
La estructura arborescente hace posible que en un soporte de memoria masiva residan ficheros con el mismo nombre, aunque en directorios distintos.



El comando COPY autoriza la obtención de copias de ficheros sobre distintos directorios y volúmenes.

es repetible, de forma que un subdirectorio puede contener a su vez uno o varios subdirectorios.

Una forma gráfica de ilustrar dicha estructura, la aporta el sistema de archivos de una hipotética empresa. En ella, a cada cliente se le asigna un archivador, archivador que está dividido en dos zonas: una



La aplicación del comando de borrado DELETE debe respetar la estructura jerárquica. Ello significa que antes de eliminar un subdirectorio, es preciso borrar los archivos asociados al mismo.

que contiene los pedidos y otra las facturas correspondientes. El área destinada a las facturas está dividida a su vez en otras dos partes: la que contiene las facturas pagadas al contado y la que acoge a las facturas de pago aplazado. Si se trata de reproducir, por medio de directorios y subdirectorios, la estrutura que acabamos de definir, el directorio sería el equivalente al archivador; mientras que los subdirectorios coincidirían con cada una de las divisiones sucesivas: pedidos, facturas y, dentro de esta última clasificación, con los epígrafes de facturas pagadas al contado y facturas de pago aplazado. Los elementos finales, los ficheros, serían los pedidos y las facturas en sus diferentes formas.

LOCALIZACION DE FICHEROS

Una vez establecida esta estructura, para acceder a un fichero determinado el sistema operativo ProDOS exige indicar los diferentes pasos a seguir dentro de la cadena descendente, especificando el nombre de directorio, los nombres de los diversos subdirectorios intermedios y el nombre del fichero; cada uno de estos nombres se separa por medio de una barra inclinada (/). Hay que tener en cuenta, a su vez, que la longitud del conjunto, en caracteres, no puede ser superior a 64, y que el nombre del fichero no puede exceder de 15.

Esta estructura permite que varios ficheros con el mismo nombre se encuentren almacenados en un mismo disquete o disco rígido, siempre y cuando se encuentren en directorios o subdirectorios distintos.

El sistema operativo sigue la trayectoria de directorios y subdirectorios especificada al designar el fichero, moviéndose a través de la estructura de almacenamiento. Mientras que el "trazado" a seguir sea distinto, no importa que el elemento final, el fichero, tenga igual nombre que otro al que se accede por otro camino.

Como se ha visto, este tipo de estructura exige suministrar al sistema operativo el conjunto de nombres de directorio y subdirectorios que conducen al fichero requerido. Para aliviar este esfuerzo y agilizar la gestión de acceso, el ProDOS permite definir lo que denomina un prefijo.

Para explicar su función es conveniente regresar al anterior ejemplo de la empresa. Si deseáramos efectuar un control de las facturas aplazadas aún no cobradas, consultándolas continuamente, cada vez que tuviéramos que acceder a una de ellas sería preciso especificar el directorio (archivador de clientes), subdirectorio (facturas), subdirectorio (facturas de pago aplazado) y, finalmente, nombre del fichero (factura X). Podemos imaginar fácilmente que este proceso resultará largo y tedioso. Para hacer más cómoda la tarea podemos recurrir a los "prefijos" antes señalados. En este caso, el prefijo designaría a la zona de esta estructura constituida por los nombres de directorio y subdirectorios, de tal forma que cuando quisiéramos acceder a un fichero determinado tan sólo habría que especificar el prefijo y el nombre del fichero. Sin lugar a dudas, el trabajo de definición se reduce enormemente. El prefijo, como vemos, actúa en forma de referencia mnemotécnica.

Otra forma de agrupar los ficheros es por medio de las denominadas referencias ambiguas o "wildcards". En el sistema operativo ProDOS es posible escoger entre dos tipos de referencias ambiguas, aunque el hecho de elegir una u otra significa una modificación del modo de trabajo. En esencia, una referencia ambigua es una forma de indicar al sistema operativo que la orden formulada debe hacerse ex-

tensiva a aquellos ficheros que comparten una determinada característica en sus nombres; por ejemplo, que empiecen o terminen por una cierta letra, o que contengan un determinado grupo de letras. Partiendo de esta base, la referencia ambigua "=" señala, además, al sistema operativo que la acción a realizar sobre el fichero o ficheros no requiere confirmación por parte del usuario. Por el contrario, si la referencia ambigua utilizada coincide con el símbolo "?", antes de efectuar cualquier acción, el sistema operativo solicita confirmación al usuario. En definitiva, este doble matiz de las referencias ambiguas permite manipular bloques de ficheros, garantizando que la acción a llevar a cabo es la correcta y que no se modifica o destruye información valiosa.

COMANDOS DE FICHERO DEL ProDOS

En el ProDOS, la manipulación a nivel de ficheros se realiza bajo el control de un menú específico —FILE COMMANDS—, asociado a la opción FILER del menú principal.

A continuación se describen una serie de comandos del sistema operativo ProDOS relativos a la gestión a nivel de fichero. Cabe señalar que todos ellos están disponibles dentro del menú de "Comandos de fichero", de ahí que consideraremos que el usuario se encuentra ya frente a dicho menú que permite seleccionarlos directamente, no en el menú principal del sistema operativo ni en el menú de la opción FILER.

- LIST DIRECTORY

Tiene una función análoga a la del índice de un libro, ya que especifica qué elementos de información (ficheros, subdirectorios) contiene un directorio o subdirectorio. Para acceder al mismo basta con pulsar la tecla L. Hecho esto, el sistema operativo presentará una pantalla con la petición del nombre del directorio o subdirec-

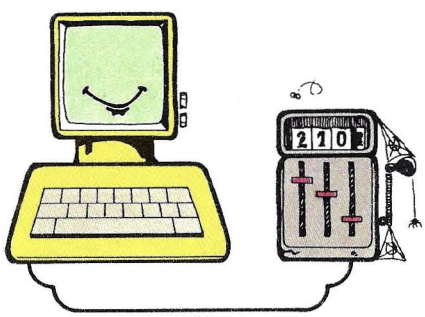
Los ordenadores electromecánicos

En la actualidad todos tendemos a pensar en el ordenador como en esa máquina compuesta de diminutos circuitos electrónicos, potente, rápida, precisa y, aparentemente, sin posibilidad de equivocarse. Olvidamos, sin embargo, que han existido ingenios, que aún sin acogerse a la tecnología que hoy sustenta a los modernos ordenadores, han llevado a cabo tareas similares.

Así, en 1926 L. J. Comric, Director del Observatorio de Greenwich, empleó una máquina Burroughs y 500.000 tarjetas perforadas para calcular la posición de la luna a medio día y a media noche hasta el año 2000, en la primera aplicación científica de un equipo de tabulación destinado en su origen a aplicaciones comerciales.

Posteriormente, y a partir de este trabajo, W. J. Eckert integró las órbitas de los asteroides.

El 1930 un profesor del Instituto de Tecnología de Massachusets, Vannevar Busch, construyó un ordenador de gran tamaño, denominado "Analizador diferencial", que operaba de forma analógica. Por medio de amplificadores mecánicos de par generaba la potencia necesaria para mover los trenes de engranajes que efectuaban las operaciones lógicas.

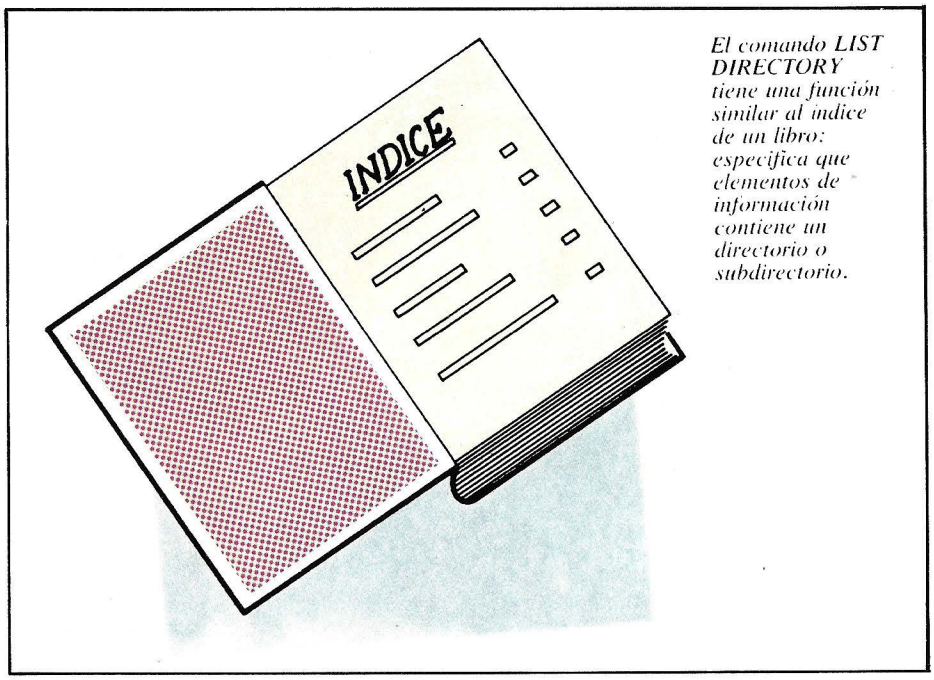


Una de las primeras aplicaciones del teclado para la introducción de datos y del teletipo para la salida de los mismos, se dio en el denominado Complex Computer.

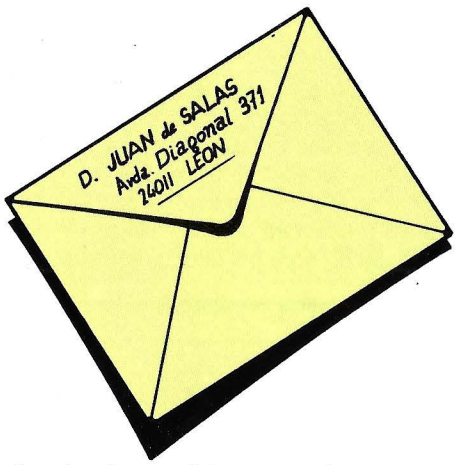
Este equipo, desarrollado en los laboratorios Bell a principios de los años 30, empleaba circuitos eléctricos de tipo biestable (los denominados flip-flop) e incluso fue ensayado en operación remota.

El primer ordenador electromecánico digital totalmente automático fue concebido en 1937 por Howard Aiken. Operaba con relés mecánicos y poseía capacidad para almacenar datos numéricos. La entrada y la salida se operaba a través de tarjetas perforadas, y podía ejecutar hasta 200 pasos por minuto.

Como vemos, el ordenador ha sido algo más que la máquina electrónica que hoy conocemos.



El comando LIST DIRECTORY tiene una función similar al índice de un libro: especifica que elementos de información contiene un directorio o subdirectorio.



Para localizar un fichero es preciso suministrar al ProDOS una información completa y ordenada de su emplazamiento. Algo semejante al método que se sigue a la hora de detallar la dirección completa del destinatario de una carta.

torio, debiendo especificar los nombres de los elementos superiores en jerarquía (directorio y subdirectorios) a aquel cuyo índice se desea examinar. La información presentada contiene los siguientes datos:

— tipo de fichero (binario, texto)
— tamaño de cada fichero en bloques
— protección de cada fichero
— fecha de modificación
— número de bloques disponibles y número de bloques utilizados.

Una vez presentada esta información, puede regresarse al menú de comandos accionando la tecla ESCAPE.

● COPY

El comando COPY permite copiar un fichero de un directorio a otro, ya sea dentro de un mismo volumen o bien de un volumen a otro.

Para acceder a este comando se pulsa la tecla C (COPY). De inmediato aparecerá una pantalla en la cual se solicita el nombre del fichero original y el nombre del fichero copia; en ambos casos hay que especificar la jerarquía de directorios y subdirectorios correspondiente.

En el caso de que el fichero copia tenga un nombre que coincida con el de otro fichero ya existente, aparecerá en la pantalla el siguiente mensaje:

DELETE EXISTING FILE (Y/N)

Dependiendo de la respuesta del usuario, se efectuará o no la copia en tal caso. Si ésta se realiza y es correcta, en la pantalla aparecerá el mensaje:

COPY COMPLETE

La operación de copia admite la intervención de referencias ambiguas, tanto en el nombre del fichero original como en el nombre de los ficheros copia. Su empleo hará que todos los ficheros de un directorio que compartan una característica determinada en su nombre, sean copiados a otro directorio.

● DELETE

El comando DELETE tiene por objeto la eliminación de un fichero o grupo de ficheros de un directorio determinado. Su uso permite liberar espacio de almacenamiento, y de esta forma mantener el rendimiento de las unidades de memoria de masa. Se accede al mismo pulsando la tecla D (DELETE) dentro del menú de comandos de fichero. La pantalla que presenta el sistema operativo como respuesta solicita toda la información relativa al nombre del fichero. Una vez realizada la operación de borrado, aparecerá en la pantalla el mensaje:

DELETE COMPLETE

El proceso de borrado también admite el empleo de referencias ambiguas a la hora de especificar el nombre del fichero; aunque ha de tenerse en cuenta lo indicado al hablar de éstas: si se emplea la referencia ambigua "=" no se solicitará confirmación del proceso a realizar, con lo cual corremos el riesgo de borrar un fichero valioso. Por ello, antes de efectuar esta operación es recomendable emplear el comando LIST DIRECTORY.

Los directorios también pueden ser borrados; si bien, en atención al criterio jerárquico de la estructura arborescente, sólo es posible eliminarlos cuando se encuentren totalmente vacíos de ficheros, y no antes.

● MAKE DIRECTORY

Cuando se formatea un disquete, éste recibe un nombre y se genera un directorio, tal y como hemos señalado anteriormente; sin embargo, los subdirectorios no están creados. Para ello, el ProDOS brinda el comando MAKE DIRECTORY, el cual permite efectuar esta creación según las necesidades del usuario.

Para acceder al mismo se pulsa la tecla M en el menú de comandos de ficheros (MAKE). Hecho esto, aparecerá una pantalla en la cual se solicita el nombre del directorio, estableciendo la dosificación jerárquica completa para acceder al mismo. Una vez completado el proceso de creación, se visualizará en la pantalla del ordenador el siguiente mensaje:

MAKE DIRECTORY COMPLETE

Tras este mensaje, la tecla ESCAPE devolverá el control al menú principal.

dBASE II (2)

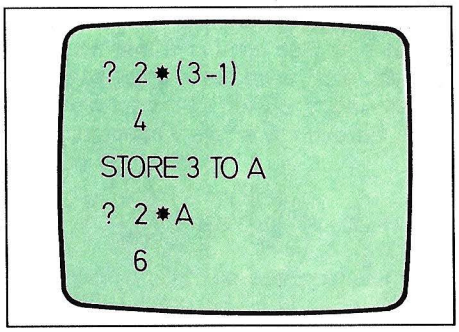
El lenguaje de comandos del dBASE II

En el capítulo precedente dio comienzo el estudio de la base de datos para ordenadores personales dBASE II. Como primera aproximación se describieron las operaciones elementales que permiten explorar el programa. En este segundo capítulo se estudiará la posibilidad de utilizar el dBASE II como herramienta de cálculo de alta potencia, exponiendo algunos comandos complementarios. Por último, se describirá el lenguaje de comandos para programación estructurada; lenguaje que dará una idea precisa de la potencia de este paquete de gestión, sin duda uno de los mejores dentro de su categoría.

dBASE II COMO CALCULADORA INTERACTIVA

Evidentemente, no se puede afirmar que una de las misiones fundamentales de una base de datos sea la de servir también como instrumento de cálculo; no obstante, la existencia de algún comando destinado a tal efecto puede resultar muy útil en determinados casos. Esto es lo que ocurre con el dBASE II. El comando interactivo representado por el símbolo "?", sirve para ejecutar instantáneamente expresiones de tipo matemático.

La forma de utilizar este comando es bien simple: basta con teclear el símbolo "?" seguido por un espacio en blanco y una expresión matemática, para que inmediatamente aparezca en la pantalla el resultado de dicha expresión. Por ejemplo, si tecleamos textualmente: "? 2 + 4" y a continuación pulsamos la tecla RETURN, en la siguiente línea podremos ver el resultado: 6. Para terminar de describir este comando interactivo, vamos a definir con precisión sus posibilidades:



Mediante el comando interactivo "?", el dBASE II permite utilizar al ordenador como una potente calculadora.

1. Obtención de resultados

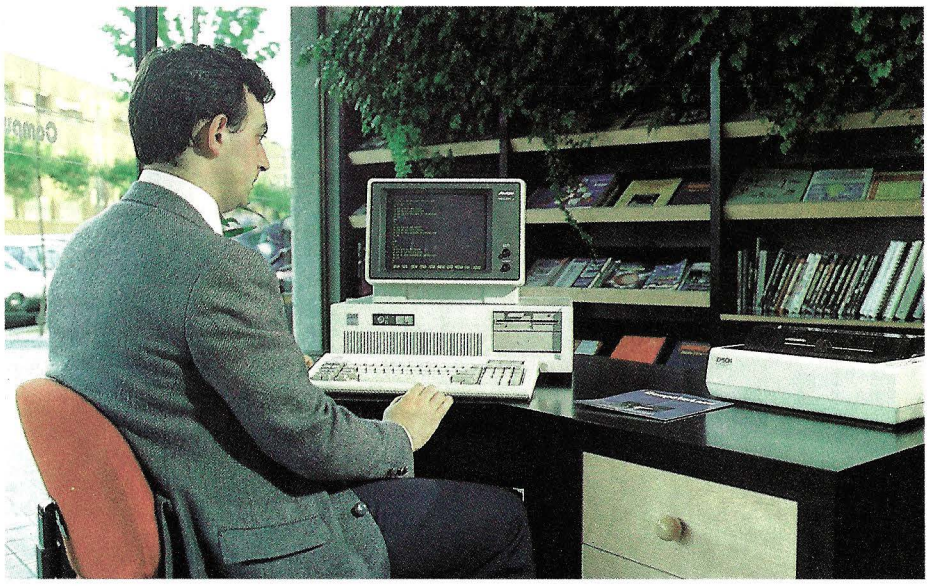
En el caso de teclear "?" seguido de una expresión, el programa contestará en la siguiente línea. Si el usuario desea obtener el resultado en la misma línea en la que se visualizará la expresión, la única modificación necesaria consiste en teclear ("??") dos interrogaciones en vez de ("?") una.

2. Utilización de variables

Mediante el comando STORE se puede almacenar cualquier valor en una variable; por ejemplo, podríamos teclear "STORE 2 TO A" y "STORE 4 TO B", de forma que a partir de ese momento la variable A tendrá como valor 2 y la variable B 4. Si a continuación tecleamos "? A + B" obtendremos como resultado el valor 6. En definitiva, en el comando interactivo "?" se pueden utilizar variables siempre que estas tengan asignado previamente un valor.

3. Utilización del comando "?" sobre relaciones

Siempre que previamente se haya ejecutado el comando USE, identificando a una relación, al teclear "? NOMBRE", (donde NOMBRE es la denominación de alguno



El paquete gestor de bases de datos dBASE II es una aplicación de la que existen versiones destinadas a varios sistemas operativos y equipos específicos; entre ellos se encuentra el IBM-PC y toda su saga de compatibles.

de los campos de la relación utilizada) se obtendrá el valor de dicho campo en el registro activo.
Como se observa, este sencillo comando utilizado con precisión puede servir para mucho más que la evaluación de expresiones matemáticas. En el fondo, su verdadera utilidad se obtiene mezclando, por un lado su capacidad de cálculo, y por otro la posibilidad de utilizar variables inicializadas manualmente o extraídas de la base de datos.

NUM	NOM	SAL	CAT
1	nieto lopez, luis	1500000	ingeniero
2	garcia perez, julian	1250000	administrativo
3	ruiz ortiz, ernesto	1900000	ingeniero
4	perez lopez, andres	2000000	director

Para mostrar un ejemplo relativo a la obtención de informes con el dBASE II, partiremos de una relación de personal con cuatro datos por empleado: número (NUM), nombre (NOM), salario (SAL) y categoría (CAT).

```
 Enter options, m = left margins, l = line/page, w = page width
<RETURN>
 Page heading ? (y/n)
<'Y' RETURN>
 Enter page heading:
<'                                    ' RETURN>
 Double space report? (y/n)
<'Y' RETURN>
 Are total required? (y/n)
<'N' RETURN>
 Col        Width,  Contens
  01          4, NUM
 Enter     heading: num.
  02         20, NOM
 Enter     heading: nombre
  03          7, SAL
 Enter     heading: salario
  04         20, CAT
 Enter     heading: categoria
<RETURN>
```

El primer paso para producir un informe consiste en describir las características del mismo. En la figura se observa la definición de un sencillo informe de personal.

```
Page no. 00001
05/07/85

                    LISTADO DE PERSONAL

num.      nombre          salario     categoria

  1  nieto lopez, luis    1500000   ingeniero

  2  garcia perez, julian 1250000   administrativo

  3  ruiz ortiz, ernesto  1900000   ingeniero

  4  perez lopez, andres  2000000   director
```

A partir de la relación y de los formatos descritos en las figuras precedentes, se obtendrá el informe que reproducimos.

PRINCIPALES COMANDOS DEL dBASE II

Además de los comandos elementales ya descritos en el número anterior y los dedicados a operaciones de cálculo, dBASE II cuenta con todos los comandos tradicionales para el mantenimiento de una base de datos. Con objeto de no resultar reiterativos, no vamos a incluir aquí una lista exhaustiva de dichos comandos. Sin embargo, lo que sí vamos a estudiar son dos herramientas de gran utilidad: una es la destinada a la obtención de informes a partir de la base de datos, y la segunda consiste en el operador JOIN. Si bien la primera, más o menos sofisticada, se puede encontrar en cualquier programa gestor de ficheros, la segunda es difícilmente tratable y, precisamente por ello, sólo los sistemas más avanzados de bases de datos la incorporan.

- REPORT

Cuando se necesita obtener un informe basado en los datos contenidos en la base, y dicho informe tiene un nivel de complicación que resulta excesivo para los comandos elementales de obtención de información —bien sea por la estructura del informe, por las características de edición o por cualquier otra circunstancia—, el paquete dBASE II ofrece el comando REPORT para obtenerlo sencilla y cómodamente.
El formato del comando REPORT es el siguiente:

REPORT FORM <NF> [[ALL] | [RECORD N] | [NEXT M]]
[FOR <EXPRESION>] [TO PRINT]

Donde el significado de los distintos argumentos es el siguiente:

a) FORMATO DEL INFORME (FORM <NF>)

Indica el formato con que debe imprimirse el informe; por defecto, el dBASE II trabaja con líneas de 8 columnas reservadas, 56 líneas por página y 80 caracteres por línea. No obstante, siempre resulta necesario definir formatos de edición.
Para crear un nuevo formato basta con teclear REPORT, para que inmediatamente aparezca un mensaje solicitando un nombre (NF) para dicho formato; dBASE II solicitará este nombre con la siguiente pregunta: "ENTER REPORT FORM NAME". Una vez tecleado el nombre propio, el programa solicitará, de forma interactiva, el resto de las características del informe, como modificación del formato por defecto, cabeceras, espaciado, subtotales y, por supuesto, el contenido de cada una de las columnas que formará parte del informe. Una vez detalladas todas estas características se debe pulsar la tecla ESCAPE, con lo que quedará definido el formato.

b) AMBITO DE ACTUACION

Para solicitar un informe sobre cualquier relación (previamente se debe haber especificado el nombre de la misma me-

diante el comando USE), se podrá definir el ámbito de actuación de forma análoga a como se ha hecho con el resto de los comandos ya estudiados; esto es: se indicará ALL para toda la relación, RECORD N para un registro determinado y NEXT M para un bloque de registros.

c) CONDICION DE VALIDACION (FOR <EXPRESION>)

Otro parámetro opcional de este comando consiste en la imposición de una condición que debe ser satisfecha por los campos de un registro para que éste pase a formar parte del informe. Como en otros casos, dicha condición se puede indicar mediante la palabra reservada FOR, seguida por la expresión que incluye la condición de validación.

d) IMPRESION (TO PRINT)

Por último, en el caso de que se desee obtener el informe por impresora, se puede incluir el parámetro TO PRINT para que, de forma automática, el informe pase a impresión.

Además de los parámetros detallados anteriormente existen otras posibilidades adicionales para la edición de informes; por ejemplo: el comando COUNT que cuenta el número de registros procesados, el comando SUM para calcular sumatorios, etc.

- JOIN

Una operación característica del álgebra relacional, y por lo tanto sólo utilizable en bases de datos relacionales, es el JOIN de dos relaciones. El dBASE II dispone de este comando y puede ser invocado de la siguiente forma:

```
JOIN TO <NUEVA RELACION>
        FOR <EXPRESION> [FIELDS <lista>]
```

El funcionamiento de este operador exige que previamente se hayan "abierto" las dos relaciones a unir, para lo cual se debe ejecutar USE RELACION-1, SELECT SECUNDARY, USE RELACION-2; de esta forma una de las relaciones queda activa y otra —mediante la instrucción SELECT— queda "recordada".

Una vez realizadas estas operaciones, el comando JOIN producirá una NUEVA RELACION mediante la unión de las dos relaciones iniciales. Para ello, en la EXPRESION se debe indicar qué campos de ambas relaciones permiten establecer la unión y, por último, mediante el parámetro FIELDS, se indican los campos que se incluirá en la relación producida.

La potencia de esta operación queda clara, puesto que mediante ella no se utilizan simples datos como elemento fundamental de trabajo, ni tan siquiera registros, sino que se trabaja con relaciones completas produciendo nuevas relaciones.

Generadores de programas

Uno de los paquetes de aplicación más relacionado con las bases de datos es el denominado generador de programas. Existen buenas aplicaciones de este tipo desarrolladas para su explotación en grandes ordenadores; en cambio, para ordenadores personales nos tenemos que conformar con versiones reducidas.

Como su propio nombre indica, un generador de programas no es más que un programa cuya misión consiste en producir de forma automática nuevos programas. De alguna manera se puede decir que su misión es sustituir a un programador y permitir que un usuario no especializado en informática produzca nuevos programas de acuerdo a sus necesidades.

Las principales características de los generadores de programas son las siguientes:

1. Metalenguaje

Cuando un usuario encarga el desarrollo de un programa a un informático, debe transmitirle sus necesidades. En este caso, en vez de comunicarse con el informático, el usuario deberá comunicarse con el generador. Para esta misión todo generador debe incorporar un metalenguaje que será utilizado por el usuario.

Uno de los factores críticos para el éxito de un generador de programas es que el metalenguaje sea de muy alto nivel y fácil de manejar. De poco servirá si la complejidad del lenguaje manejado por el usuario es similar a la de los lenguajes convencionales de programación.

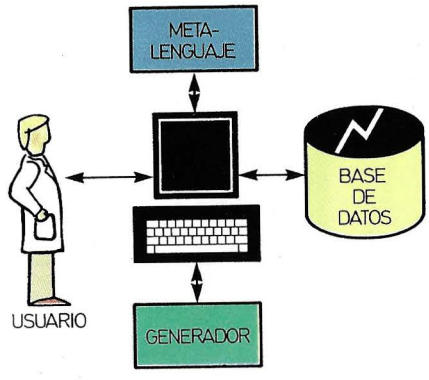
2. Objeto generado

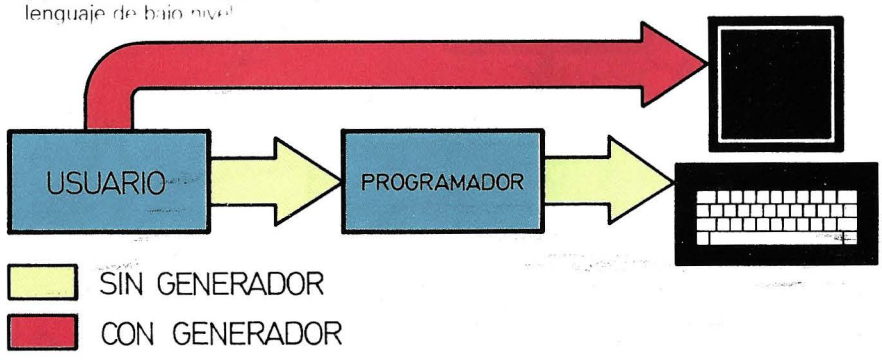
A partir de las especificaciones indicadas con el metalenguaje, el generador debe producir un programa. Existen tres tipos fundamentales de técnicas para dicha generación.

- Producción de un programa redactado en un lenguaje convencional (COBOL, FORTRAN, BASIC ...).
- Producción de un programa interpretable directamente por el ordenador.
- Producción de un programa compilable, y por lo tanto más rápido en ejecución, en lenguaje de bajo nivel.

3. Base de datos

El tercer elemento importante de un generador de programas es la estructura de información donde obtendrán los datos los programas generados. Normalmente dicha estructura de información suele consistir en una base de datos relacional.





PROGRAMACION ESTRUCTURADA CON dBASE II

Una de las más potentes herramientas del dBASE II consiste en la posibilidad de programar nuevos comandos. Al respecto, basta con teclear MODIFY COMMAND NOMBRE, donde este último es precisamente la denominación que se dará al nuevo comando (en realidad, se trata de un auténtico programa). Después de haber indicado el nombre, dBASE II presentará una pantalla en blanco si dicho comando no existía previamente, o una pantalla con las instrucciones que anteriormente se hubieran introducido.
La actualización o introducción de nuevas instrucciones se realiza con un editor de pantalla completo (full screen) que resulta extremadamente sencillo y cómodo de utilizar.
En la programación se pueden utilizar prácticamente todos los comandos y funciones aportadas por el dBASE II. Las principales instrucciones para la programación son las siguientes:

- IF .. ELSE .. ENDIF

Permite imponer una condición con operadores lógicos, de forma que en caso de verificarse dicha condición se ejecutará un conjunto de comandos (los situados antes de la palabra ELSE), mientras que si la condición no se verifica se ejecutará otro conjunto distinto de instrucciones (las situaciones entre ELSE y ENDIF).

- DO WHILE ... ENDDO

En este caso se trata de una instrucción repetitiva. A continuación de DO WHILE se debe imponer una condición y uno o varios comandos, situados entre la condición y la palabra ENDDO; todos ellos se ejecutarán repetidamente, siempre que se verifique la condición impuesta.

- WAIT TO VARIABLE

Sirve para permitir al usuario introducir un simple carácter que quedará almacenado en una variable en memoria.

- INPUT "mensaje" TO VARIABLE

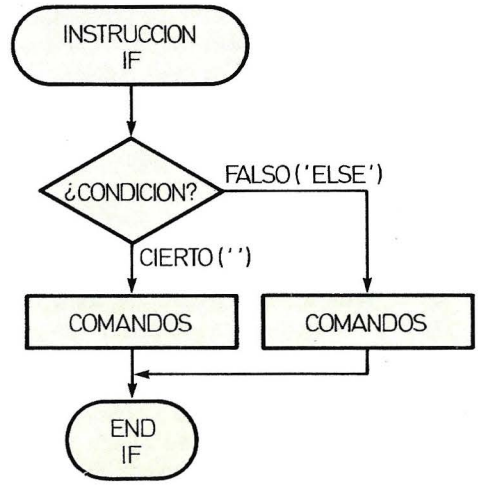
Se utiliza para entrada de datos que serán solicitados al usuario mediante el "mensaje" especificado y almacenados en variables de memoria.

- ACCEPT "mensaje" TO VARIABLE

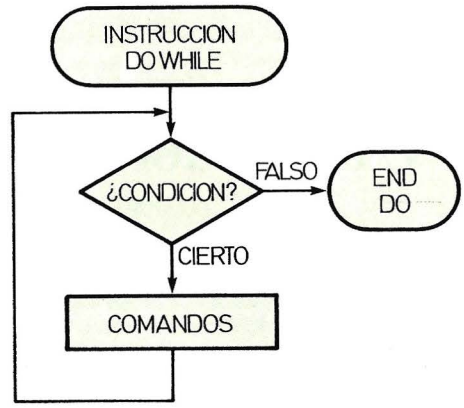
Su misión es análoga a la del comando anterior y suele utilizarse para entradas de datos alfabéticas.

- @ <coordenadas> [SAY <'mensaje'>]

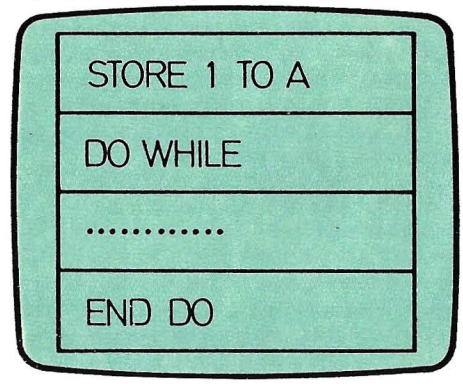
Sirve para escribir mensajes o posicionar al cursor en una posición determinada de la pantalla. Mediante su utilización se pueden generar sistemas de menús que guíen al usuario en la explotación de una aplicación.



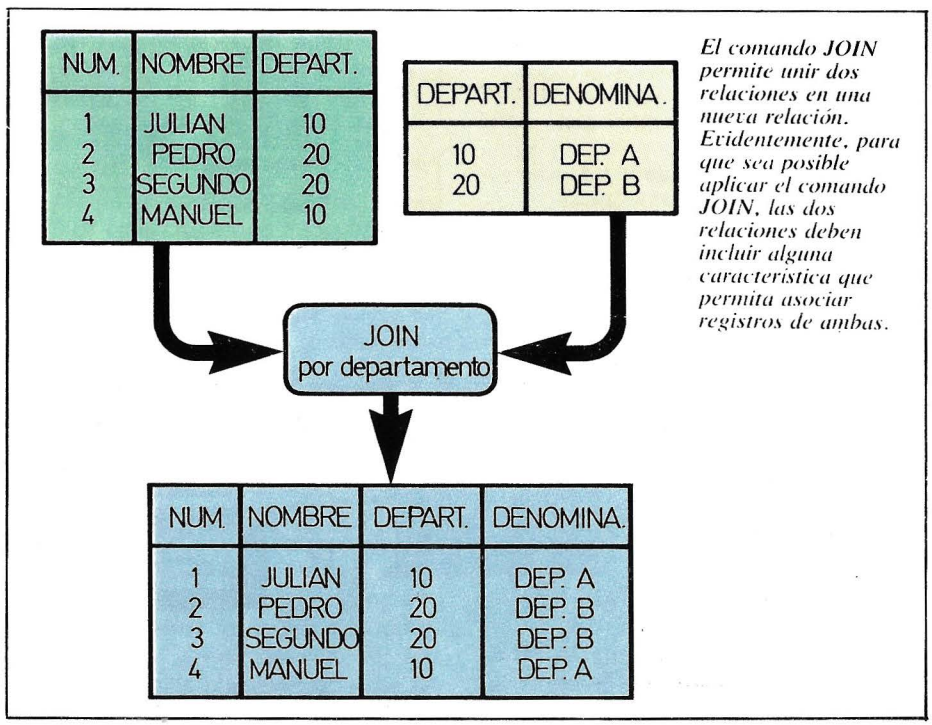
Por medio de la instrucción IF... ELSE... ENDIF se pueden tomar decisiones según se verifique o no una determinada condición.



La instrucción DO WHILE... ENDDO se puede utilizar para definir bloques de comandos que se ejecutarán reiterativamente mientras se verifique la condición impuesta.



La orden MODIFY COMMAND permite crear y/o modificar "programas" que serán ejecutados por el dBASE II como si se tratara de verdaderos comandos.



El comando JOIN permite unir dos relaciones en una nueva relación. Evidentemente, para que sea posible aplicar el comando JOIN, las dos relaciones deben incluir alguna característica que permita asociar registros de ambas.

Del BASIC al código máquina

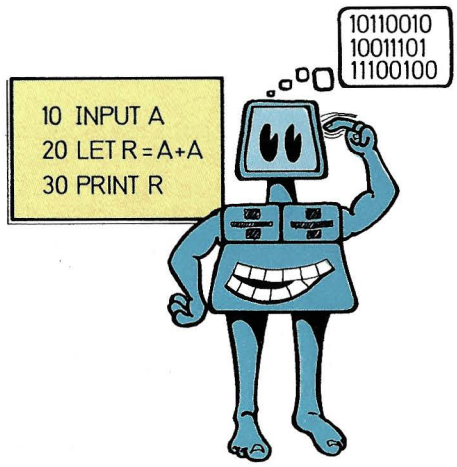
En la recóndita intimidad del ordenador

El BASIC es un lenguaje de alto nivel. Ello significa que se trata de una representación no directamente inteligible por el ordenador, sino próxima al lenguaje hablado convencional. En consecuencia, el ordenador ha de convertir las expresiones BASIC a otro código que sea capaz de entender y ejecutar: el código máquina. Así pues, ¿por qué no programar directamente en código máquina?
El manejo del código máquina tiene sus ventajas, pero también sus inconvenientes. Este es, precisamente, el tema en discusión a lo largo del presente capítulo.

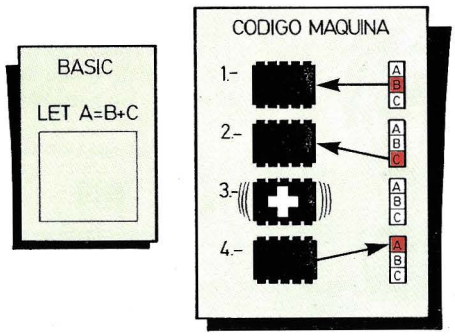
ASPECTO DEL CODIGO MAQUINA

El código máquina corresponde a la representación de instrucciones más cercana al ordenador. Esta representación cabe imaginarla como filas de unos y ceros. Cada ocho de estas cifras se agrupan en un byte, y cada byte o grupo de ellos es capaz de dar cuerpo a una instrucción elemental.
El repertorio de instrucciones depende de la configuración física (hardware) del ordenador, y más concretamente del microprocesador que constituye su cerebro. Distintos microprocesadores utilizan diferentes repertorios de instrucciones y, por lo tanto, los programas en código máquina no coincidirán.
Las instrucciones en código máquina son mucho más elementales que las del lenguaje BASIC. Por ello, una instrucción BASIC suele descomponerse en varias de código máquina. Así, por ejemplo, la actividad asociada al tratamiento de la sentencia BASIC LET A=B+C ha de seguir los siguientes pasos:

— Recoger el dato almacenado en la posición de memoria indicada como B.
— Recoger el dato almacenado en la posición de memoria referenciada como C.
— Sumar ambos datos.
— Almacenar el resultado en la posición de memoria identificada por A.



A pesar de que nos comunicamos con el ordenador en BASIC, éste no es su lenguaje «natural». La máquina se ve obligada a traducir las instrucciones BASIC a su propio código interno: el código máquina.



Las instrucciones del código máquina son mucho más elementales que las del BASIC. Ello hace que la más simple instrucción BASIC se traduzca en varias instrucciones de código máquina.

Este método sería válido si se tratara de datos de un byte (ocho bits). No obstante, si los datos a sumar fueran números enteros, se utilizarían dos bytes para codificar cada uno de ellos, con lo cual la cosa se complica. Habría que recoger dos bytes por dato y realizar dos sumas, una con cada byte. Más compleja todavía sería la suma de datos de doble precisión, que utilizan ocho bytes.
Como se observa, el código máquina puede resultar bastante incómodo a la hora de programar procesos complicados. Sin embargo, la ejecución de subrutinas en código máquina es mucho más rápida que en BASIC. Ello se debe a que el ordenador no tiene que traducir las órdenes, puesto que ya están en su lenguaje interno.

EMPLEO DE CODIGO MAQUINA EN UN PROGRAMA BASIC

Dentro de un programa en BASIC se pueden emplear rutinas escritas en código máquina. Estas rutinas pueden coexistir perfectamente con el programa BASIC, siempre y cuando las instrucciones escritas en código máquina se sitúen en la zona de memoria adecuada.
Para escribir una rutina en código máquina hay que partir de un buen conocimiento del repertorio de instrucciones del microprocesador. Una vez planificada la rutina, ésta debe traducirse a los códigos binarios correspondientes a las respectivas instrucciones. Los números calculados se

POKE

Deposita un dato numérico en la posición de memoria especificada en su argumento.

FORMATO: POKE <posición>, <dato numérico>
EJEMPLOS:

```
10 POKE 32500,DATO
50 POKE 2000,22
```

trasladan a la zona de memoria adecuada por medio del comando BASIC POKE. Este tiene como misión depositar un número en una determinada posición de memoria, y para ello hace uso de dos datos en su argumento: el primero indica la posición de memoria a rellenar, mientras que el segundo coincide con el dato a depositar en la misma. Por lo tanto, su formulación tendrá el siguiente aspecto:

(Número de línea) POKE <dirección>, <número>

A partir del comando POKE es posible escribir un programa capaz de emplazar una rutina en código máquina en las posiciones de memoria que desee el usuario. Su intervención reiterada se consigue al incluir el comando POKE dentro de un bucle FOR/NEXT.
El siguiente ejemplo coincide con una porción de programa BASIC adecuado para escribir en memoria una rutina en código máquina:

```
100 FOR I=1000 TO 2000
110 READ A
120 POKE I, A
130 NEXT I
...
■
```

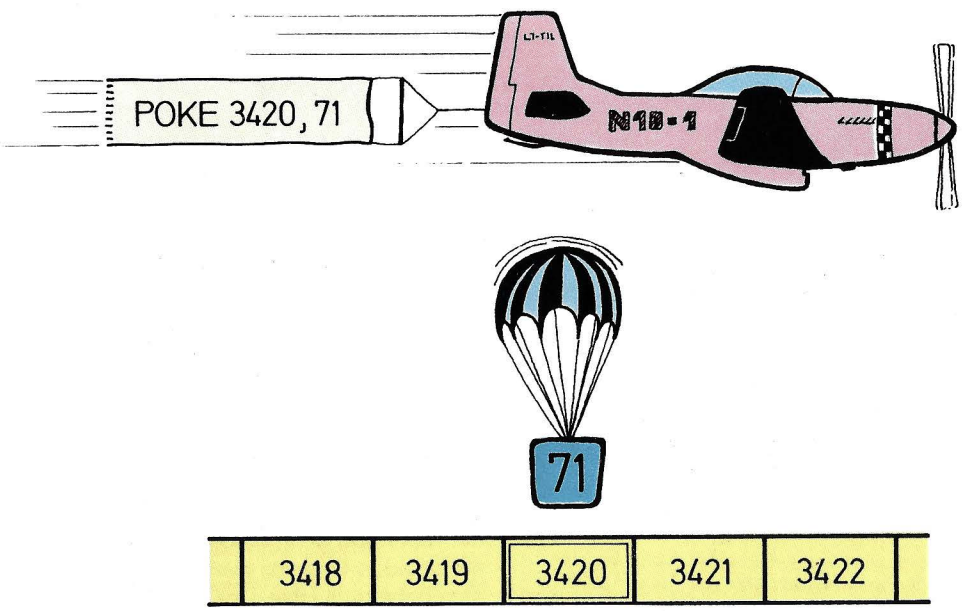
En efecto, estas líneas colocarán una rutina en código máquina en las posiciones de memoria que van desde la 1000 a la 2000. La línea 110 lee los datos que conforman la rutina extrayéndolos del argumento de instrucciones DATA. En éstas se han de encontrar los datos numéricos que corresponden a las instrucciones escritas en código máquina. Como ya se ha señalado, estos códigos serán distintos para cada microprocesador.
De la misma forma que se dispone de un comando para depositar un dato en una posición de memoria, también se puede hacer uso de una función para recuperarlo. Esta función se asocia a la palabra clave PEEK. Su formulación es la que sigue:

(Número de línea) <var.>=PEEK (<dirección>)

En este formato se ha situado a la función PEEK dentro de una sentencia de asignación. Sin embargo, al tratarse de una función, PEEK puede utilizarse en el argumento de un comando. Por ejemplo:

PRINT PEEK(4000)

mostrará en pantalla el contenido de la posición de memoria número 4000.



El comando POKE se encarga de depositar un dato numérico en una determinada posición de memoria especificada en su argumento.

LA DIVISION DE LA MEMORIA

Anteriormente, se ha precisado que pueden coexistir en memoria rutinas en código máquina y programas en BASIC. Para ello es necesario emplazar a las primeras en una zona de la memoria que esté libre; tarea que supone conocer el uso que el ordenador hace de las distintas zonas de su memoria interna.
Por regla general, un ordenador basado en un microprocesador de ocho bits será capaz de direccionar una memoria de 64 Kbytes. La razón es bien sencilla. Un microprocesador de ocho bits utiliza dos bytes para direccionar la memoria. Esto es, para identificar cada posición de memoria en la que se almacena un byte. Dos bytes forman un conjunto de 16 (8+8) bits, con los que se pueden obtener un número máximo de 65.536 combinaciones o palabras distintas. Veamos: con un bit se pueden dar dos posibilidades (1 ó 0), con dos bits cuatro (00, 01, 10 y 11)... Y, en general, con n bits se tienen 2^n combinaciones diferentes. Un sencillo cálculo muestra que con los dieciséis bits indicados se pueden diferenciar 65. 536 posibilidades. Si se divide esta cantidad por 1024 (1024 es igual a 2^{10}: el múltiplo de "Kilo" (1 K) en el sistema binario) se obtiene como resultado 64. Es decir, 65.536 corresponde a 64 K.
La distribución de la memoria principal o

interna es muy particular de cada máquina. Si bien, en términos generales cabe distinguir dos grandes zonas. Por un lado está la memoria de uso general, en la que se almacenan los programas y datos. Esta zona, que permite operaciones de lectura y escritura de su contenido, es conocida como RAM (siglas de Random Access Memory: memoria de acceso aleatorio o acceso directo). El otro bloque de memoria contiene el intérprete BASIC (en aquellos aparatos que no necesitan cargarlo de una unidad externa) y las rutinas básicas para el funcionamiento del sistema. El contenido de esta segunda zona es fijo y no puede ser modificado bajo ningún concepto. Se trata de la denominada ROM (de Read Only Memory: memoria de sólo lectura).

La memoria RAM del ordenador se subdivide, a su vez, en varias porciones, cada una de ellas con un cometido preciso.

En primer lugar está la zona destinada al almacenamiento del programa BASIC. Ahí es donde se guardan las líneas de programa introducidas por el usuario, y de donde se leen cuando se ejecuta. Además de la zona de programa, el BASIC necesita otra para almacenar datos. En esta segunda es donde se sitúan las variables que utiliza el programa. Ambas zonas de memoria son las que se emplean para los programas BASIC.

Aparte de las zonas destinadas al BASIC, pueden existir otras porciones de la RAM reservadas al uso particular del sistema; por ejemplo, habrá una zona destinada a guardar la información que se visualiza en pantalla, y otras para almacenar los datos recibidos del teclado o mandados a la impresora, etc. El sistema se reserva otro segmento de la RAM para almacenar datos de uso propio: la denominada zona de variables del sistema.

Los límites de las distintas partes de la memoria vendrán establecidas por el diseño hardware del aparato. Esta información no es de uso habitual, por lo que no suele incluirse en los manuales básicos de cada ordenador. De ahí que la distribución interna de la memoria haya de buscarse en manuales avanzados o libros dedicados especialmente a las interioridades más profundas de cada máquina.

Por regla general, las rutinas en código máquina se situarán en la zona BASIC; bien sea ocupando el espacio sobrante de la zona de programa, o bien en el espacio de variables que no se utilice. Para colocarlas debidamente es necesario tener un conocimiento preciso de la longitud, situación y funcionamiento de dichas zonas de memoria.

Con los comandos PEEK y POKE se puede acceder a cualquier posición de la memoria. Hay que tener en cuenta que la ROM, al ser su contenido fijo y no modificable, no permitirá el uso de POKE.

Como se ha indicado, la distribución de la memoria depende del aparato; luego resulta totalmente inútil emplear los comandos PEEK o POKE sin conocer tal distribución. Es más, su empleo indiscriminado puede originar el mal funcionamiento de la máquina.

Conociendo a fondo el cometido de cada una de las posiciones de la memoria del ordenador, será posible alterar su contenido para obtener resultados concretos.

En la zona reservada a variables del sistema se encuentran posiciones que pueden ser muy útiles al programador. Algunas de estas variables pueden contener datos como la frecuencia de repetición de las teclas, el número de línea en ejecución, la posición del cursor en la pantalla, o incluso los límites de las zonas de programa y de variables. En algún caso, leer o modificar esos datos puede ser importante. Ello se consigue con los citados PEEK y POKE.

PEEK

Extrae el contenido de la posición de memoria indicada.

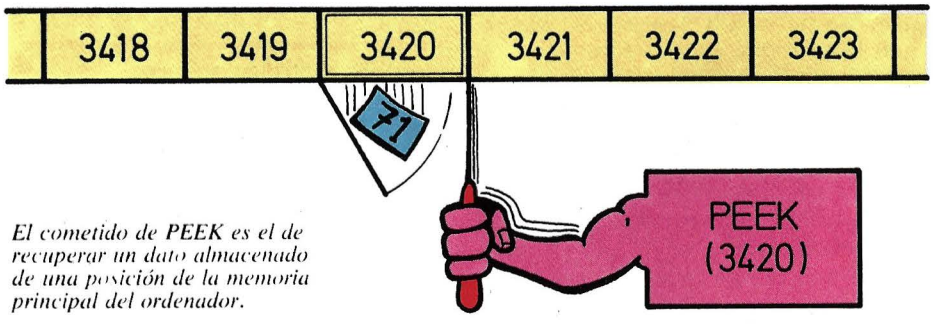
FORMATO: PEEK (<posición>)
EJEMPLOS:

```
20 LET D=PEEK(32500)
70 PRINT PEEK(2000)
```

OCUPACION DE MEMORIA

Cuando se desea utilizar rutinas en código máquina como complemento de un programa BASIC, es necesario controlar el espacio de memoria en uso. Ello también es de gran utilidad cuando se está introduciendo un programa muy largo y nos acercamos al límite de memoria.

El BASIC incluye una función capaz de revelar el número de bytes libres. Se trata de la función FRE. Esta admite un dato como argumento; si bien, el referido dato



El cometido de PEEK es el de recuperar un dato almacenado de una posición de la memoria principal del ordenador.

FRE

Proporciona el número de bytes que quedan sin utilizar.

FORMATO: FRE(0)
EJEMPLOS:

```
20 PRINT FRE(0)
70 LET LIBRE=FRE(0)
```

CLEAR

Borra el contenido de todas las variables del BASIC.
Opcionalmente sitúa el límite de la zona de BASIC.

FORMATO: CLEAR [<posición>]
EJEMPLOS:

```
20 CLEAR
70 CLEAR 33450
```

no afecta al funcionamiento de FRE. Por ello, suele utilizarse FRE(0) para obtener la cantidad de memoria que aún no ha sido ocupada.
Cabe precisar que el dato proporcionado por FRE se refiere únicamente a la zona de memoria reservada a los programas BASIC; por lo que su respuesta no contempla la zona de variables BASIC, y mucho menos a los segmentos de memoria reservados al sistema.
Se ha indicado anteriormente que las rutinas en código máquina se suelen situar en la parte de memoria dedicada al BASIC. Ello exige adoptar una primera precaución: evitar que se mezclen rutinas en código máquina y programas BASIC. Al efecto, existe normalmente una variable del sistema que contiene el valor de la posición de memoria que marca el límite de la zona de programa. Este límite no puede ser rebasado durante el proceso de introducción de un programa BASIC. Así pues, es un buen método situar esas rutinas a continuación de dicho límite. Aunque ello no es siempre posible, puesto que la memoria situada a continuación puede servir para otro cometido. En tal caso, el truco consiste en "engañar" a la máquina diciéndole que la zona de programa acaba antes de lo estipulado. Para lograrlo hay que cambiar el valor de la posición límite, accediendo a la adecuada variable del sistema.
Esta misma operación puede realizarse directamente por medio de un comando BASIC. Dicho comando se invoca mediante la palabra clave CLEAR.

El comando CLEAR suele tener un doble cometido: por una parte se encarga de poner a cero todas las variables del programa, y por la otra se ocupa de definir el límite anteriormente indicado. Esto último se consigue especificando la nueva posición límite en su argumento.
En general, la formulación de una instrucción CLEAR coincide con la siguiente:

(Número de línea) CLEAR [,<posición>]

En ella, la cláusula <posición> indica la posición de memoria que se ha de tomar como límite para la zona de programa.
Al igual que suele ocurrir con muchos otros comandos BASIC, CLEAR no realiza la misma función en todos los aparatos. Cada ordenador tiene, como ya se ha señalado, una distribución de memoria diferente, lo que implica que la zona de BASIC puede estar situada en distintas posiciones. Además, es posible que el límite modificado con CLEAR no corresponda a la zona de programas, sino a la de variables BASIC.

UTILIZANDO RUTINAS EN CODIGO MAQUINA

Suponga que ya se ha diseñado la rutina en código máquina. Se han traducido las instrucciones a sus correspondientes códigos numéricos, y se han situado éstos en las posiciones de memoria adecuadas, mediante comandos POKE. Ahora ya sólo falta utilizarla.
El BASIC contempla dos formas de llamar a una rutina en código máquina. La primera y más elemental hace uso del comando CALL. Para emplear CALL es necesario conocer la posición de memoria en la que comienza la rutina en cuestión, posición que se ha de indicar en el argumento de CALL. Su formato muestra el siguiente aspecto:

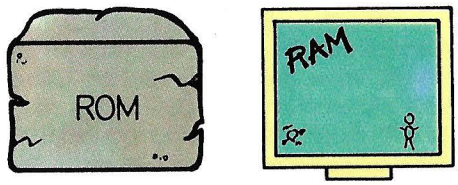
(Número de línea) CALL <posición de memoria> [(<parámetro>[,<parámetro>, ...])]

CALL

Efectua una llamada a una subrutina en código máquina situada en la posición que se indica.

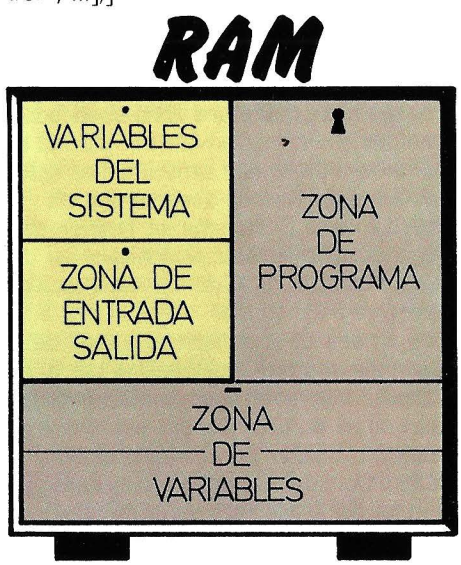
FORMATO: CALL <posición>
EJEMPLOS:

```
10 CALL 32500
```



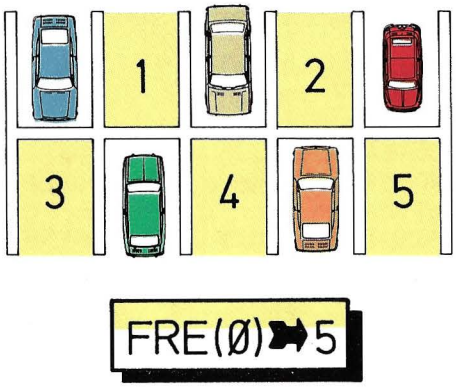
MEMORIA

La memoria principal del ordenador consta de dos zonas bien diferenciadas: la ROM, cuyo contenido es fijo e imposible de alterar, y la RAM en la que es posible tanto leer como escribir datos.



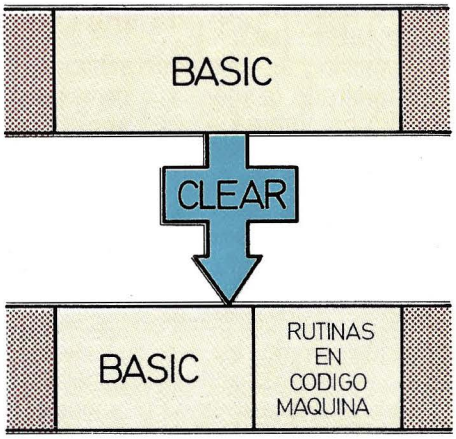
La memoria RAM se subdivide en varias zonas, de las cuales unas son utilizadas por el BASIC mientras que otras quedan reservadas para uso exclusivo del sistema.

Cuando la secuencia de ejecución del programa tropieza con un comando CALL, el control se transfiere a la rutina especificada. Al terminar la ejecución de la misma, la secuencia de proceso retorna a la instrucción BASIC situada a continuación de CALL. Se observa pues que CALL actúa como una llamada a subrutina; equivalente a GOSUB, pero con la salvedad de que, mientras GOSUB llama a una subrutina BASIC, CALL hace lo propio con una subrutina escrita en código máquina.
El comando CALL no sólo sirve para llamar a una subrutina, sino que puede realizar esa llamada "pasando un parámetro".



La función FRE(0) proporciona el número de bytes libres que existen en la memoria RAM a disposición de los programas que desee introducir el usuario.

Si la rutina en código máquina necesita que se le proporcione dato, éste puede ser mandado, desde el programa BASIC, a modo de "parámetro". Dicho parámetro será un dato que utilizará la rutina en código máquina para elaborar sus cálculos. Es posible mandar varios parámetros, si bien el uso que se haga de ellos dependerá exclusivamente del funcionamiento de la rutina.
Existe un método alternativo para utilizar rutinas en código máquina dentro de un programa BASIC: se trata de utilizar dichas rutinas como funciones definidas por el usuario.



El comando CLEAR puede ser empleado para alterar la posición límite de la zona BASIC. Con ello se reserva espacio para rutinas en código máquina.

FUNCIONES DE USUARIO EN CODIGO MAQUINA

En un capítulo precedente se habló de las funciones definibles por el usuario en su doble aspecto: definición y uso. En la definición se indicaban las operaciones a realizar, por medio del comando DEF FN, mientras que para utilizarlas bastaba con dar el nombre de la función precedido por las letras FN.
El BASIC permite definir otro tipo de funciones. Si aquellas eran creadas bajo el ámbito del BASIC, estas nuevas funciones se escriben en código máquina, aunque el modo de operación es similar. Previamente hay que memorizar la rutina que realiza las operaciones pertinentes en el lugar apropiado. Con la rutina situada en una posición de memoria conocida, el resto se hace directamente desde el BASIC. Para empezar, se ha de definir la función en código máquina; y como quiera que la rutina ya está creada sólo falta indicar su dirección de comienzo. Ello se consigue utilizando el comando DEF USR cuyo formato coincide con:

(Número de línea) DEF USR<dígito>=<dirección>

En el formato se observan dos datos a especificar. Por una parte, el denominado <dígito>: dato que servirá para diferenciar unas funciones de otras. Como se trata de un sólo dígito, sólo se podrán definir diez funciones en código máquina al mismo tiempo. De todas formas, este número se revela más que suficiente en la mayor parte de los casos.
El otro dato hace referencia a la dirección de comienzo de la rutina en código máquina. Así pues, los siguientes serían ejemplos válidos de definición de funciones de esta categoría:

```
10 DEF USR1=33450
20 DEF USR3=1000
30 DEF USR6=23
```

Para pasar el control de la ejecución del BASIC a una rutina en código máquina se hace uso del comando CALL.

DEF USR

Define una función de usuario utilizando una rutina en código máquina.

FORMATO: DEF USR<dígito>=<posición>
EJEMPLOS:

```
20 DEF USR3=32500
70 DEF USR7=23
```

USR

Proporciona un dato calculado en una rutina escrita en código máquina.

FORMATO: USR<dígito>(<parámetro>)
EJEMPLOS:

```
20 PRINT USR1(35)
70 LET LIBRE=USR3(A$)
```

Los ejemplos definen las funciones de usuario USR1, USR3 y USR6, cuyas rutinas en código máquina comienzan en las posiciones 33450, 1000 y 23, respectivamente. Como es lógico, en esas direcciones se deben encontrar las correspondientes rutinas.
Este es el método a seguir para definir las funciones. Para utilizarlas bastará con invocar el nombre que se ha utilizado en la definición. Al tratarse de funciones, éstas deben figurar dentro de sentencias de asignación o deben estar precedidas por comandos. Por ejemplo:

```
180 LET A=USR1(10)
200 LET B=A+USR3(A)
300 PRINT USR6(B)
```

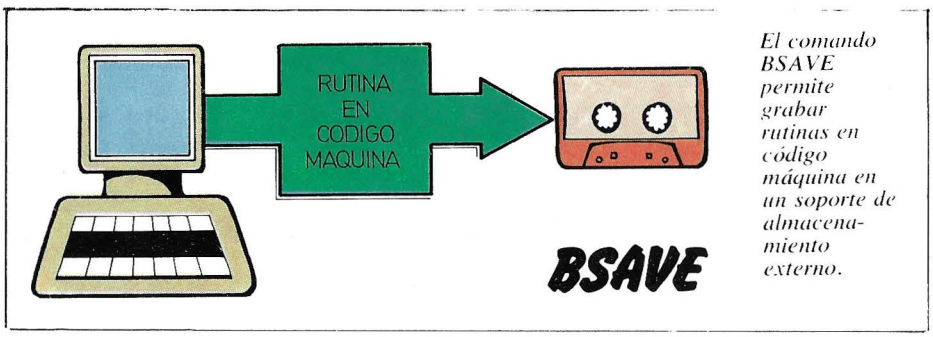
Se observa que, al igual que ocurre con casi todas las funciones BASIC, éstas se acompañan de un parámetro situado entre paréntesis en su argumento. En realidad, las rutinas en código máquina explotadas de acuerdo a este método pueden emplear dos tipos de parámetros.
Por un lado, hay que hablar del parámetro de entrada, ya comentado. Este es idéntico al que admite opcionalmente el comando CALL. Se trata pues de un dato que se le transfiere a la rutina en código máquina para que ésta lo procese. El otro tipo de parámetro consiste en el dato de salida proporcionado por la propia rutina; dato que se asigna a la variable (si está en una sentencia de asignación) o se muestra por pantalla (si se encuentra como argumento de un PRINT). Esta última posibilidad no la facilitaba el uso de CALL.
Todas las características relativas a los parámetros de estas funciones vienen marcadas por la correspondiente rutina en código máquina.

ALMACENAMIENTO Y RECUPERACION DE RUTINAS EN CODIGO MAQUINA

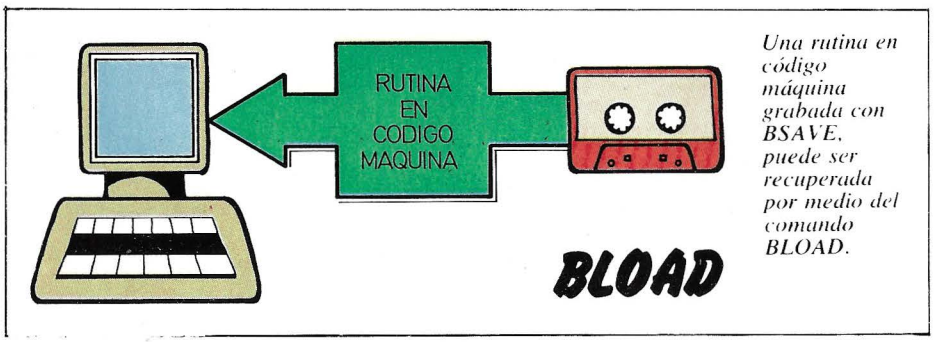
Las rutinas en código máquina se sitúan en una zona bien determinada de la memoria. Siguiendo los pasos comentados anteriormente (uso de CLEAR para abrir espacio en la memoria), existe una diferenciación entre el programa BASIC y las rutinas en código máquina. Aparte de su distinta naturaleza, residen en zonas de memoria separadas. Ello implica que el empleo del comando NEW alterará la zona de programa, pero no la de rutinas. Para borrar estas últimas es necesario restablecer el límite inicial por medio de CLEAR.
Otro problema lo constituye la introducción de las rutinas. Al principio del capítulo se esbozó un método basado en el uso del comando POKE; ejemplo que consistía en un programa "cargador". Este tomaba los códigos de la rutina que figuraban dentro de instrucciones DATA.
El referido programa puede servir para volver a crear la rutina: guardándolo en un soporte de almacenamiento externo, el programa cargador permanecerá siempre a disposición del programador.
Para volver a introducir la rutina habrá que cargar dicho programa (con LOAD) y posteriormente ejecutarlo (con RUN). Si la rutina en código máquina está asociada a un segundo programa BASIC, éste deberá cargarse a continuación.
La que sigue es una nueva versión, más completa, del programa cargador de rutinas en código máquina:

```
10 CLEAR <posición límite>
20 FOR I=<posición inicial> TO <posición
   final>
30 READ A
40 POKE I, A
50 NEXT I
60 DATA <códigos...>
70 NEW
```

El programa sintetiza todos los pasos necesarios para situar en memoria la rutina y el programa BASIC que la acompaña. La línea 10 fija el nuevo límite que reserva espacio para la rutina. Las líneas comprendidas entre la 20 y la 60 (ambas inclusive) cargan la rutina en código máquina. Por último, la línea 70 destruye el programa cargador, dejando todo listo para la introducción del programa BASIC que haga uso de la rutina creada.
Se podrían crear varios programas de este tipo, cargando cada uno de ellos una o varias rutinas en código máquina. Estos programas son fácilmente almacenables y recuperables con los conocidos comandos SAVE y LOAD.
Sin embargo, lo idóneo sería almacenar el segmento de memoria que contiene las susodichas rutinas. Pues bien, esto es factible. Para ello han de introducirse dos nuevos comandos BASIC. El primero es un comando capaz de almacenar parte del contenido de la memoria en un dispositivo externo; se trata de BSAVE (de Binary SAVE: almacenamiento binario). BSAVE almacena cualquier porción de la memoria tratándola como una lista de bytes. Así pues, la porción de memoria estipulada se guardará byte a byte en el soporte de almacenamiento externo.
Dicho comando incluye en su argumento, junto con el nombre del archivo a crear, dos datos que especifican la zona de me-



El comando BSAVE permite grabar rutinas en código máquina en un soporte de almacenamiento externo.



Una rutina en código máquina grabada con BSAVE, puede ser recuperada por medio del comando BLOAD.

TABLA DE CONVERSION

ORDENADOR	ACCESO A POSICIONES DE MEMORIA		ESPACIO DE MEMORIA		LLAMADA A RUTINAS EN CODIGO MAQUINA		ALMACENAMIENTO Y RECUPERACION DE RUTINAS EN C. M.		LOCALIZACION DE VARIABLES
	POKE	PEEK	FRE (0)	CLEAR	CALL	DEFUSR	BSAVE	BLOAD	VARPTR
APPLE II (APPLESOFT)	POKE	PEEK	FRE (0)	HIMEM	CALL/USR	—	BSAVE	BLOAD	—
APRICOT (M-BASIC)	POKE	PEEK	FRE (0)	CLEAR	CALL	DEFUSR	BSAVE	BLOAD	VARPTR
ATARI	POKE	PEEK	FRE (0)	—	USR	—	—	—	ADR (<var>)
CBM64	POKE	PEEK	FRE (0)	—	USR	—	—	—	—
DRAGON	POKE	PEEK	MEM	CLEAR	—	DEFUSR	—	—	VARPTR
EQUIPOS MSX	POKE	PEEK	FRE (0)	CLEAR	CALL	DEFUSR	BSAVE	BLOAD	VARPTR
HP-150	POKE	PEEK	FRE (0)	CLEAR	CALL	DEFUSR	BSAVE	BLOAD	VARPTR
IBM PC	POKE	PEEK	FRE (0)	CLEAR	CALL	DEFUSR	BSAVE	BLOAD	VARPTR
MPF	POKE	PEEK	FRE (0)	Sólo borra variables	—	—	—	—	—
NCR DM-V (MS-BASIC)	POKE	PEEK	FRE (0)	CLEAR	CALL	DEFUSR	—	—	VARPTR
NEW BRAIN	POKE	PEEK	FREE	Sólo borra variables	CALL	—	—	—	—
ORIC	POKE	PEEK	FRE (0)	HIMEM	CALL	DEFUSR	—	—	—
SHARP MZ-700 (MZ-BASIC)	POKE	PEEK	SIZE	LIMIT	USR	—	—	—	—
SINCLAIR QL	POKE	PEEK	—	Sólo borra variables	CALL	—	SBYTES	LBYTES	—
SPECTRAVIDEO	POKE	PEEK	FRE (0)	CLEAR	CALL	DEFUSR	BSAVE	BLOAD	VARPTR
ZX-SPECTRUM	POKE	PEEK	—	CLEAR	USR	—	SAVE "<nom>" CODE	LOAD "<nom>" CODE	—

moria que se almacenará. Su formato es el siguiente:

(Número de línea) BSAVE <nombre>, <dirección inicial>, <número de bytes>

Donde <nombre> debe especificar el nombre del archivo en el que va a residir la información, mientras que <dirección inicial> muestra el byte de omienzo de la zona de memoria a almacenar. El tercer dato aporta la longitud de la misma, medida en bytes.
Por ejemplo, para guardar una rutina en código máquina cuya dirección de comienzo es la 32500 y que agrupa a 150 bytes, basta con ejecutar la siguiente instrucción:

BSAVE "RUT1",32500,150

La referida información se deposita, en nuestro ejemplo, en un archivo cuyo nombre es "RUT1".

Las rutinas así almacenadas pueden ser devueltas a la memoria del ordenador por medio del comando BLOAD. El formato

BSAVE

Almacena una porción de la memoria en un soporte de almacenamiento externo.

FORMATO: BSAVE<nombre>,<inicio>,<bytes>
EJEMPLOS:

```
50 BSAVE "RUTINA",32500,500
85 BSAVE "CM",2000,20
```

BLOAD

Carga en memoria una rutina en código máquina o una zona de memoria cualquiera.

FORMATO: BLOAD <nombre>[,<posición inicial>]
EJEMPLOS:
20 BLOAD "RUTINA"
70 BLOAD "CM",33000

VARPTR

Proporciona la posición de memoria en la que se almacena el contenido de una variable.

FORMATO: VARPTR (<variable>)
EJEMPLOS:
20 LET A=VARPTR(BL$)
70 PRINT VARPTR(A)

general de este nuevo comando es el siguiente:

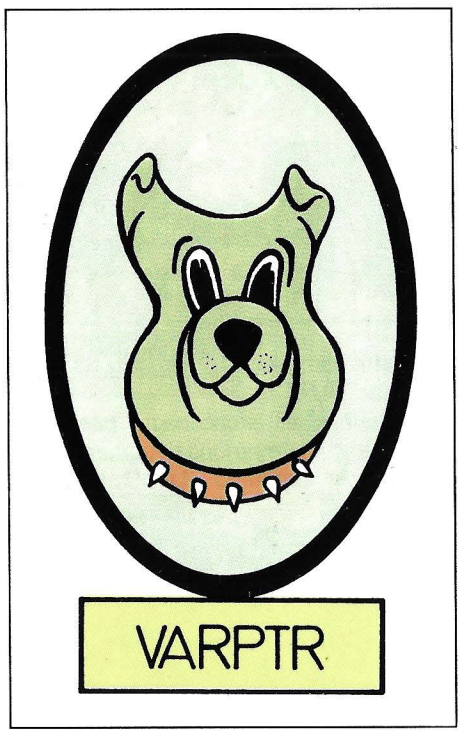
(Número de línea) BLOAD <nombre> [,<dirección inicial>]

Como en el caso anterior, <nombre> indica el nombre del archivo que, en este caso, se desea cargar en memoria. El segundo argumento señala, opcionalmente, la nueva dirección de comienzo. Este último dato permite ubicar la misma rutina en una posición diferente. Si no se especifica la dirección de comienzo, se cargará en el mismo lugar del que la rutina fue leída y grabada en la unidad externa con BSAVE.

Como se observa, el modo de funcionamiento de BSAVE y BLOAD es muy parecido al de SAVE y LOAD. Utilizando estos nuevos comandos para el almacenamiento y recuperación de rutinas en código máquina, se puede crear con comodidad una biblioteca de rutinas de esta categoría.

El programa que se encargaba de situar en memoria la rutina en código máquina se simplifica en gran medida con el uso de BLOAD. Este sería el aspecto del nuevo programa cargador:

```
10 CLEAR <posición límite>
20 BLOAD <nombre del archivo>
70 NEW
```



La función VARPTR sirve para localizar la posición de memoria en la que se almacena el contenido de una variable dada.

LOCALIZACION DE VARIABLES

En algunas ocasiones es interesante conocer la posición de memoria en la que se almacena el contenido de una variable. Se ha comentado que las zonas de memoria se distribuyen de forma distinta, dependiendo del diseño del propio microodenador. Tal distribución puede encontrarse en manuales o libros avanzados que ofrezca el fabricante del aparato. En todo caso, la posición que ocupa una variable específica, puede no ser fija. Puede, incluso, que la distribución de memoria se vea alterada con la conexión de periféricos. Con todo ello se hace difícil la localización de una variable concreta.

Cuando se está trabajando con programas mixtos BASIC/código máquina, es habitual el paso de datos de unas rutinas a otras. Tanto en los argumentos de CALL como de USR se pueden especificar parámetros de entrada; parámetros que transfieren datos (constantes o variables) de la zona en BASIC a la rutina en código máquina.

Este transporte de datos puede realizarse también especificando la posición de memoria donde se encuentra el dato en cuestión; para lo cual es necesario conocerla.

El modo de averiguar la posición en la que se sitúa el contenido de una variable implica el uso de una nueva función BASIC: VARPTR, cuyo formato ofrece el siguiente aspecto:

(Número de línea) LET <variable numérica>=VARPTR (<nombre de variable>)

VARPTR es una función, lo cual significa que debe ir dentro de una sentencia de asignación, o como argumento de un comando u otra función. En el formato se ha situado dentro de una sentencia de asignación, por lo que el dato proporcionado por VARPTR se depositará en la variable que acompaña a LET.

Dentro del argumento de VARPTR se indica el nombre de la variable a localizar, esto es: de la variable cuya posición de memoria se desea conocer.

La función VARPTR también puede ser utilizada para pasar una posición de memoria como dato destinado a una rutina en código máquina. En el ejemplo que se muestra a continuación se manda como parámetro la posición en la que se almacena la variable A$:

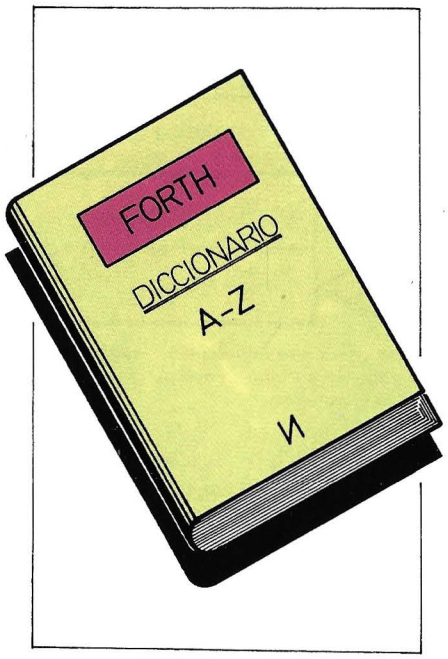
```
150 PRINT USR5(VARPTR(A$))
```

Al igual que los restantes comandos y funciones introducidos en este capítulo, VARPTR será de especial interés para programadores experimentados. En general, el uso del código máquina resulta excesivamente complejo para los principiantes. Para utilizarlo con éxito es preciso conocer a fondo la máquina que se está empleando. Y ello no es tarea fácil.

Forth (4)

Definición y manejo de las palabras del Forth

Como ya se explicó en el primer capítulo de esta serie, el FORTH es un lenguaje que no trabaja ni en forma compilada ni interpretada. Realmente, opera de acuerdo a dos modalidades cuya intervención depende de la tarea en curso; éstos son ejecución inmediata y compilación.
Hasta el momento, todos los ejemplos incluidos en los artículos precedentes correspondían a la modalidad de trabajo en ejecución inmediata. Veamos ahora como también es posible operar en modo compilación.
En modo de ejecución inmediata, cada palabra introducida en la máquina es buscada en el diccionario y ejecutada inmediatamente. El modo compilación se emplea para proceder a la definición de nuevas palabras FORTH. En efecto, ahora la mayor parte de las palabras que se introducen no son ejecutadas; sino que, a través de la definición de la nueva palabra, se introduce en el diccionario una referencia a las palabras elementales que la componen. Tras la compilación, la nueva palabra pasa a formar parte del diccionario y puede ser invocada en cualquier momento; ya sea de forma inmediata, o bien por efecto de una referencia a la misma en otra definición (esto es, en modo de compilación).



El diccionario FORTH está compuesto por el conjunto de palabras utilizables en las tareas de programación y sus correspondientes definiciones.

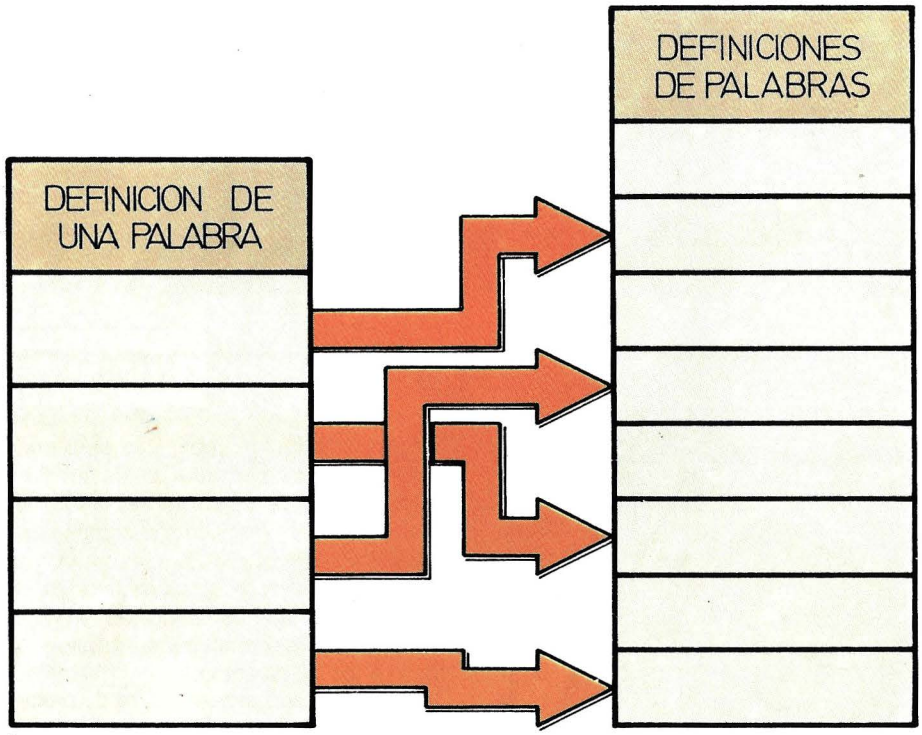
COMO DEFINIR NUEVAS PALABRAS

La definición de palabras es el método que permite realizar la programación, propiamente dicha, en lenguaje FORTH. Gracias a la estructura del diccionario que hace posible emplear palabras previamente definidas para construir las nuevas palabras, el diccionario se ve ampliado, tanto en extensión como en potencia, a medida que se va trabajando con este lenguaje.
Para definir una nueva palabra es necesario proporcionar al ordenador una definición exacta de su significado. Para que éste sepa que el programador se dispone a darle la definición de una palabra, es preciso advertirlo mediante una orden representada por la palabra ":".
Una vez que el ordenador sabe que se encuentra en disposición de definir una palabra, el siguiente paso será comunicarle de qué palabra se trata; es decir: cuál es el nombre de la nueva palabra. Dicho nombre se escribirá a continuación. Es muy importante saber que el referido nombre puede contener cualquier símbolo que forme parte del repertorio de caracteres del ordenador, exceptuando a los símbolos gráficos y espacios en blanco. La longitud del nombre varía de una a otra versión del lenguaje FORTH, si bien, por lo general, suele situarse en torno a los 30 caracteres.
Una vez aportado el nombre, ha de procederse ya a la definición de la nueva palabra propiamente dicha; para lo cual el programador puede basarse tanto en las palabras que aporta el traductor FORTH, como en las que él hubiera definido previamente. El final de la definición de la nueva palabra ha de indicarse al ordenador mediante la palabra delimitadora ";".
Veamos un ejemplo de definición de palabras FORTH.
En primer lugar, se trata de definir una palabra que permita sumar dos unidades al número situado en la cima de la pila:

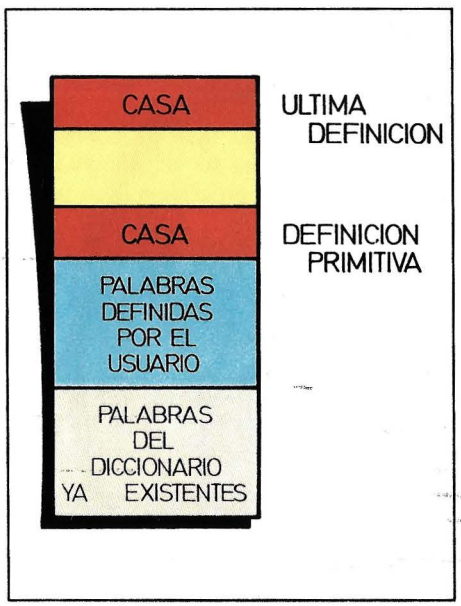
```
: PALABRA <CR>
2 + <CR>
; <CR>
```

Tras accionar por tercera vez la tecla RETURN (CR, o retorno de carro), el ordenador responderá con el siguiente mensaje por pantalla:

```
: PALABRA 2 + ; OK
■
```



El modo «compilación» se emplea en el FORTH para proceder a la definición de nuevas palabras. Por efecto de la definición de una nueva palabra, se introduce en el diccionario FORTH una referencia a las palabras elementales que la componen.



Al modificar la definición de una palabra o proceder a la nueva definición de la misma, ésta se almacena por encima de la antigua. Las nuevas referencias a dicha palabra afectarán a la definición más reciente de la misma. Sin embargo, las antiguas palabras seguirán existiendo. Para actualizar estas definiciones y eliminar las antiguas, es necesario emplear la palabra REDEFINE.

Con lo cual, la nueva palabra queda ya incluida en el diccionario y puede ser utilizada a voluntad.

Para escribir la definición de una palabra no sólo puede emplearse el método aplicado en el ejemplo anterior. Existe otra posibilidad alternativa que consiste en escribir la definición a modo de una serie de palabras, una tras otra, separadas por espacios. En el caso del ejemplo descrito, la definición también podría haberse realizado como sigue:

```
: PALABRA 2 + ; <CR>
```

Ambos métodos de definición no muestran, en principio, ninguna diferencia aparente; en pantalla obtendremos el mismo mensaje que en el caso anterior.

Veamos a continuación cómo puede utilizarse la palabra definida. De trabajar en modo inmediato, es suficiente con introducir el nombre de la palabra en cuestión para que ésta se ejecute. Para ello, en primer lugar, se depositará en la pila un valor, y tras ello, ordenaremos a la máquina que visualice en la pantalla el número situado en la cima de la pila:

```
3 PALABRA . <CR>
```

La respuesta será:

```
3 PALABRA . 5 OK
```

Para trabajar con una palabra en modo compilación es necesario definir una nueva palabra que la contenga. En nuestro caso añadiremos otra palabra al diccionario FORTH. Esta nueva palabra realizará la misma función que la anterior, aunque incluiremos en su definición la orden de salida por pantalla del resultado. El nombre otorgado a la nueva palabra a definir será, por ejemplo, el de 8A. Como se observa, el nombre comienza con un número. Este es un hecho un tanto extraño para las personas acostumbradas a otros lenguajes de programación; si bien, en FORTH ello resulta de lo más habitual de hecho. El nombre de una palabra FORTH puede constar, si así se desea, únicamente de números o incluso de símbolos de puntuación.

```
: 8A <CR>
PALABRA <CR>
2 + . <CR>
; <CR>
```

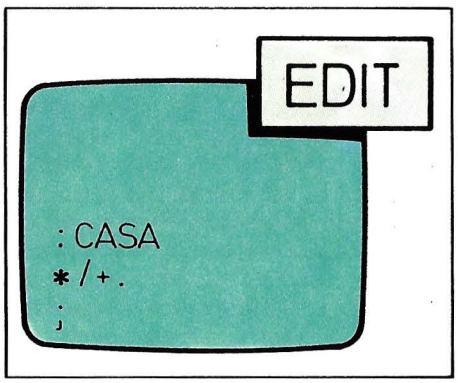
Tras introducir la nueva palabra definida, la pantalla mostrará el siguiente mensaje:

```
: 8A PALABRA 2 + . ; OK
```

Con ello, el ordenador informa que la nueva palabra ha pasado a formar parte del diccionario, y puede ya utilizarse libremente:

```
3 8A <CR>
3 8A 7 OK
```

Por supuesto, las palabras definibles pueden ser mucho más complejas que la contemplada en el ejemplo. En sucesivos ejemplos se verán definiciones de palabras cada vez más evolucionadas, a medida que se avance en el conocimiento de los bucles, decisiones y otros tipos de estructuras.
Llegados ya a este punto, resulta interesante poder conocer las palabras que existen en el diccionario que estamos utilizando. El propio lenguaje FORTH incluye una palabra al efecto capaz de mostrar por pantalla todas las palabras existentes en el diccionario. Se trata de la palabra VLIST. Una vez introducida esta palabra, y tras pulsar la tecla de retorno de carro, aparecerán en la pantalla todas las palabras que componen el diccionario.



La palabra EDIT traslada la definición de una palabra al buffer de entrada, permitiendo con ello realizar en la misma las modificaciones que sean necesarias.

También es conveniente saber cómo es posible incluir dentro de una palabra comentarios que más tarde servirán como orientación a la hora de efectuar posibles modificaciones. Sencillamente, se escribirá el comentario que se desee encerrado entre paréntesis. Por ejemplo:

```
: MENSA <CR>
( GENERACION DE UN MENSAJE ) <CR>
." ESTE ES EL MENSAJE" <CR>
; <CR>
```

La respuesta en pantalla coincidirá con:

```
: MENSA ( GENERACION DE UN MENSAJE )
  ." ESTE ES EL MENSAJE" ; OK
```

Desde luego, podemos ejecutar la nueva palabra definida de forma inmediata:

```
MENSA <CR>
ESTE ES EL MENSAJE OK
■
```

FORGET es la palabra que se utiliza para eliminar selectivamente palabras del diccionario.

CAMBIANDO DEFINICIONES DE PALABRAS

La definición de una nueva palabra no es algo inamovible, sino que una palabra puede ser borrada del diccionario o cambiada cuando así se desee; aunque, por supuesto, hay que tener en cuenta el hecho de que pueden existir otras palabras que hagan referencia a la palabra sujeta a modificación.
En una determinada situación, puede resultar interesante conocer cuál es la definición de una cualquiera de las palabras que existen en el diccionario. Para ello basta con ejecutar la palabra LIST, cuyo formato coincide con el siguiente:

```
LIST <palabra> <CR>
```

Donde <palabra> es la palabra cuya definición se desea visualizar en la pantalla. Por ejemplo:

```
LIST PALABRA <CR>

: PALABRA
2 +
;
 OK
```

También puede resultar de interés borrar algunas de las palabras que forman parte del diccionario. Para ello también existe una palabra FORTH especializada: se trata de FORGET, cuya formulación debe ofrecer el siguiente aspecto:

```
FORGET <palabra> <CR>
```

En donde <palabra> es la palabra que deseamos que desaparezca del diccionario. Evidentemente no es posible hacer desaparecer ninguna de las palabras incluidas originalmente en el propio traductor de FORTH.
Veamos un ejemplo:

```
FORGET MENSAJE <CR>
```

La respuesta del ordenador será:

```
FORGET MENSAJE OK
```

Si a continuación se intenta visualizar la definición de la palabra borrada, la respuesta de la máquina coincidirá con un mensaje de error (LIST ERROR).

TABLA DE ORDENES FORTH

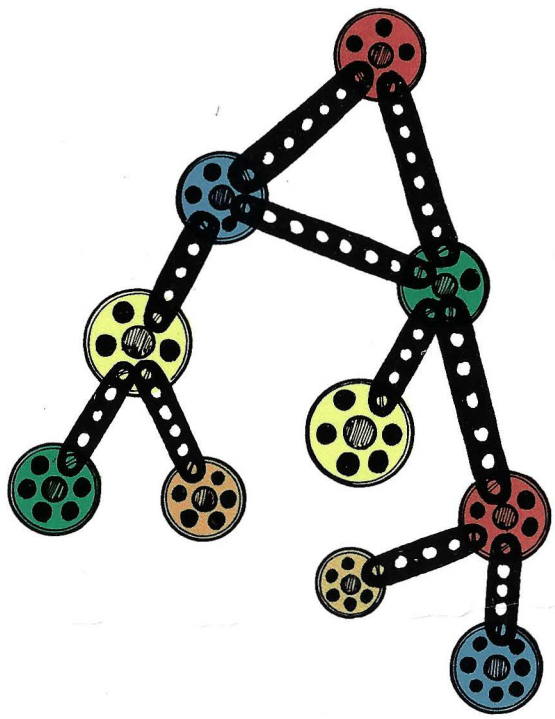
PALABRA	DESCRIPCION	TIPO
:	Indica el comienzo de la definición de una palabra.	Palabra de definición.
;	Indica el fin de la definición de una palabra.	Palabra de definición.
LIST <palabra>	Lista la definición de una palabra.	Tratamiento de palabras.
FORGET <palabra>	Borra una palabra del diccionario.	Tratamiento de palabras.
EDIT <palabra>	Edita la definición de una palabra.	Tratamiento de palabras.
REDEFINE <palabra>	Actualiza el diccionario y las referencias a una palabra.	Tratamiento de palabras.

También es posible introducir modificaciones en la definición de una palabra ya existente. Al respecto, basta con llevar la definición de dicha palabra al buffer de entrada (hay ordenadores en los que es suficiente con que la definición se encuentre en la pantalla). Para conseguirlo, hay que hacer uso de la palabra EDIT, cuyo formato es el siguiente:

EDIT <palabra> <CR>

Donde <palabra> es la palabra que se desea trasladar al buffer de entrada en orden a realizar en ella las modificaciones pertinentes.
Veamos a continuación un ejemplo ilustrativo de su puesta en práctica. Se empezará definiendo una palabra que eleve un número al cuadrado.

```
: CUADRADO <CR>
* . <CR>
; <CR>
```



En el lenguaje FORTH, las palabras se van definiendo de una forma jerárquica; la definición de una palabra puede edificarse en base a definiciones de palabras ya existentes.

En esta definición hay un error, ya que lo lógico sería introducir el número del que se desea obtener el cuadrado, y que el ordenador entregara la respuesta; algo semejante a:

```
3 CUADRADO <CR>
```

Sin embargo, y lamentablemente, el ordenador entrega por toda respuesta un mensaje de error. Ello se debe a que se ha ordenado a la máquina que multiplique dos números; estos serán los situados más cerca de la cima de la pila. Como quiera que la definición tan sólo deposita uno de ellos en la pila, el ordenador no es capaz de encontrar el segundo número. Para evitar tal situación, es preciso incluir en la definición una palabra que duplique la cima de la pila (la instrucción DUP, por ejemplo).
Para proceder a la modificación de la palabra CUADRADO, se empezará escribiendo la instrucción:

```
EDIT CUADRADO <CR>
```

Acto seguido, es posible ya acondicionar la definición de la palabra hasta dejarla de la siguiente forma:

```
: CUADRADO
DUP * .
;
```

No obstante, si ahora se hace uso de la palabra VLIST, el programador descubrirá que existen dos definiciones de la misma palabra.
En principio, y en el caso de operar en el modo inmediato, tal situación no plantea problema alguno. No obstante, si antes de realizar dicha modificación se hubiera definido otra palabra —por ejemplo TARA— que hiciera uso de la palabra CUADRADO, tras efectuar la modificación se observaría que la palabra TARA sigue utilizando la primitiva definición de la palabra CUADRADO. Para actualizar esta referencia y, en consecuencia, eliminar las definiciones redundantes, se hace necesario el empleo de la palabra REDEFINE. Esta actualiza las versiones de una palabra y las referencias que otras palabras hagan a la misma. Su formato es el siguiente:

```
REDEFINE <palabra> <CR>
```

A partir de los conocimientos expuestos en los párrafos anteriores, es posible ya ir pensando en la elaboración de los primeros programas en lenguaje FORTH.

OASIS (1)

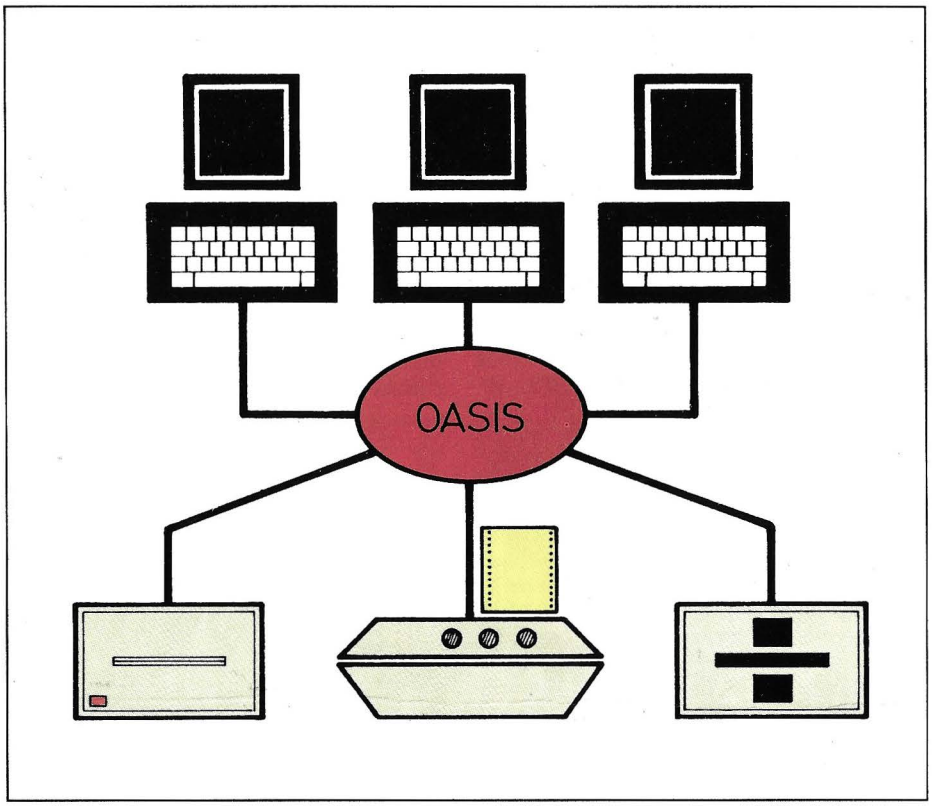
La potencia al alcance de los microordenadores

Tal y como se ha señalado al hacer mención a ciertos sistemas operativos, el éxito de los mismos está fundado en su capacidad de respuesta a una determinada solicitud. Así, el éxito del sistema operativo MS/DOS tuvo su base en que respondía a las necesidades establecidas por IBM cuando esta compañía consideró el lanzamiento de su ordenador personal: el IBM-PC. Para considerar el éxito del sistema operativo OASIS es conveniente dar un repaso al campo actual de usuarios de equipos informáticos. Por un lado se encuentran las grandes compañías, usuarios tradicionales de los mismos, normalmente con una estructura de uso plenamente establecida y definida, así como con personal especializado en informática. Por otro lado están los pequeños usuarios de equipos informáticos, muchos de los cuales no han considerado nunca la adquisición de un ordenador y que, potencialmente, dan cuerpo a una gran parte del mercado informático. Estos pequeños usuarios constituyen, como grupo, una mezcolanza heterogénea en cuanto a usos y necesidades; si bien, en su centro destaca básicamente la necesidad de un equipo de elevada flexibilidad y prestaciones aceptables.

Este tipo de usuarios no puede acceder, normalmente por razones monetarias, a equipos informáticos de gran capacidad, como pueden ser minis o superminis. Igualmente, el volumen de información que gestionan no es tan grande que requiera uno de estos equipos, aunque ello no significa que el tipo de gestión que realizan con la información sea distinto al que lleva a cabo una gran empresa. Evidentemente, el tratamiento de una factura en su forma intrínseca es igual en una pequeña tienda que en una gran empresa. Lo expresado indica que, en la mayoría de los casos, el referido segmento de potenciales usuarios se inclinará por la alternativa que supone el microordenador, ya sea de mayor o menor capacidad.

El sistema operativo OASIS: prestaciones de gran equipo sobre el hardware de un microordenador.

Ahora cabe hacerse la siguiente pregunta: ¿Qué sistema operativo elegir? En muchos casos el entorno ha de ser multiusuario, con lo cual han de estar previstos métodos de seguridad y control normalmente no presentes en ciertos sistemas operativos. En otros casos, será necesario que el equipo sea multitarea, con las servidumbres exigidas por esta modalidad de operación. En cualquier caso, parece lógico que el sistema operativo debe ser independiente del equipo.



El OASIS es un sistema operativo que permite la operación simultánea de varios usuarios sobre el ordenador y sobre distintos periféricos asociados al mismo.

APARECE EL OASIS

En el año 1977 aparece el sistema operativo OASIS, creado por la firma americana Phase One. Un sistema operativo destinado a microordenadores basados en el microprocesador de 8 bits Z 80 A. Casi de inmediato, este sistema operativo empezó a ser considerado entre los usuarios de equipos de esta categoría, ya que su filosofía de diseño, según establecieron sus propios creadores, estaba específicamente adaptado a las necesidades de estos usuarios.

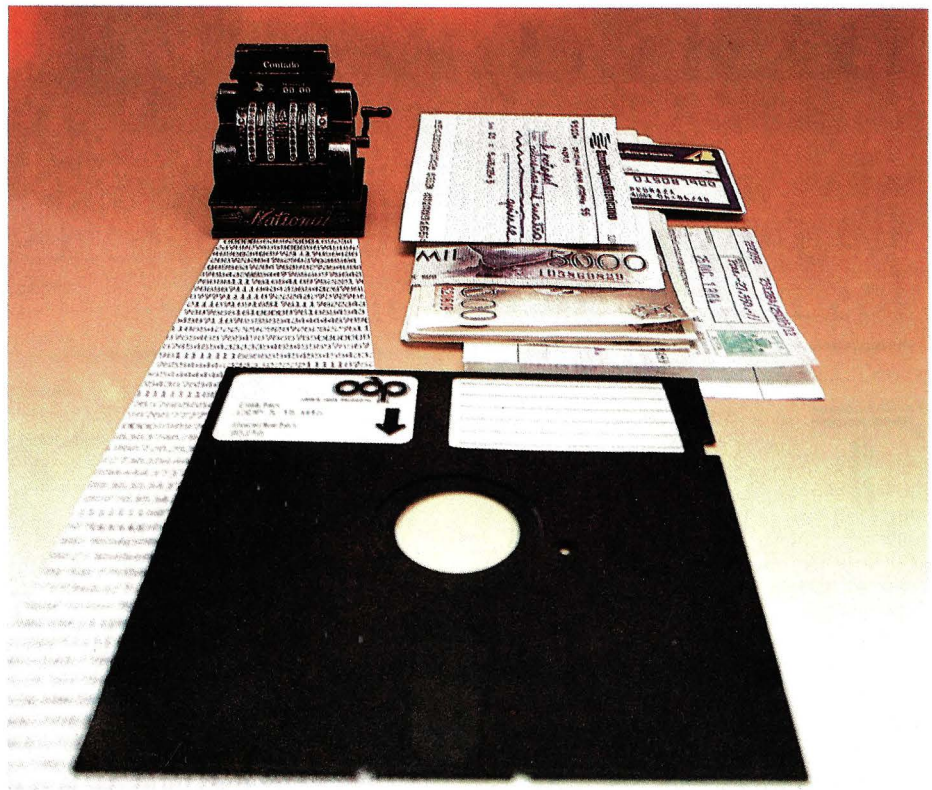
La filosofía de diseño de OASIS se resume en los siguientes puntos:

1. El sistema operativo ha de estar concebido para equipar a microordenadores, aunque ofreciendo características propias de miniordenadores o mainframes.
2. El sistema operativo ha de poder trabajar sobre equipos de distintos fabricantes; esto es: debe ser independiente de la máquina sobre la que opera.
3. Debe estar orientado a usuarios sin experiencia informática previa.

De estos puntos, el que ha resultado más difícil de conseguir de manera consistente ha sido el segundo: la independencia de la máquina. Sin embargo, su consecución representa para el usuario de OASIS una enorme ventaja, ya que puede ejecutar sus programas en distintos ordenadores sin tener que efectuar cambios en los mismos.

También resulta complicada la consecución de un sistema operativo "amigable" para el usuario, algo que no es particularmente común en los sistemas operativos clásicos como el CP/M o el MS/DOS. En el caso del OASIS ello se consigue haciendo consistentes todas las funciones del sistema operativo, dando la posibilidad de activar una orden de ayuda ("help") en cada comando, y empleando palabras del vocabulario inglés convencional para todos los comandos.

Parece obvio que este sistema operativo ha sido pensado teniendo en mente a todos aquellos usuarios de pequeños equipos que, sin embargo, requieren prestaciones elevadas.



El sistema operativo OASIS está especialmente concebido para equipar a microordenadores orientados a mecanizar actividades de gestión en el marco de la empresa.

Tal y como señalan sus creadores, el sistema operativo OASIS está orientado a operar en el ámbito de la empresa, controlando el tratamiento de aplicaciones de gestión.

EL OASIS DESDE DENTRO

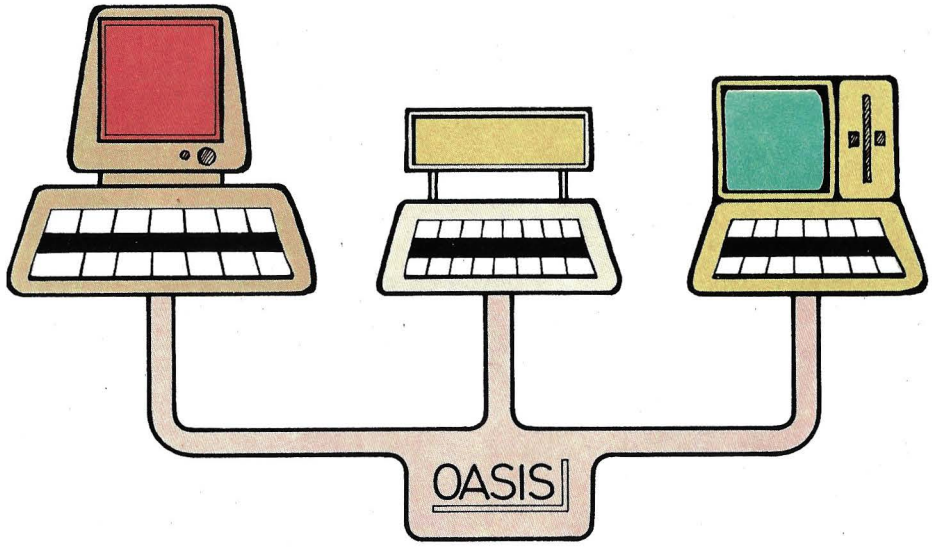
Como se ha señalado anteriormente, las primeras versiones de este sistema operativo (hoy existen ya versiones para equipos de 16 bits) estaban preparadas para trabajar sobre microordenadores basados en un microprocesador de 8 bits; permitiendo la operación en régimen multiusuario así como en tiempo compartido.

Sus necesidades, por lo que respecta a memoria residente exigida al ordenador que lo soporta, se concretan en un mínimo de 64 Kbytes; si bien, puede llegar a soportar hasta 784 Kbytes. El núcleo del sistema requiere un total de 16 Kbytes, aunque dependiendo de la versión puede expandirse hasta 32 Kbytes, con lo cual se eliminan alguno de los "overlays" más corrientes, haciéndolos residentes en memoria.

Internamente, el sistema operativo se divide en las tres zonas básicas que se describen a continuación:

- Núcleo

Se denomina SYSTEM NUCLEUS. Durante el proceso de carga es lo primero a que se accede, y está compuesto por un conjunto de subrutinas de propósito general que proporcionan y establecen la integración del sistema en un bloque común.
Asociados a este núcleo, aunque físicamente separados, se encuentran los programas que interaccionan con los controladores de periféricos. A través de estos programas el sistema operativo interacciona con los mismos y gestiona las operaciones de entrada/salida, de inicialización de periféricos y de detección y manejo de errores. La concepción de estos programas presenta al usuario diferentes posibilidades, como es el hecho de permitir desconectar un periférico en el caso de que tenga una avería, sin efectos perniciosos para el sistema. O bien cambiar el nombre lógico de un determinado periférico por medio de un comando; por ejemplo, de tal forma que si un programa especifica dirigir una información a la pantalla, ésta puede derivarse hacia la impresora sin necesidad de modificar dicho programa; para ello basta tan sólo con asignar a ésta el nombre lógico dado a la pantalla en el programa.



La estructura del sistema operativo OASIS permite su adaptación a diversos tipos de ordenadores, con la condición de que compartan el mismo microprocesador en su unidad central.

Una ventaja adicional reside en el hecho de que permite la adopción de periféricos de diferentes fabricantes, al actuar estos programas como si se tratara de verdade-

La protección del software

En los últimos años el campo informático se ha ido abriendo paulatinamente a más usuarios; en gran medida, a través de la senda aportada por los microprocesadores. Cada vez es más amplia la circulación de software y son más y más las personas que tienen acceso a la información.
Una de las facetas del software, de indudable importancia, es la relativa a la integridad e intimidad de la información. Afrontar este problema lleva inevitablemente a situaciones de difícil resolución. Por una parte, el objetivo primordial de la informática es ofrecer información con la mayor sencillez posible para su fácil entendimiento; mientras que por otra hay que mantener una cierta integridad de la información, para que sea veraz, y además hay que controlar el acceso a la misma: de poco valdría tener una gran cantidad de datos si de ellos no se extrayeran conclusiones correctas.
A la hora de acometer la resolución de este difícil tema, cabe empezar distinguiendo entre dos tipos de información: la información de tipo general, que ha de ser de fácil acceso y que va dirigida a cualquier usuario; y la información confidencial, destinada a personas cuyo cometido las obligue a tomar decisiones importantes o comprometidas. Esta última debe quedar al amparo de miradas curiosas, dada su vital importancia y, por lo tanto, debe estar arropada con todas las posibles medidas de protección.
Dentro de estas medidas de protección se encuentran las llamadas palabras clave o "passwords" que impiden el acceso a la información a aquellas personas que las desconocen. Actúan, realmente, como una especie de llave de software, aunque su fiabilidad no es muy alta puesto que no es muy difícil su divulgación entre personal no autorizado.

La codificación de la información también se utiliza para estos fines. En tal caso, la técnica a aplicar se basa en tablas de sustitución de unos caracteres por otros; con ello el acceso a la información cifrada resulta más difícil al ser más complejo el método de enmascaramiento.
Aumentando la dificultad de acceso a la información, cabe hablar del cifrado que se fundamenta en la mezcla de la información, sometiendo el texto a una serie de operaciones matemáticas previstas en un algoritmo de naturaleza pseudoaleatoria; la operación de descifrado resulta ahora prácticamente imposible si no se deduce el algoritmo correspondiente.
Otro método frecuente es el acceso con derechos restringidos a los ficheros que almacenan información, de tal manera que no todos los usuarios puedan leer o escribir en un determinado fichero.
Aun aplicando toda esta serie de medidas de protección, nunca es posible garantizar una plena y total integridad de la información. Siempre existe la técnica opuesta a la utilizada; en todo caso, cuantas más barreras de protección se introduzcan para que los datos no sean accesibles indiscriminadamente, mayor será el nivel de seguridad logrado.

A pesar de su indudable potencia operativa, el OASIS se caracteriza por su reducida dificultad de uso, hasta el punto de que resulta adecuado para usuarios con escasa formación informática.

ros adaptadores de los periféricos al sistema.
En el caso de que se tratara de la versión multiusuario del sistema operativo OASIS (la más usual y difundida) el núcleo es el encargado de controlar la gestión de memoria así como la compartición de recursos entre los diversos usuarios.

- CSI

El CSI es la abreviatura de lo que se denomina "Command String Interpreter". Esta zona actúa como controlador de las operaciones de acceso al sistema y a los programas de usuario.

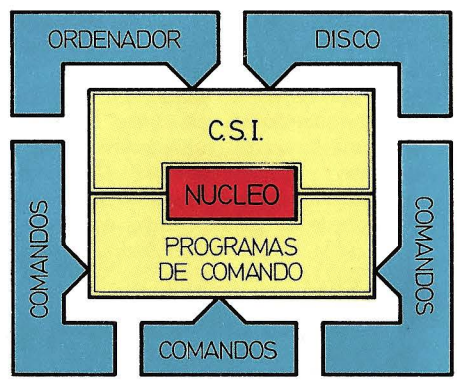
Una vez lanzado el sistema, la función del CSI es la de buscar en el disco que lo contiene los comandos introducidos por el usuario, pasando el control a los mismos si llega a localizarlos, o bien emitiendo un mensaje de error en el caso contrario.

Similar tarea lleva a cabo cuando el usuario especifica el acceso a un programa: efectúa la búsqueda en las diversas unidades de disco y, de localizarlo, lo carga en memoria y arranca su ejecución.

Esencialmente, la combinación del núcleo, junto con los programas de control de periféricos y el CSI, constituye el eje que soporta todas las operaciones de cálculo y entrada/salida.

- Programas

Bajo este apelativo tan simple se ocultan los medios de proceso de información que proporciona el sistema operativo OASIS. Estos programas son esencialmente los comandos del sistema además de los procesadores de lenguajes, como el MACRO Assembler o el BASIC, así como el lenguaje de control de procedimientos EXEC. Este último es una poderosa herramienta que opera a través de los programas y comandos del sistema. Un programa en este lenguaje de control permite detallar toda una serie de tareas que debe llevar a cabo el ordenador, sin necesidad de que el usuario se vea obligado a especificarlas una a una. Normalmente, puede emplearse este procedimiento para ordenar que se ejecuten uno tras otro varios programas, en cuyo caso las variables empleadas por estos pueden almacenarse en un fichero con registros de hasta 512 bytes, incluyendo el fichero hasta 255 registros. A través de programas de este tipo pueden llevarse a cabo tareas como enviar mensajes específicos o generales a los restantes usuarios del sistema, en el caso de operar como sistema multiusuario.
Una característica a señalar relacionada con este último punto es el hecho de que si un usuario no puede visualizar un mensaje en un momento determinado, este mensaje se almacena y cuando es posible su visualización se presenta al usuario, borrándose a continuación.

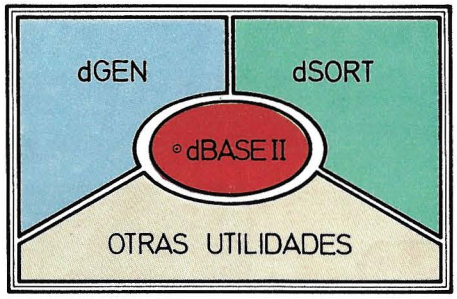


Estructura básica del sistema operativo OASIS.

dBASE II (y 3)

Sesión de trabajo con el dBASE II

Para concluir el estudio de la base de datos relacional dBASE II, se describirán en este capítulo dos utilidades de gran importancia: el generador de programas dGEN y el programa de clasificación dSORT. Ambos pueden considerarse como programas auxiliares de dBASE II y, por lo tanto, no incluidos en el núcleo principal. No obstante su adquisición se realiza en un único paquete y proporcionan una notable ayuda para la explotación de la base de datos. También como síntesis de todos los comandos descritos, realizaremos una sesión de trabajo que incluirá las principales operaciones soportadas por el paquete dBASE II.



El conjunto de opciones aportadas por el dBASE II se puede dividir en cuatro grandes grupos: un núcleo central de comandos (dBASE II), un generador de programas (dGEN), un programa de clasificación (dSORT) y otras utilidades.

- dGEN

El programa dGEN permite generar de forma automática nuevos programas que actuarán sobre la base de datos dBASE II. Para generar un programa el usuario debe limitarse a contestar las preguntas que dGEN le formulará.
Existe un procedimiento interactivo para la utilización de este programa de utilidad, el cual es invocado tecleando sencillamente la palabra dBASE, dGEN o DO dGEN; de inmediato, aparecerá un menú en la pantalla ofreciendo cinco posibilidades distintas entre las que el usuario debe elegir:

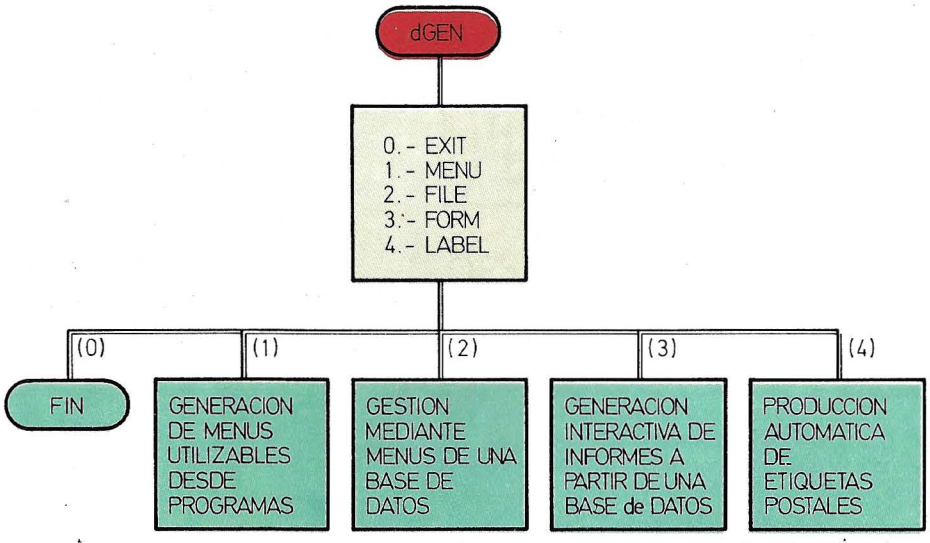
0 – EXIT
Mediante esta opción el usuario indica que desea dar por terminado el trabajo con el programa dGEN.

1 – MENU generator
Puede utilizarse para generar menús explotables por los programas que actuarán sobre dBASE II. De esta forma, los usuarios de dichos programas dispondrán de un sencillo mecanismo para la introducción de datos, la recuperación de resultados y la gestión del programa en ejecución.

2 – FILE generator
Como ya vimos en el primer capítulo dedicado a este programa, los comandos elementales para la actualización y manejo de las bases de datos son de tipo interactivo. Mediante el FILE generator se pueden crear un menú principal y una serie de programas para la gestión sofisticada de la información residente en una base de datos; un menú cuyas opciones permitirán ver, añadir, editar o comprimir los registros.

3 – REPORT FORM generator
Al igual que en el caso anterior, mediante esta opción se puede preparar un menú y un conjunto de programas para la obtención de informes a partir de una base de datos. El menú generado es similar, en su estructura y preguntas interactivas, al asociado al comando REPORT ya estudiado.

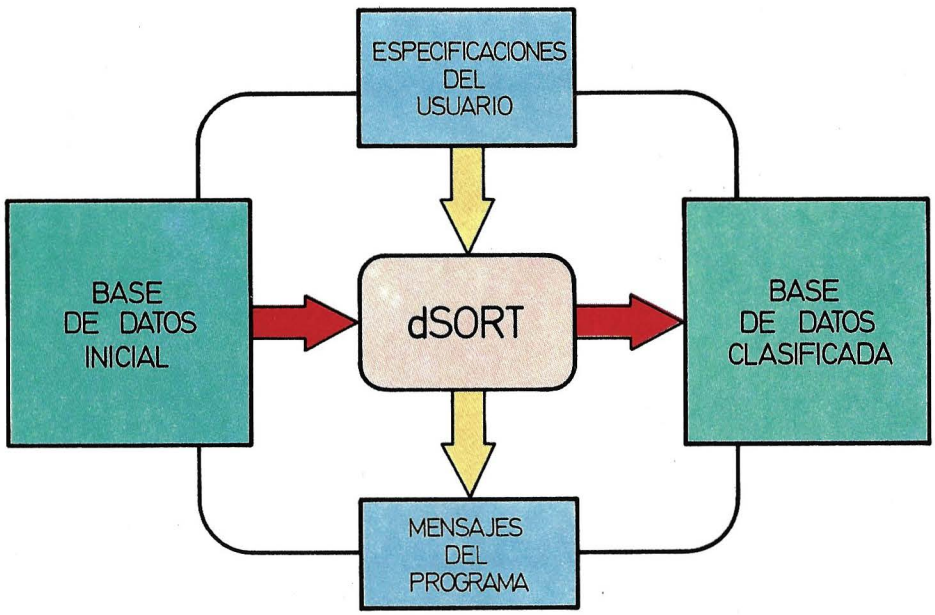
4 – LABEL generator
En muchas ocasiones es preciso mandar correspondencia a un conjunto más o menos numeroso de destinatarios. Las cartas a enviar pueden editarse y producirse ya sea con un procesador de textos, o bien mecanografiándolas manualmente. La opción LABEL generator, incluida en el programa dGEN, está diseñada para la generación automática de un programa que se



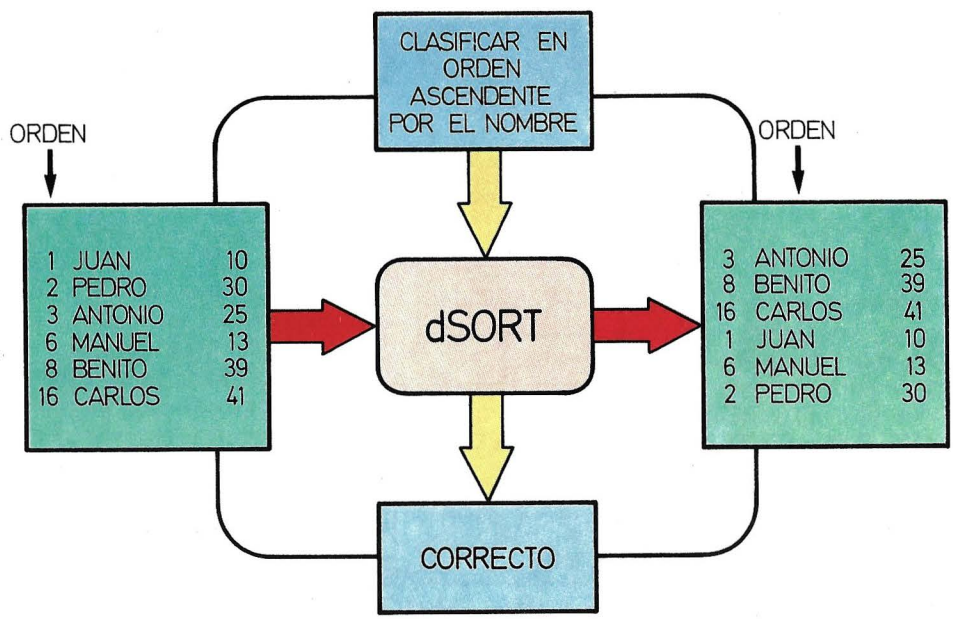
Cuando se invoca al programa dGEN aparece un menú en la pantalla del ordenador, el cual permite al usuario elegir entre cinco opciones.

encargará de imprimir etiquetas autoadhesivas para ser pegadas en los sobres que contendrán, las cartas a enviar. Evidentemente, para explotar esta opción resulta indispensable disponer en una base de datos de las direcciones que deben imprimirse en las etiquetas.

En resumen, podemos afirmar que el programa dGEN es una utilidad destinada a la generación automática de programas. Sin embargo no es un auténtico generador de programas, ya que las posibilidades que se ofrecen al usuario se reducen a las comentadas anteriormente, mientras que un auténtico generador debe ser capaz de producir cualquier tipo de programa. En cualquier caso, resulta notorio que un soft ware de aplicación destinado a su explotación en ordenadores personales, como es dBASE II, haga una incursión en este apasionante mundo de la inteligencia Artificial (evidentemente un generador de programas puede ser considerado como un programa inteligente).



El programa dSORT necesita como entrada una base de datos inicial y ciertas especificaciones sobre los parámetros de clasificación. Produce como salida una base de datos clasificada y mensajes para el usuario.



Ejemplo de clasificación mediante el programa dSORT.

- dSORT

Los datos contenidos en una base de datos tienen una clasificación inicial que coincide con su número de orden. En muchos casos, a la hora de producir un informe, será necesario alterar dicha clasificación, de forma que el orden de los registros se establezca de acuerdo a alguno de los datos contenidos en la base.

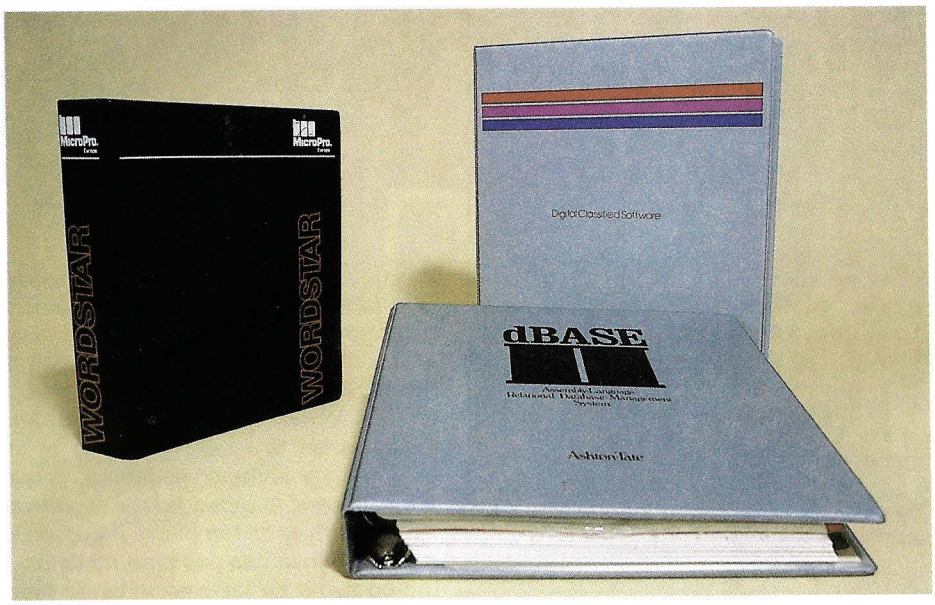
dBASE II ofrece una utilidad para realizar clasificaciones sobre bases de datos; las principales características de este programa son las siguientes:

1. La clasificación se puede realizar hasta por 32 campos distintos; es decir, el resultado final puede estar ordenado por el valor de un campo, y dentro de los que tengan un mismo valor, por el contenido de otro campo, etc. Hasta un máximo de 32 campos.
2. El orden de clasificación es decidido por el usuario, y puede ser ascendente (1, 2, 3, ... ó A, B, C, ...) o descendente (... 3, 2, 1 ó ... C, B, A).
3. También se pueden indicar dos tipos de órdenes alfabéticas: un primer tipo que podemos denominar "orden de listín telefónico" y otro interno de dBASE II.
4. Una última característica importante de este programa consiste en su capacidad para buscar espacio libre en la unidad de almacenamiento —en orden a grabar el resultado de la clasificación— antes de empezar la grabación.

Cualquier programador experto sabe la gran importancia que tienen los programas de clasificación, generalmente denominados SORT. Su utilización es muy frecuente y, por consiguiente, sus ventajas o desventajas se ven multiplicadas por el alto número de veces que se ejecutan. En este caso podemos afirmar que dSORT se trata de una utilidad de notable interés que permite una explotación eficiente de los datos almacenados.

SESION DE TRABAJO CON dBASE II

Para finalizar esta serie de capítulos dedicados al paquete dBASE II, vamos a describir los pasos a seguir en lo que podemos denominar una sesión típica de trabajo:



El paquete dBASE II pertenece a la categoría del software horizontal, o lo que es lo mismo, a las aplicaciones informáticas no concebidas exclusivamente para automatizar una tarea específica. Muy al contrario, la explotación del dBASE II está abierta a cualquier ámbito en el que sea preciso gestionar una base de datos.

1. Después de haber conectado el ordenador y activado el directorio de almacenamiento en el que se encuentre grabado el programa dBASE, basta con teclear la palabra dBASE y, seguidamente, pulsar la tecla RETURN, para que dé comienzo la sesión de trabajo. Inmediatamente, aparecerá en la pantalla un texto de introducción en el que, entre otras cosas, se visualizará la versión del programa en uso y una serie de comentarios de carácter general.
2. La segunda operación a realizar depende del nivel de conocimientos previos que posea el usuario. Si ya domina convenientemente el programa, podrá pasar automáticamente al paso 3.

En otro caso, es recomendable utilizar el comando de ayuda HELP. Existen dos formas distintas para el manejo de este comando. La primera consiste en teclear HELP y pulsar la tecla RETURN. Como respuesta, la pantalla del ordenador mostrará un comentario inicial y una lista exhaustiva de comandos con una breve descripción de su utilidad; dado que la lista es relativamente extensa, cada vez que se llene la pantalla el programa se detendrá mostrando la palabra WAITING, y esperará a que el usuario pulse cualquier tecla para continuar. La segunda posibilidad de manejo del comando HELP consiste en teclear HELP y, a continuación, antes de pulsar RETURN, teclear el nombre del comando que deseamos utilizar. Ahora, el programa contestará con una descripción completa, tanto del comando indicado como de los argumentos que se le deben suministrar.

"SORT" por el método de selección

En este mismo capítulo hemos descrito el objetivo de los programas de clasificación o SORTS: ordenar el contenido de un fichero atendiendo a los valores de algún campo, y producir un nuevo fichero con los registros ya clasificados.

Existen numerosos algoritmos de clasificación que han dado lugar a distintos programas de SORT. No se puede afirmar tajantemente cuál de ellos es el mejor; unos aventajan a otros en determinados aspectos, pero son aventajados en otros. En resumidas cuentas, no existe un algoritmo óptimo.

Una de las mejores formas de comparar los distintos algoritmos entre sí consiste en medir el tiempo que consumen en clasificar un mismo fichero.

A continuación, vamos a detallar uno de los algoritmos más antiguos y también más intuitivo que sirve para la ordenación de un conjunto de números.

Suponga que nuestro objetivo es ordenar en sentido creciente (el razonamiento sería similar para decreciente) un conjunto de N valores, que denominaremos K_1, K_2, ..., K_N.

El algoritmo de selección se basa en la ejecución de N−1 iteraciones: en la primera se garantizará que en K_1 queda almacenado el elemento más pequeño, en la segunda se minimizará K_2 entre todos los valores restantes ... y en la iteración N−1 se garantizará que K_{N-1} contiene el menor valor de los previamente no seleccionados; resulta obvio que el elemento K_N quedará automáticamente clasificado.

a

PASO 1	I = 1
PASO 2	BUSCAR LA CLAVE K_J MENOR ENTRE LAS CLAVES I.....N
PASO 3	$L = K_I$
PASO 4	$K_I = K_J$
PASO 5	$K_J = L$
PASO 6	I = I + 1
PASO 7	SI I = N-1 FIN DEL ALGORITMO; EN CASO CONTRARIO, BIFURCAR A PASO 2

b

	K_1	K_2	K_3	K_4	K_5	K_6
VALOR INICIAL	4	5	8	2	9	1
1ª ITERACION	1	5	8	2	9	4
2ª ITERACION	1	2	8	5	9	4
3ª ITERACION	1	2	4	5	9	8
4ª ITERACION	1	2	4	5	9	8
5ª ITERACION	1	2	4	5	8	9

La técnica empleada en cada iteración se basa en la elección del mínimo valor de los elementos situados entre el ordinal de la iteración y el último lugar. Si a este elemento le llamamos K_j y suponemos que el original de la iteración es i, se intercambiarán a continuación los valores almacenados en K_i y K_j entre sí: es decir, K_i pasará a valer lo que valía K_j, mientras K_j tomará el valor que tenía inicialmente K_i.

Sin duda, este algoritmo no resulta el más efectivo en todos los casos de clasificación, no obstante sí cabe considerarlo como uno de los métodos más sencillos de poner en práctica; y precisamente por ello puede ser utilizado para la realización de sencillos "programas domésticos" para la clasificación de ficheros.

```
***  dBASE II/86   Ver 2.43  1 November  1984

COPYRIGHT (c) ASHTON-TATE 1984
AS AN UNPUBLISHED LICENSED PROPRIETARY WORK.
ALL RIGHTS RESERVED.

Use of this software has been provided under a Software
License Agreement (please read in full). In summary,
you may produce only three back-up copies and use this
software only on a single computer and single terminal.
You may not grant sublicenses nor transfer the software
or related materials in any form to any person unless
Ashton-Tate consents in writing.  This software
contains valuable trade secrets and proprietary
information, and is protected by federal copyright
laws, the violation of which can result in civil
damages and criminal prosecution.

dBASE II is a registered trademark and
dBASE and ASHTON-TATE are trademarks of Ashton-Tate.
```

El primer «pantallazo» de comunicación con el usuario aparece inmediatamente después de ser invocado el programa dBASE II.

CARACTERISTICAS TECNICAS DEL dBASE II

Los derechos de este paquete de aplicación estandarizado pertenecen a la compañía americana ASHTON-TATE.
En cuanto a las limitaciones técnicas de dBASE II podemos citar las siguientes:
a) El máximo número de registros almacenables en una base de datos es de 65.535. Esta limitación es teórica, ya que, por supuesto, la capacidad del soporte de almacenamiento también influye en el máximo tamaño de una base de datos.
b) La suma de los caracteres de todos los campos incluidos en un registro no puede sobrepasar las 1.000 posiciones.

```
================================================================================
||                          d G E N    M A I N    M E N U                      ||
================================================================================
||                                                                            ||
||                              0. exit                                       ||
||                              1. MENU generator                             ||
||                              2. FILE generator                             ||
||                              3. REPORT FORM generator                      ||
||                              4. LABEL generator                            ||
||                                                                            ||
||                                                                            ||
================================== select : : ==================================
```

Reproducción del menú principal presentado por el programa dGEN.

3. El siguiente paso dentro de la sesión de trabajo, será distinto según el objetivo que se persiga. En general, se comenzará abriendo una base de datos ya existente mediante el comando USE, o generando una nueva a través del comando CREATE.
4. Tras acceder a la base de datos a utilizar, el usuario estará en disposición de ejecutar cualquiera de los comandos existentes. Por lo tanto, según sus objetivos realizará una serie de operaciones que no pueden ser indicadas con carácter general.
5. Cuando la explotación de una base de datos vaya a ser intensiva, es decir cuando contenga un gran volumen de información y/o sobre ella se realicen operaciones sistemáticas y complejas, resulta necesario preparar programas que simplifiquen los comandos a ejecutar. Por supuesto, el usuario podrá explotar la base de datos sin preparar ningún programa; si bien, el número de pasos a dar para conseguir sus objetivos será tan elevado que crecerá la probabilidad de cometer errores, y además la explotación resultará farragosa.

Uno de los mayores "encantos" del dBASE II es su generador automático de programas. La sencillez de uso de este programa es sustantiva y con él se pueden generar cómoda y rápidamente programas para la explotación de una base de datos. Cuando un usuario invoca al programa dGEN, aún sin poseer conocimientos de informática, se verá en condiciones de desarrollar programas que faciliten la gestión de la base de datos.

c) El máximo número de campos por registro, es decir de datos que pueden agruparse, es de 32.
d) En cuanto al tope de caracteres que pueden utilizarse en un único campo, éste se encuentra fijado en 254. Este mismo número, 254, es el máximo número de caracteres que pueden conformar una cadena de caracteres, una línea de comando o la cabecera de un informe.
e) Los datos numéricos deben estar comprendidos entre $\pm\ 1{,}8 \times 10^{63}$ y $\pm\ 1 \times 10^{-63}$; estos son los números extremos procesables con el dBASE II. La precisión de cálculo llega hasta los 10 dígitos.
f) Si bien el tamaño de un campo puede llegar hasta 254 caracteres, cuando se trata de un índice su longitud no debe sobrepasar los 100 caracteres.

El BASIC en acción

Batalla naval

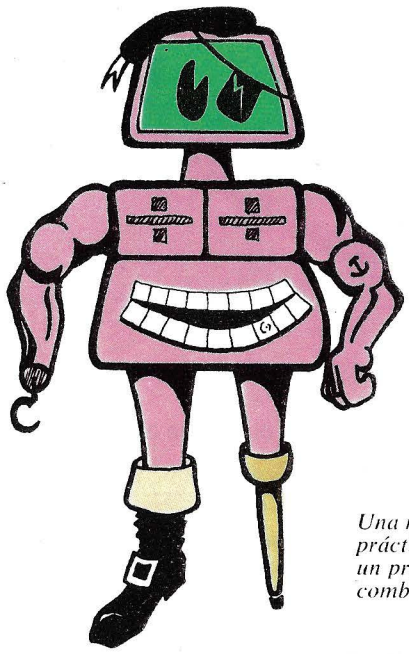
A lo largo de este capítulo tendrá lugar una nueva incursión en el terreno práctico, a través de la confección, paso a paso, de un programa capaz de simular un combate naval. Una feroz persecución entre un submarino y un barco, en donde el ordenador desempeña el papel de submarino y el usuario guiará el barco perseguidor.

El juego es muy similar al tradicional juego de los barcos. La superficie de juego es la misma: una retícula con cien casillas, con diez filas identificadas por las letras que van de la A a la J y de diez columnas numeradas del 1 al 10. Dentro de esta superficie se encuentra escondido el submarino al que se pretende hundir. Este submarino no tiene ninguna intención, como es lógico, de hacernos la tarea fácil. Esa es la razón de que durante el juego se esté moviendo sin parar. Por nuestra parte disponemos de un sensor bastante perfeccionado. Este proporcionará una medida de la distancia a la que se encuentra el submarino tras efectuar el lanzamiento de una de las minas.

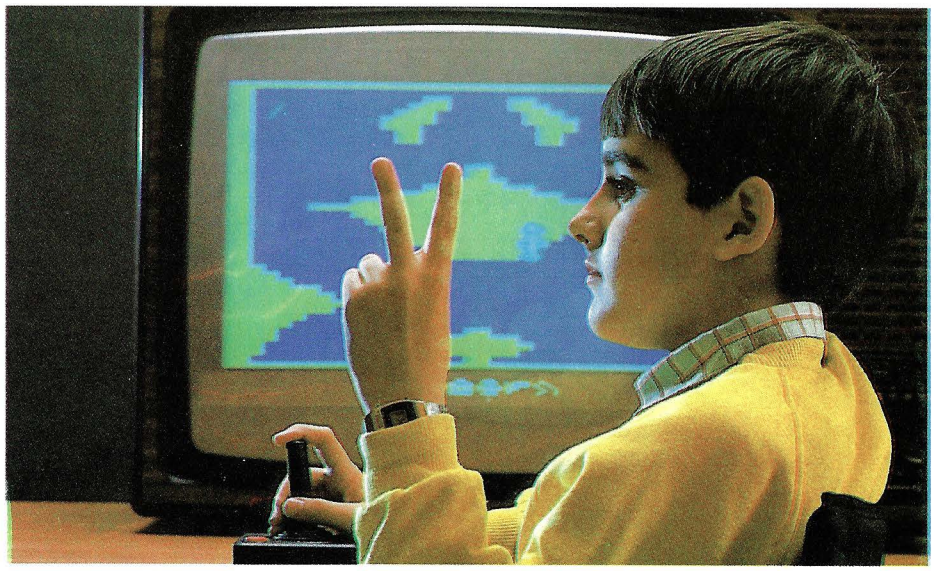
En primer lugar, resulta conveniente situar en pantalla una referencia para conocer los puntos del plano sobre los que está permitido lanzar las cargas de profundidad. Para ello se utilizarán unas cabeceras de fila y columna. Las filas se distinguirán por las letras que van de la A a la J. Por su parte, las columnas se identificarán con un número del 1 al 10.

Para dibujar las cabeceras de las columnas es conveniente utilizar un bucle, lo que proporciona una solución bastante elegante. Como cabecera de la columna se escribirá el valor que toma en cada pasada la variable contadora del siguiente bucle:

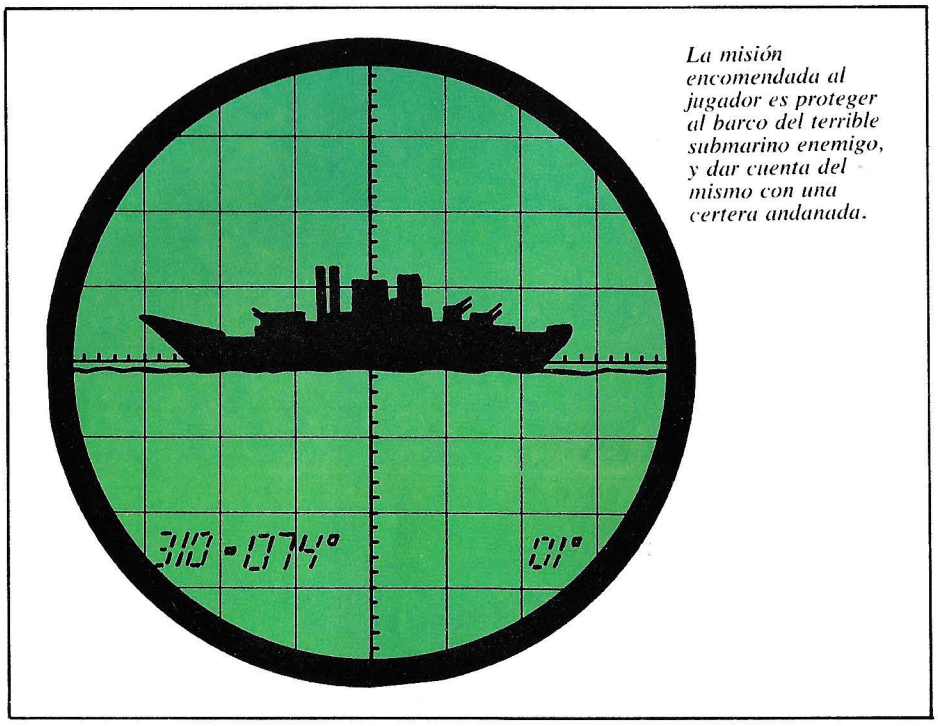
```
20 FOR I=1 TO 10
30 PRINT AT (0,2*I+5);I;" ";
40 NEXT I
```



Una nueva incursión en el terreno de la práctica, en este caso confeccionando un programa capaz de simular un combate naval.



Las facultades del BASIC como lenguaje de propósito general quedan bien patentes en la programación de juegos.



La misión encomendada al jugador es proteger al barco del terrible submarino enemigo, y dar cuenta del mismo con una certera andanada.

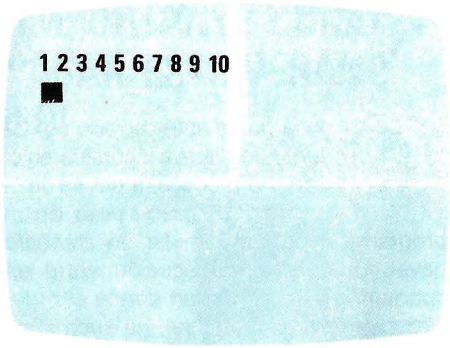
Resulta conveniente añadir una instrucción previa que borre la pantalla. Esta instrucción puede ser la siguiente:

```
10 CLS
```

En el caso de que el ordenador con el que se esté trabajando no posea la función AT, el bucle construido puede ser sustituido por las siguientes líneas de programa:

```
20 PRINT SPC(5); FOR I=1 TO 10
30 PRINT I; " ";
40 NEXT I
```

El resultado que se obtiene en pantalla será el siguiente:



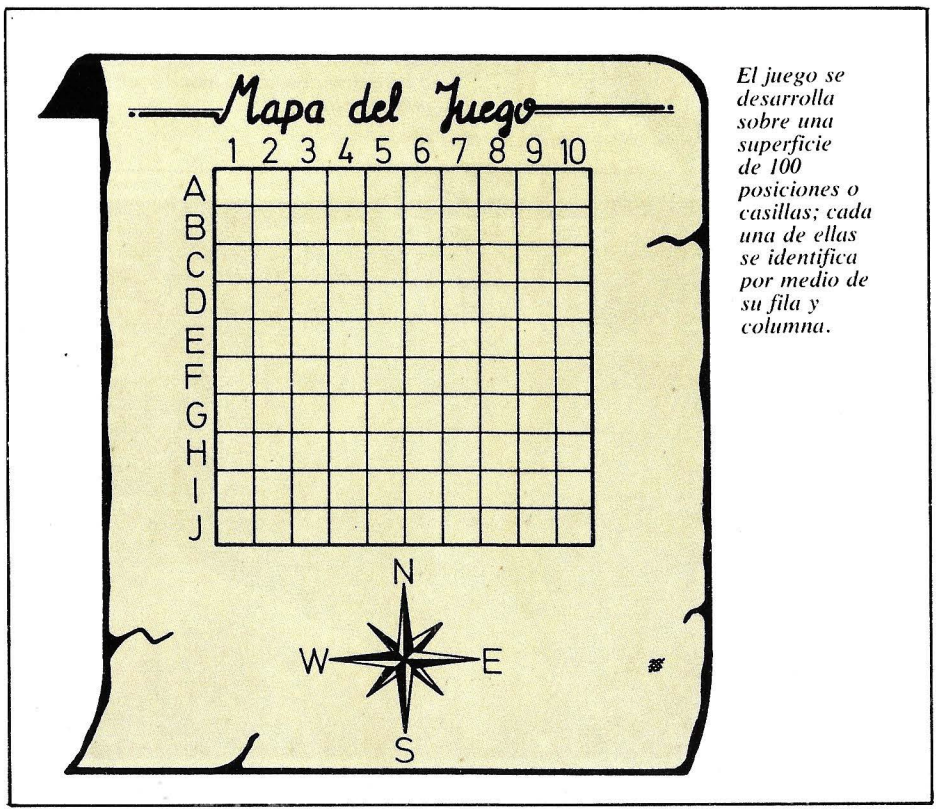
En último extremo, si existiesen aún problemas de compatibilidad con el dialecto BASIC del ordenador que esté utilizando, la rutina podría reducirse a la siguiente línea de programa:

```
20 PRINT " 1 2 3 4 5 6 7 8 9 10"
```

A continuación, hay que proceder a colocar los índices de las filas; tarea que puede realizarse de la siguiente forma:

```
 50 PRINT TAB(4); "A"
 60 PRINT
 70 PRINT TAB(4); "B"
 80 PRINT
 90 PRINT TAB(4); "C"
100 PRINT
110 PRINT TAB(4); "D"
120 PRINT
130 PRINT TAB(4); "E"
140 PRINT
150 PRINT TAB(4); "F"
160 PRINT
170 PRINT TAB(4); "G"
180 PRINT
190 PRINT TAB(4); "H"
200 PRINT
210 PRINT TAB(4); "I"
220 PRINT
230 PRINT TAB(4); "J"
```

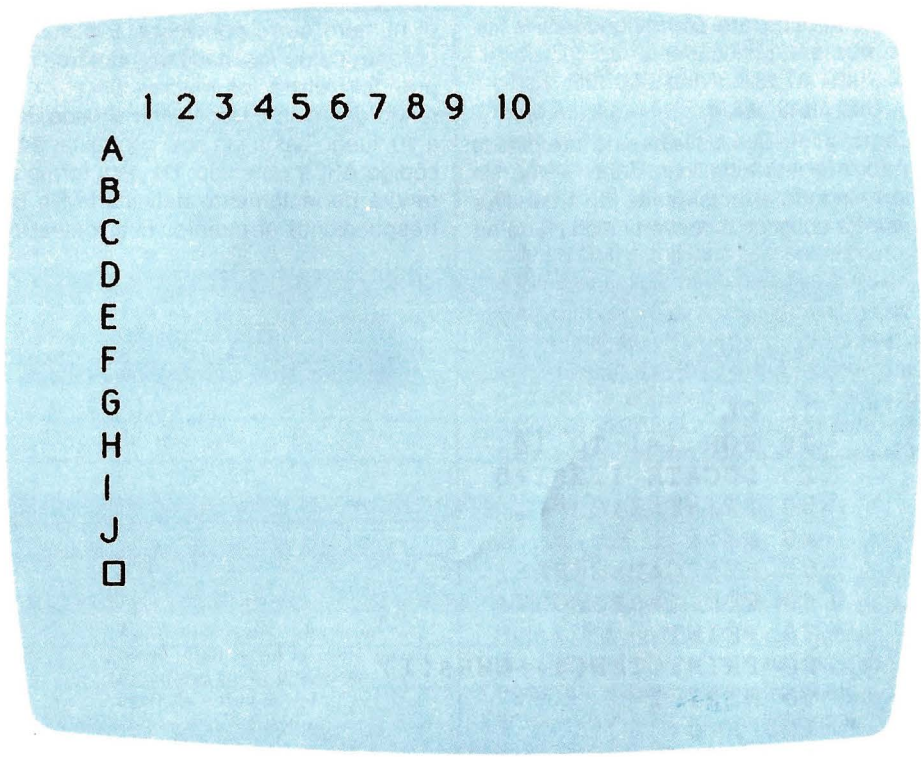
Desde luego que la rutina propuesta es plenamente eficaz, aunque también es cierto que resulta muy pesada de escribir. Además, ocupa mucha memoria y es pro-



El juego se desarrolla sobre una superficie de 100 posiciones o casillas; cada una de ellas se identifica por medio de su fila y columna.

pensa a que se cometan errores a la hora de introducirla. En su lugar, cabe pensar en construirla en base a un bucle, de forma semejante al anterior. En este caso, no se pretende escribir un número sino una letra, por lo que la escritura directa del índice del contador no resulta una solución oportuna. Pero, si se piensa un poco, se llega a la conclusión de que estas letras consecutivas tienen además unos códigos ASCII también consecutivos. Dichos códigos ASCII van del 65 al 74 y, dado que es posible a partir del código ASCII generar el carácter correspondiente, lo que haremos será utilizar un bucle cuyo valor inicial sea 65 (es decir, el código ASCII de la primera de las letras a representar) y como valor final el 74 (código de la última de las letras a representar).

```
60 FOR I=65 TO 74
70 PRINT
80 PRINT TAB(4);CHR$(I)
90 NEXT I
```



Pantalla de juego generada por la rutina que se incluye en el texto.

En pantalla se obtendría el resultado que puede observarse en la correspondiente figura.

Para medir los disparos o intentos de abatir el submarino, se utilizará una variable contadora; variable que es necesario inicializar a cero. La instrucción a ejecutar para poner el contador a cero es la siguiente; por supuesto, en caso de que el contador coincida con la variable K:

```
100 LET K=0
```

A continuación, y dentro todavía del conjunto de instrucciones que sólo se ejecutarán una vez al principio del juego, queda por confeccionar una zona del programa que fije la posición de partida del submarino de forma aleatoria.

Aquí se hace necesario el empleo de la instrucción RND en orden a generar un número aleatorio. Desde luego, la función RND genera un número que en principio no servirá de mucho, puesto que se tratará de un número comprendido entre 0 y 1, pero sin alcanzar ninguno de estos límites. Por esta razón, se hace necesario efectuar un cambio de escala, convirtiendo el número generado en un valor comprendido entre uno y diez. La función capaz de obtener un número entre dos límites establecidos puede expresarse como sigue:

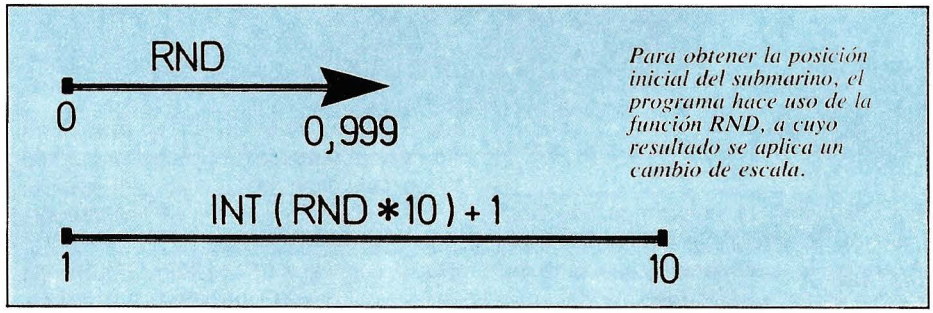
<número deseado>=INT (<número de posibilidades>*RND) +
<valor mínimo del número>

Aplicado al caso que nos ocupa, dicha expresión se convierte en las dos instrucciones siguientes, las cuales proporcionan las coordenadas X e Y iniciales:

```
110 LET X=INT(10*RND)+1
120 LET Y=INT(10*RND)+1
```

En este punto hay que hacer la salvedad de que, probablemente, será necesario sustituir la instrucción RND en muchos ordenadores por la variante RND(0); ésta es la formulación correcta en muchos dialectos BASIC.

A continuación se entra ya en el juego propiamente dicho. De entrada hay que brindar al jugador la oportunidad de realizar su intento. Existen varias formas de instruir al ordenador para que pida al jugador su intento. La más sencilla consiste en ejecutar un INPUT en el que se pregunte por las coordenadas de lanzamiento. Estas se suministrarán de forma similar a como se hace en el tradicional juego de los barcos; esto es, mediante una letra y



Para obtener la posición inicial del submarino, el programa hace uso de la función RND, a cuyo resultado se aplica un cambio de escala.

un número. La letra corresponderá a la fila y el número a la columna:

```
140 PRINT AT 23,5: INPUT "DONDE
    DISPARAS";A$,M
```

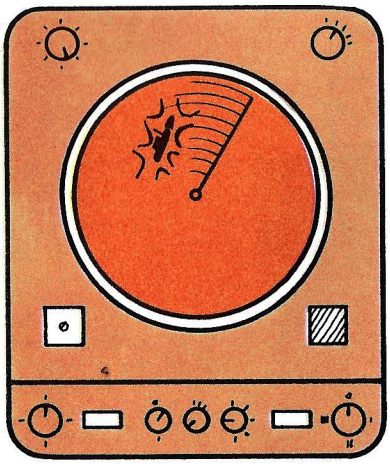
Como quiera que la fila ha sido introducida mediante una letra, para saber a qué fila corresponde exactamente, el ordenador se verá obligado a convertir dicha letra en el número correspondiente. Ello supone obtener como resultado un número comprendido entre los valores 65 y 74. El rango del mismo debe estar situado de 1 a 10, luego basta tan sólo con restar 64 al código ASCII obtenido. De esta forma se tendrá perfectamente definida la fila correspondiente al intento. Esta operación se puede realizar con la instrucción siguiente:

```
150 LET N=ASC (A$)-64
```

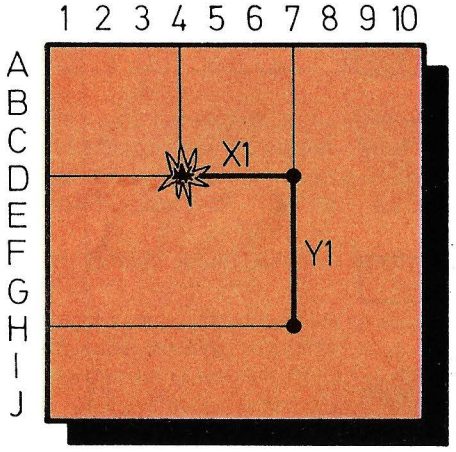
Una vez efectuado el lanzamiento, hay que determinar a qué distancia del blanco impactó la mina (evidentemente la distancia puede ser cero... ¡hundido!). La distan-

```
10 CLS
20 FOR I=1 TO 10
25 LOCATE 1,2*I+5
30 PRINT I;" "
40 NEXT I
50 REM CABECERAS
60 FOR I=65 TO 74
70 PRINT
80 PRINT TAB(4);CHR$(I)
90 NEXT I
100 K=0
105 REM POSICION DEL SUBMARINO
110 LET X=INT(10*RND(0)+1)
120 LET Y=INT(10*RND(0)+1)
130 REM INTRODUCCION DEL INTENTO
140 LOCATE 23,4:INPUT "DONDE DISPARAS";A$,M
150 LET N=ASC(A$)-64
160 LET D=ABS(M-X)+ABS(N-Y)
165 LOCATE N*2+1,M*2+5
170 PRINT D
180 LET K=K+1
190 IF D=0 THEN GOTO 280
200 LET X=X+INT(3*RND(0))-1
210 LET Y=Y+INT(3*RND(0))-1
220 IF X=0 THEN X=2
230 IF X=11 THEN X=9
240 IF Y=0 THEN Y=2
250 IF Y=11 THEN Y=9
260 GOTO 140
270 CLS
280 LOCATE 8,8:PRINT "HUNDIDO EN ";K;" INTENTOS"
290 END
```

LISTADO 1.
En el mismo se han sustituido los comandos PRINT AT por LOCATE/PRINT. Las líneas afectadas son 25/30, 140 y 165/170.



El jugador cuenta con la ayuda de un preciso instrumento, el cual indica la distancia a la que se encontraba el submarino cuando se lanzó la última carga.



La distancia del submarino viene dada como la suma de lo valores de la diferencia entre la posición del submarino y punto de impacto del disparo, tanto en sentido horizontal (abscisas) como vertical (ordenadas).

cia no se medirá en diagonal, sino que, al menos en un principio, se evaluará de una forma un tanto curiosa aunque eficaz. Exactamente, se medirá la referida distancia en número de columnas desde el blanco al impacto y, análogamente, se medirá la separación en filas entre ambos puntos considerados (impacto y posición del submarino). Ambos valores se sumarán dando como resultado la información que se dará al jugador para que perfile su próximo lanzamiento.

Sobre el punto de la pantalla al que se dirigió el disparo aparecerá indicada la distancia a la que se encuentra el blanco. A su vez, tras cada andanada, se incrementará en una unidad el contador de lanzamientos efectuados. Todo ello es posible programarlo por medio de las siguientes instrucciones:

```
160 LET D=ABS(M-X)+ABS(N-Y)
170 PRINT AT N*2,M*2+5;D
180 LET K=K+1
```

En el caso de que el ordenador con el que se esté trabajando no posea la función AT, la línea 170 puede ser sustituida por la siguiente:

```
170 <HOME>:FOR I=1 TO N*2:PRINT:NEXT I:
    PRINT TAB(M*2+5); D
```

<HOME> adoptará en cada caso la formulación que corresponda a cada ordenador, y cuyo efecto sea llevar al cursor a la esquina superior izquierda sin borrar la pantalla.

A continuación se inspeccionará la distancia entre el blanco y el punto de destino del disparo, para determinar si ésta es nula (lo cual significará que se ha acertado en el blanco). Si el disparo ha dado en el blanco se producirá una bifurcación de la rutina final del programa. En ella se señala que el juego ha terminado y se muestra el número de andanadas que se han disparado. La instrucción necesaria para efectuar tal verificación puede ser la siguiente:

```
190 ID D=0 THEN GOTO 280
```

Si no se ha destruido al enemigo, el ordenador moverá el submarino y, de esta forma, dificultará la tarea del jugador. Para desplazar el blanco, la máquina debe generar un número aleatorio comprendido entre −1 y +1 para las columnas, y otro valor análogo para las filas; en función de dichos valores se actualizará la posición del submarino. Las siguientes instrucciones realizan esa acción:

```
200 LET X=X+INT(3*RND)-1
210 LET Y=Y-INT(3*RND)-1
```

Puede darse el caso de que la trayectoria del submarino conduzca a éste fuera de los dominios del tablero de juego. En tal caso, es preciso corregirla simulando un rebote contra el límite del tablero. Las siguientes cuatro instrucciones se encargarán de corregir la trayectoria:

```
220 IF X=0 THEN X=2
230 IF X=11 THEN X=9
240 IF Y=0 THEN Y=2
250 IF Y=11 THEN Y=9
```

Una vez controlado el movimiento del submarino, hay que pasar a atender el próximo disparo que realice el jugador. Así pues, la secuencia del programa ha de

regresar mediante una instrucción GOTO, a la línea 140 en donde se introducen las nuevas coordenadas de disparo:

```
260 GOTO 140
```

Por último tan sólo queda por construir la rutina final. En ella se indicará el fin de la partida y el número de intentos que han sido necesarios para hundir el submarino:

```
270 CLS
280 PRINT AT 8,8; "HUNDIDO EN ";K;"
    INTENTOS"
290 END
```

Y el programa está ya completo, ofreciendo el aspecto que revela el LISTADO 1.
El programa desde luego admite algunas mejoras. Una de ellas es la de incluir una rutina al final, que muestre el record obtenido hasta el momento por las personas que hayan jugado con el ordenador durante esa misma sesión. También puede incluirse una zona que pregunte al usuario si desea continuar jugando. Todo ello puede realizarse por medio de la rutina que pasamos a describir.
En primer lugar debe solicitarse el nombre de la persona que acaba de terminar el juego:

```
300 INPUT "INTRODUCE TU NOMBRE";B$
```

Dicho nombre queda pues almacenado en la variable de cadena B$. El nombre del jugador que hasta ese momento detente la máxima puntuación estará en R$. Las puntuaciones conseguidas se encuentran en las variables K (la del último jugador) y K1 (la del jugador que obtuvo la máxima puntuación). Para saber si el nuevo jugador ha obtenido una puntuación superior es necesario proceder a una comparación. En el caso de que el record se haya mejorado, se harán K1 y R$ iguales a K y B$, respectivamente.

```
310 IF K<K1 THEN LET K1=K:R$=B$
```

A continuación, se visualizará en la pantalla el nombre y la puntuación de la persona que ostente el record actual:

```
320 PRINT "EL RECORD ES ";K1;" POR ";R$
```

Por último, nos encontramos con la zona encargada de preguntar al jugador si de-

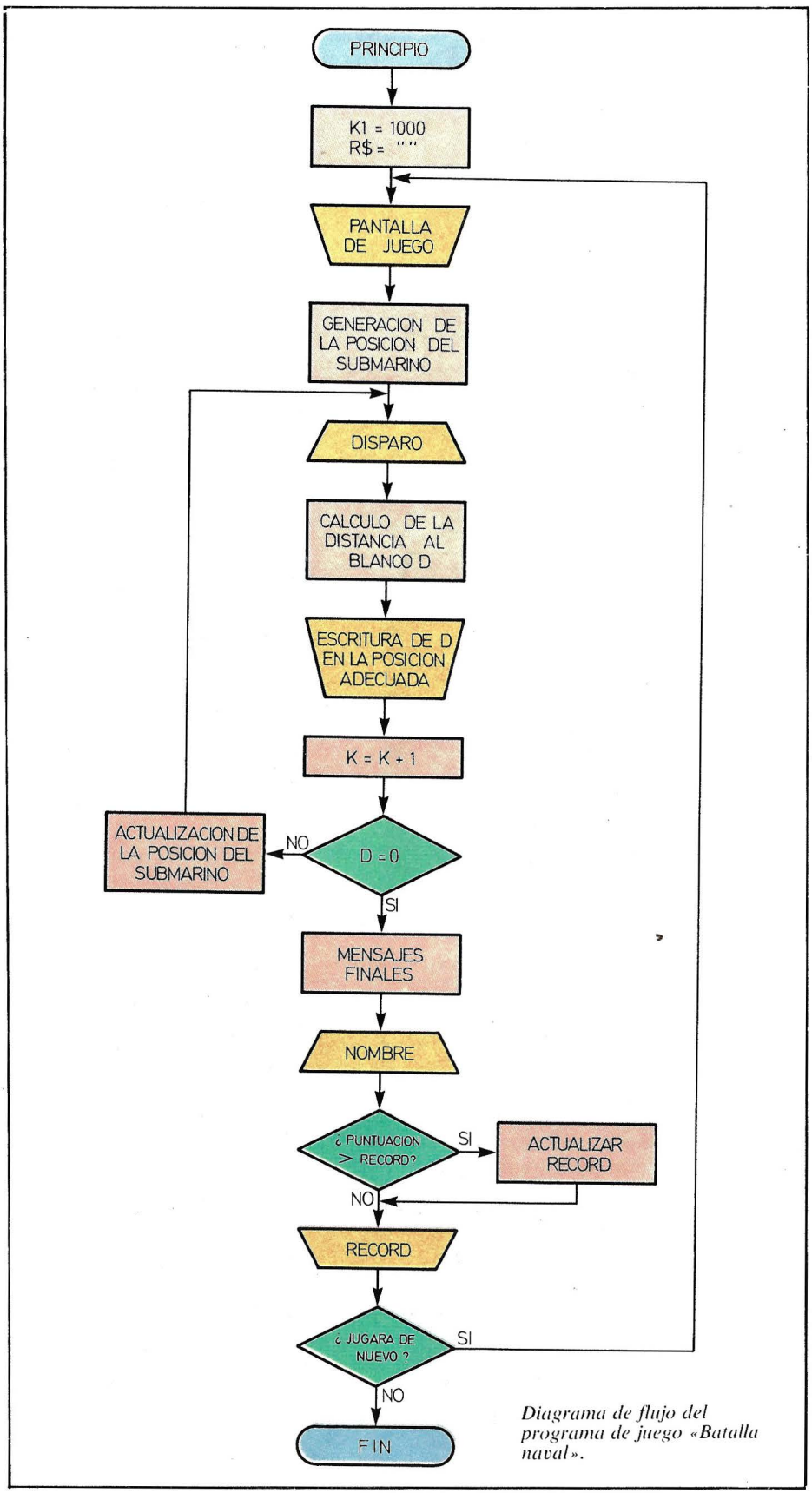
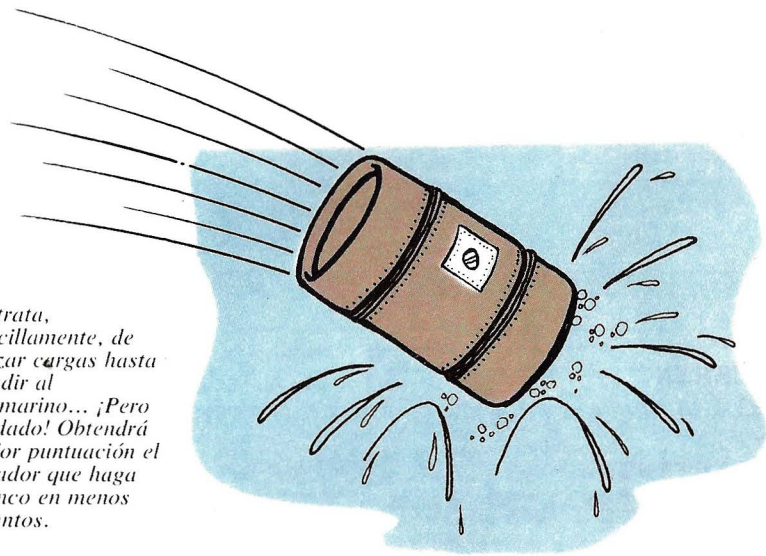


Diagrama de flujo del programa de juego «Batalla naval».



Se trata, sencillamente, de lanzar cargas hasta hundir al submarino... ¡Pero cuidado! Obtendrá mejor puntuación el jugador que haga blanco en menos intentos.

VARIABLES UTILIZADAS EN EL PROGRAMA

A$	Dato introducido.
B$	Nombre del jugador en curso.
D	Distancia al blanco.
I	Indice de bucle.
K	Número de intentos.
K1	Puntuación record.
M,N	Posición del disparo.
R$	Nombre del jugador que obtuvo el récord.
X,Y	Posición del submarino.

sea realizar una nueva partida. Al efecto, debe proyectarse en la pantalla el correspondiente mensaje. De ello se ocupará la instrucción siguiente:

```
330 PRINT "QUIERES JUGAR DE NUEVO (S/N)?"
```

Para recoger la respuesta obtenida se utilizará la función INKEY$, asignando el valor de la misma a la variable C$. A continuación, se comprueba si se ha pulsado una tecla antes de continuar el proceso. En el caso de que no se haya pulsado tecla alguna, habrá que volver a examinar la entrada por el teclado hasta obtener una respuesta.

```
340 LET C$=INKEY$: IF C$=" " THEN GOTO 340
```

Para determinar si la respuesta es afirmativa o negativa entrarán en juego instrucciones condicionales. Estas detendrán el proceso en el caso de que la respuesta sea negativa, o bien lo reiniciarán si fuera afirmativa:

```
350 IF C$="S" THEN GOTO 10
360 IF C$="N" THEN END
```

Si la pulsación detectada no corresponde con las letras S o N, será preciso inspeccionar de nuevo el teclado hasta obtener una respuesta correcta:

```
370 GOTO 340
```

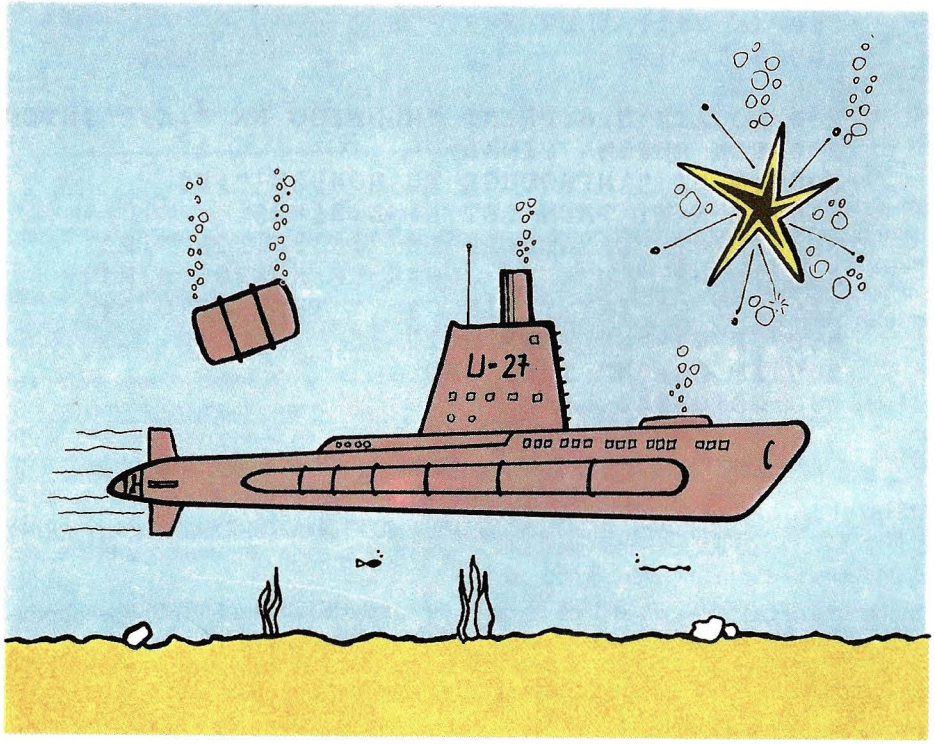
Por lo demás, es necesario añadir una línea de programa que inicialice las variables K1 y R$ con valores nulos para comenzar así el proceso. En el caso de la variable K1, es necesario inicializarla con un número relativamente alto, ya que la mejor puntuación es, en esta ocasión, la más baja. De inicializarla a 0 nadie sería capaz de obtener una mejor puntuación.

```
5 LET K1=1000:LET R$=" "
```

En ciertos ordenadores, su intérprete BASIC no incluye la función INKEY$, lo que obligará a recurrir a otro tipo de instrucciones. Por ejemplo, si estuviera disponible el comando GET, habría que sustituir la línea 340 por la siguiente:

```
340 GET C$:IF C$=" " THEN GOTO 340
```

Con ello el programa funcionará correctamente y sin ninguna diferencia respecto al



No basta con hacer blanco en las proximidades del submarino, sino que hay que acertarle de pleno; de lo contrario, este seguirá en constante desplazamiento dificultando la misión.

```
5 K1=1000:R$=""
10 CLS
20 FOR I=1 TO 10
25 LOCATE 1,2*I+5
30 PRINT I;" "
40 NEXT I
50 REM CABECERAS
60 FOR I=65 TO 74
70 PRINT
80 PRINT TAB(4);CHR$(I)
90 NEXT I
100 K=0
105 REM POSICION DEL SUBMARINO
110 LET X=INT(10*RND(0)+1)
120 LET Y=INT(10*RND(0)+1)
130 REM INTRODUCCION DEL INTENTO
140 LOCATE 23,4:INPUT "DONDE DISPARAS";A$,M
150 LET N=ASC(A$)-64
160 LET D=ABS(M-X)+ABS(N-Y)
165 LOCATE N*2+1,M*2+5
170 PRINT D
180 LET K=K+1
190 IF D=0 THEN GOTO 280
200 LET X=X+INT(3*RND(0))-1
210 LET Y=Y+INT(3*RND(0))-1
220 IF X=0 THEN X=2
230 IF X=11 THEN X=9
240 IF Y=0 THEN Y=2
250 IF Y=11 THEN Y=9
260 GOTO 140
270 CLS
280 LOCATE 8,8:PRINT "HUNDIDO EN ";K;" INTENTOS"
290 REM RUTINA FINAL
300 INPUT "INTRODUCE TU NOMBRE ";B$
310 IF K<K1 THEN LET K1=K:R$=B$
320 PRINT "EL RECORD ES ";K1;" POR ";R$
330 PRINT "QUIERES JUGAR DE NUEVO (S/N)?"
340 LET C$=INKEY$:IF C$="" THEN GOTO 340
350 IF C$="S" THEN GOTO 10
360 IF C$="N" THEN END
370 GOTO 340
```

LISTADO 2.
El programa al completo. En el listado adjunto se utilizan de nuevo estructuras LOCATE/PRINT en sustitución de los comandos PRINT AT.

anterior. No obstante, si tampoco existiera el referido comando en el dialecto BASIC utilizado, sería necesario recurrir a la instrucción INPUT. Con ella no basta con pulsar la tecla correspondiente a la respuesta, sino que además hay que accionar la tecla correspondiente al retorno de carro (RETURN o ENTER) tras introducir el valor oportuno.

```
340 INPUT C$
```

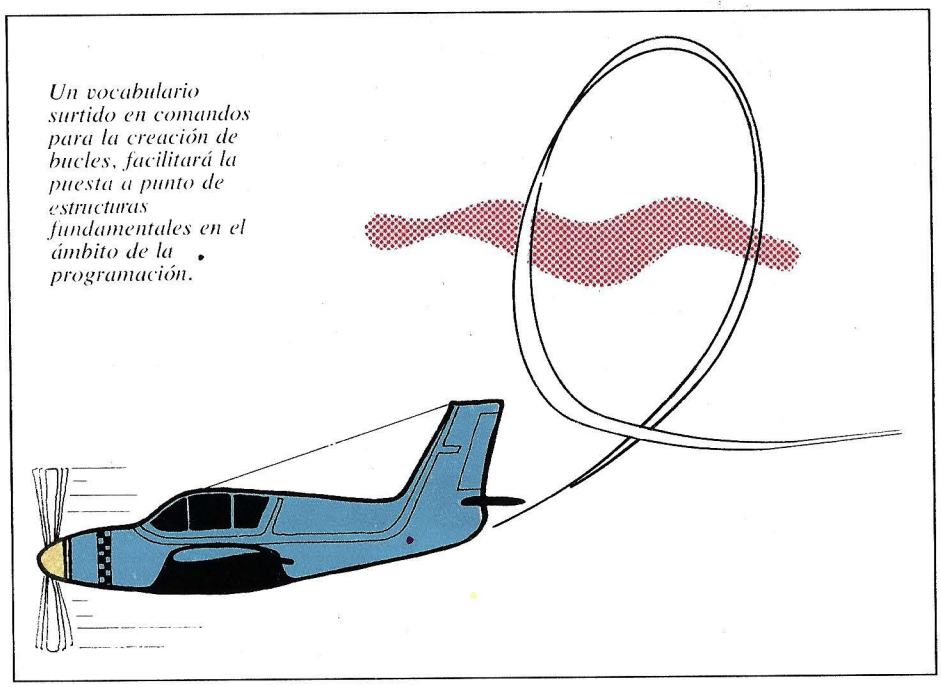
El listado completo, una vez efectuadas estas últimas correcciones, quedará tal como refleja el LISTADO 2 que acompaña al texto.

Para comprender más fácilmente la estructura y funcionamiento del programa confeccionado, es interesante examinar su correspondiente diagrama de flujo, el cual se reproduce en la figura adjunta.

Forth (5)

Decisiones y bucles

Un aspecto de gran importancia en cualquier lenguaje de programación es el relativo a sus facultades para el tratamiento de decisiones y bucles. Un vocabulario surtido en herramientas con esta finalidad, facilitará la puesta a punto de estructuras fundamentales en el ámbito de la programación.



Un vocabulario surtido en comandos para la creación de bucles, facilitará la puesta a punto de estructuras fundamentales en el ámbito de la programación.

REALIZANDO COMPARACIONES

En esencia, una decisión consiste sencillamente en la elección de una entre dos o más posibles alternativas. Cada una de las opciones a seleccionar tiene asociado un proceso diferente, ya sea en su totalidad o en parte, respecto al asociado a las restantes alternativas que se ofrecen.
Toda decisión se fundamenta en una determinada condición a evaluar. En el caso de disponer de dos alternativas, si el resultado de evaluar la condición impuesta es *cierto,* se actuará en un sentido, mientras que si es *falso* se actuará en otro. A raíz de tal razonamiento, parece obvio que lo primero que es preciso conocer es cuáles son las condiciones que pueden darse al ordenador para que éste proceda a su examen y, en consecuencia, tome una decisión. En el caso del lenguaje FORTH, las condiciones admisibles son las establecidas por medio de los operadores cuyas palabras FORTH se definen a continuación:

= Toma los dos números situados más cerca de la pila y examina si son iguales.

< Toma los dos números de la cima de la pila y comprueba si el segundo es menor que el primero. Ello equivale a examinar si el número situado en la cima de la pila es mayor que el que se encuentra a continuación de éste.

> Toma los dos números de la cima de la pila y comprueba si el segundo número es mayor que el primero.

0= Toma el número situado en la cima de la pila y comprueba si es cero.

0< Comprueba si el número situado en la cima de la pila es menor que cero.

0> Toma el número situado en la cima de la pila y comprueba si es mayor que cero.

El resultado de cualquiera de estas evaluaciones puede tomar dos valores: VERDADERO o FALSO. En el caso de ser verdadero, el ordenador deposita en la cima de la pila un uno, mientras que si es falso el valor depositado será un cero.
En este punto resulta conveniente señalar la posibilidad que existe de combinar dos o más condiciones para obtener un único resultado. Veamos algún ejemplo, expresado literalmente, ilustrativo de condiciones evaluables de acuerdo a la referida posibilidad:

"¿El número A es mayor que B y a su vez el número D es menor que B".
"¿El número A es mayor que B o A es igual a B".

El medio para establecer estas condiciones compuestas, derivadas de la síntesis de dos o más condiciones elementales, lo aportan los operadores lógicos. Como ejemplo veamos cómo se escribirían las condiciones enunciadas anteriormente en el caso de sustituir los datos A, B y D por ciertos valores numéricos constantes y determinados.

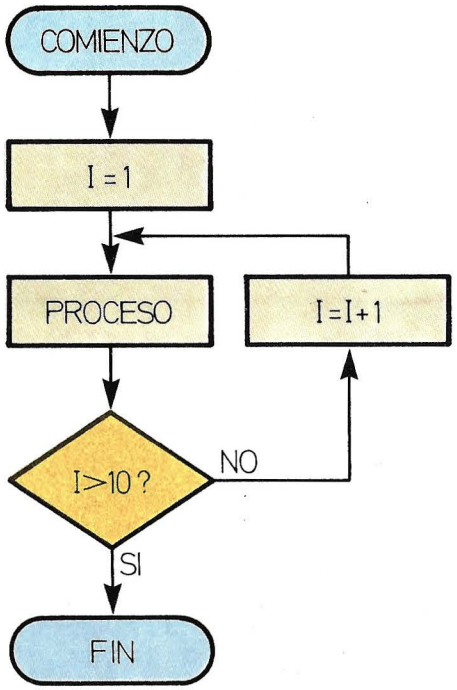


Diagrama de flujo correspondiente a un bucle controlado.

```
4 7 > 3 7 < AND . <CR>

4 7 > 3 7 < AND . 0 OK

5 5 > 5 5 = OR . <CR>

5 5 > 5 5 = OR . 1 <CR>
```

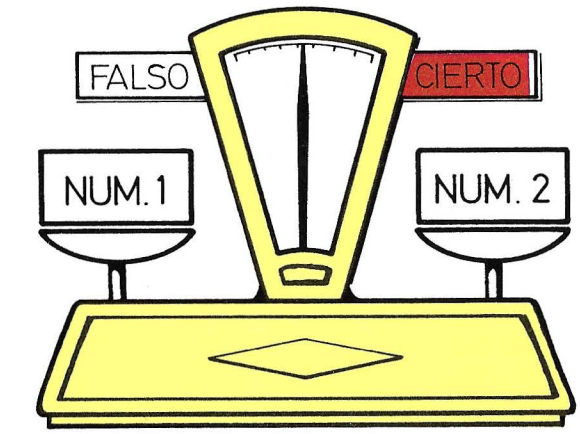
ELIGIENDO EL CAMINO ADECUADO

Una vez que se conoce la forma de imponer una condición, es preciso saber cómo aplicar su examen a la decisión entre dos alternativas.

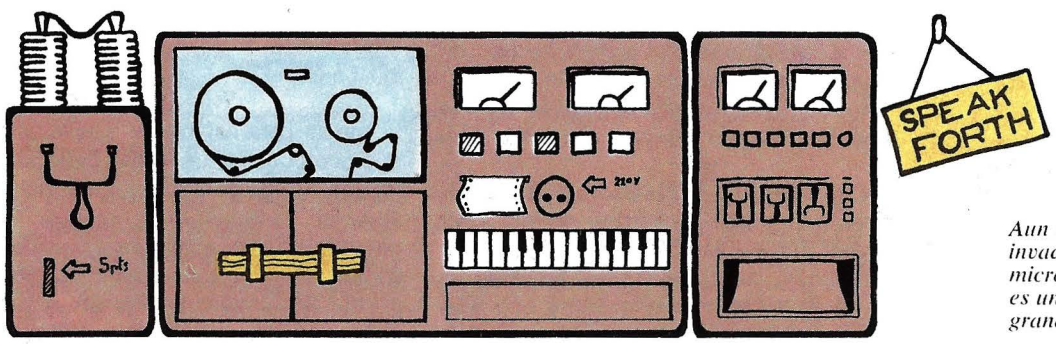
Existen tres palabras claves del diccionario FORTH ligadas con las estructuras de decisión. Estas son: IF, ELSE y THEN. La primera de ellas se emplea para ordenar a la máquina que tome una decisión. La segunda delimita el primer bloque de instrucciones, justamente el que se ejecutará en el caso de que la condición sea cierta; dicha sección de instrucciones será, por lo tanto, la comprendida entre las palabras IF y ELSE.

El segundo bloque de instrucciones, el que se ejecutará en el caso de que la condición no se cumpla, queda delimitado entre las palabras ELSE y THEN.

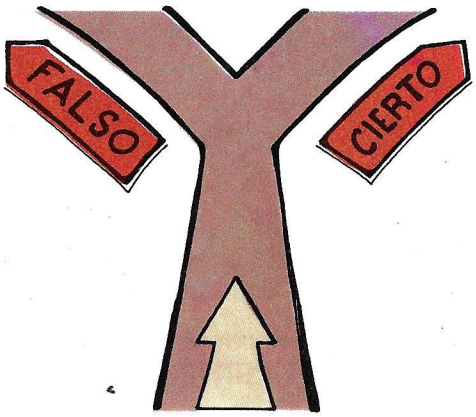
Sin lugar a dudas, un ejemplo vale más que mil palabras; veamos pues un ejemplo de aplicación de una de estas condiciones. Para ello vamos a definir una palabra que llamaremos EXAMEN. En caso de que la cima de la pila contenga un número inferior a 50, el ordenador exhibirá en la pantalla el mensaje SUSPENSO; mientras que si dicho número es mayor que 50 aparecerá el mensaje APROBADO.



Los operadores lógicos permiten comparar los elementos, obteniendo un resultado que puede adoptar tan sólo dos valores: verdadero o falso.



Aun cuando en la actualidad invade el terreno de los microordenadores, el FORTH es un lenguaje que se creó sobre grandes equipos.



Las instrucciones de bifurcación permiten seleccionar uno entre los dos caminos posibles; la elección dependerá del cumplimiento o no de la condición que se imponga.

```
: EXAMEN <CR>
50 < <CR>
   IF <CR>
      . " SUSPENSO" <CR>
   ELSE <CR>
      . "APROBADO" <CR>
   THEN <CR>
; <CR>
```

A modo de respuesta, el ordenador mostrará en pantalla la siguiente confirmación.

```
: EXAMEN 50 < IF . " SUSPENSO" ELSE."
APROBADO" THEN ; OK
```

Una vez definida la palabra EXAMEN, es posible ya utilizarla en la práctica. Por ejemplo:

```
40 EXAMEN <CR>
40 EXAMEN SUSPENSO OK
```

```
70 EXAMEN <CR>
70 EXAMEN APROBADO OK
```

Comparaciones

Los operadores de comparación trabajan con números depositados en la pila, dando como resultado los valores uno (verdadero) o cero (falso).

= Comparador de igualdad
< Menor que
> Mayor que
0= Comprueba si el número es igual a cero
0< Comprueba si el número es negativo
0> Comprueba si el número es positivo

IF/THEN/ELSE

Esta estructura permite tomar decisiones entre dos alternativas, en base a una condición impuesta. El proceso que se ha de ejecutar en el caso de que la condición sea cierta se detalla entre las palabras IF y ELSE, mientras que el proceso a ejecutar en caso de que no se cumpla la condición se especificará entre las palabras ELSE y THEN.

DO/LOOP

Permite definir bucles que se ejecutarán repetidamente un determinado número de veces, a partir de un valor inicial y hasta un valor final; estos dos valores deben depositarse previamente en la pila. El proceso que constituye el bucle debe escribirse entre las palabras clave DO y LOOP.

En ocasiones es posible que la evaluación de la condición impuesta no conduzca a dos procesos distintos entre los que elegir; en efecto, uno de los procesos puede ser nulo y dar lugar a que prosiga, sencillamente, la ejecución del programa. En semejante situación no es necesario incluir la palabra ELSE utilizada en el ejemplo al definir la palabra EXAMEN. Ejemplo:

```
: ELIGE <CR>
DUP 13 = <CR>
IF <CR>
DROP 12 <CR>
THEN <CR>
; <CR>
```

```
: ELIGE DUP 13 = IF DROP 12 THEN ; OK
```

La nueva palabra definida, ELIGE, examinará el número situado en la cima de la pila, y en caso de que éste sea igual a 13 lo sustituirá por el valor 12. Este es un ejemplo elemental de estructura de decisión en la que está ausente la zona ELSE.

BUCLES ELEMENTALES

Un bucle consiste en la repetición cíclica y bajo control de una secuencia de instrucciones. Una forma elemental de construir un bucle es la que ilustra el siguiente ejemplo:

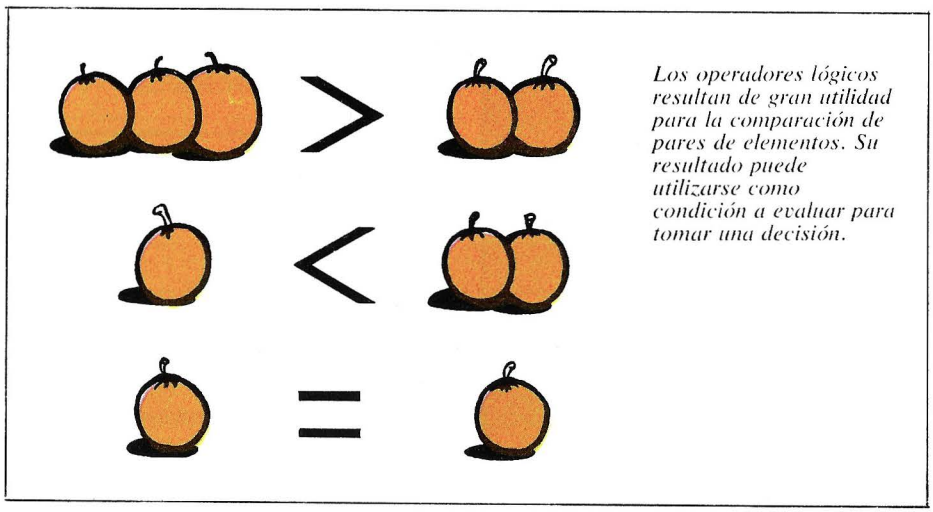
```
: BUCLE <CR>
DUP 10 < <CR>
IF <CR>
1+ DUP . BUCLE <CR>
ELSE <CR>
. " FIN" <CR>
; <CR>
```

```
: BUCLE DUP 10 IF 1+ DUP . BUCLE ELSE."
FIN" ; OK
```

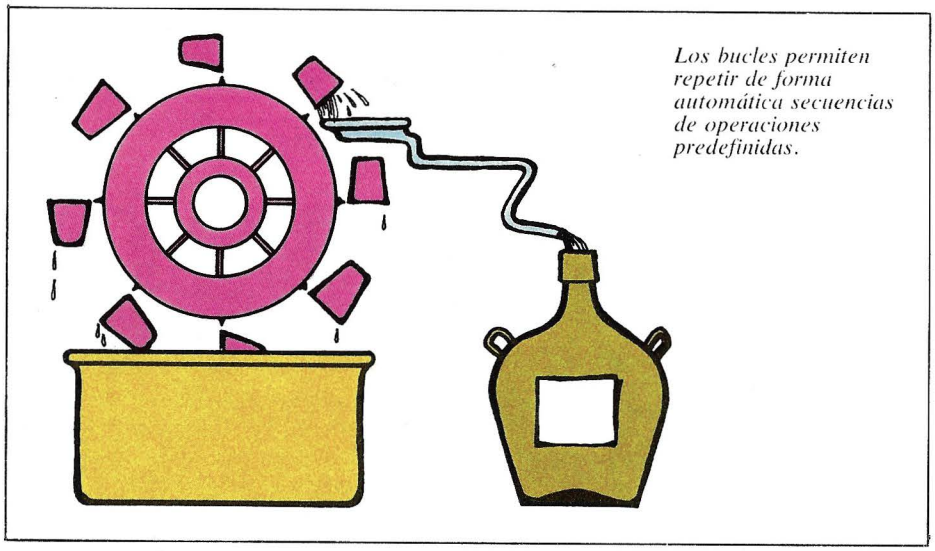
La palabra definida recoge el número situado en la cima de la pila y comprueba si es menor que 10; de ser cierto, incrementa su valor en una unidad, e invoca de nuevo a la propia palabra BUCLE. El proceso se detendrá cuando se alcance un valor superior a 10.

TABLA DE ORDENES FORTH

PALABRA	DESCRIPCION	TIPO
=	Comparador de igualdad.	Operador de comparación.
<	Menor que.	Operador de comparación.
>	Mayor que.	Operador de comparación.
0=	Comprueba si el número es igual a cero.	Operador de comparación.
0<	Comprueba si el número es negativo.	Operador de comparación.
0>	Examina si el número es positivo.	Operador de comparación.
IF/THEN/ELSE	Estructura alternativa que permite elegir entre dos procesos.	Estructura alternativa.
DO/LOOP	Estructura repetitiva que permite ejecutar un proceso un número dado de veces.	Estructura repetitiva.



Los operadores lógicos resultan de gran utilidad para la comparación de pares de elementos. Su resultado puede utilizarse como condición a evaluar para tomar una decisión.



Los bucles permiten repetir de forma automática secuencias de operaciones predefinidas.

```
2 BUCLE CR
2 BUCLE 3 4 5 6 7 8 9 10 FIN OK
```

BUCLES CONTROLADOS

No es imprescindible construir los bucles "a mano", sino que el programador tiene a su alcance algunas estructuras, incluidas en el vocabulario FORTH, que permiten el control de bucles. Tal es el caso de la estructura DO/LOOP, cuya especialidad es hacer que se repita la ejecución de una secuencia de instrucciones un determinado número de veces. Para ello, es necesario prefijar los valores inicial y final del contador de bucles. Ambos números deben depositarse previamente en la pila. El primero a situar en la pila debe ser el correspondiente al valor final y el segundo el valor inicial.

La secuencia de instrucciones a repetir (instrucciones del bucle) debe estar delimitada entre las palabras clave DO y LOOP.

Veamos a continuación un ejemplo; se trata del mismo bucle construido anteriormente, si bien, ahora intervendrán en el mismo las palabras DO/LOOP.

```
: BUCLE <CR>
2 10 2 <CR>
DO <CR>
1+ DUP . <CR>
LOOP <CR>
; <CR>

BUCLE 2 10 2 DO 1+ DUP . LOOP ; OK
```

Y aquí está el resultado de invocar a la palabra BUCLE definida:

```
BUCLE <CR>

BUCLE 3 4 5 6 7 8 9 10 OK
```

Oasis (2)

Gestión del sistema

La base esencial de cualquier actividad informática es la gestión de la información de tal forma que ésta resulte fácilmente inteligible para el usuario; en efecto, éste no debe perder un tiempo precioso en descifrar el método por el cual puede acceder a los datos.

Esta ha sido precisamente la directriz esencial adoptada a la hora de desarrollar el sistema operativo OASIS, teniendo en cuenta que su destino es un ambiente de empresa, donde priman las consideraciones de productividad.

En base a estos criterios de productividad, el sistema operativo OASIS presenta toda una serie de características, propias de sistemas más avanzados y caros, que permiten la gestión de la información en un entorno multiusuario; permitiendo o impidiendo el acceso a la misma según los niveles de prioridad otorgados a cada usuario, o bien llevando a cabo un control de las personas que acceden al sistema, a través de registros históricos. Este último factor permite —en el caso de que el ordenador sea utilizado para realizar trabajos a cargo de otra compañía—, presentar a esta última la correspondiente facturación justificada. Pasemos ahora a revisar más detenidamente todas estas características del sistema operativo OASIS.

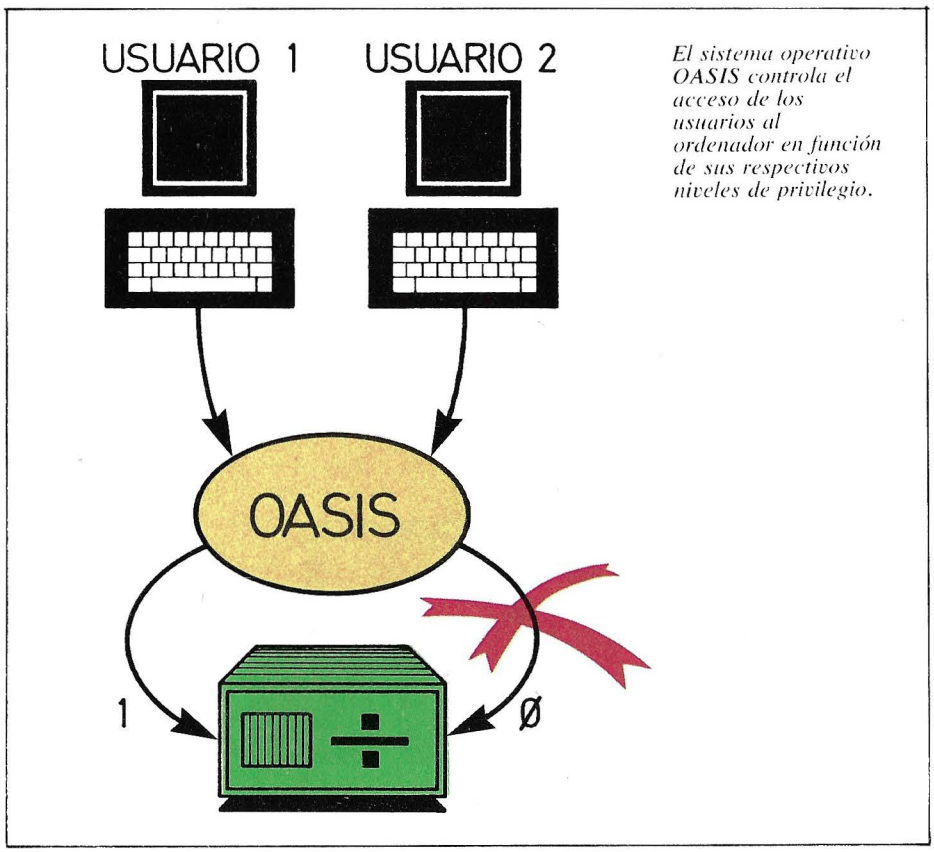
CONTROL DEL SISTEMA OPERATIVO

Las funciones de gobierno del sistema operativo suponen algo más que controlar el acceso del personal a los recursos informáticos del equipo. También representan el establecer unos niveles de acceso a la información, dependiendo de la categoría o de las necesidades específicas de acceso a la misma para garantizar el buen funcionamiento de la empresa.

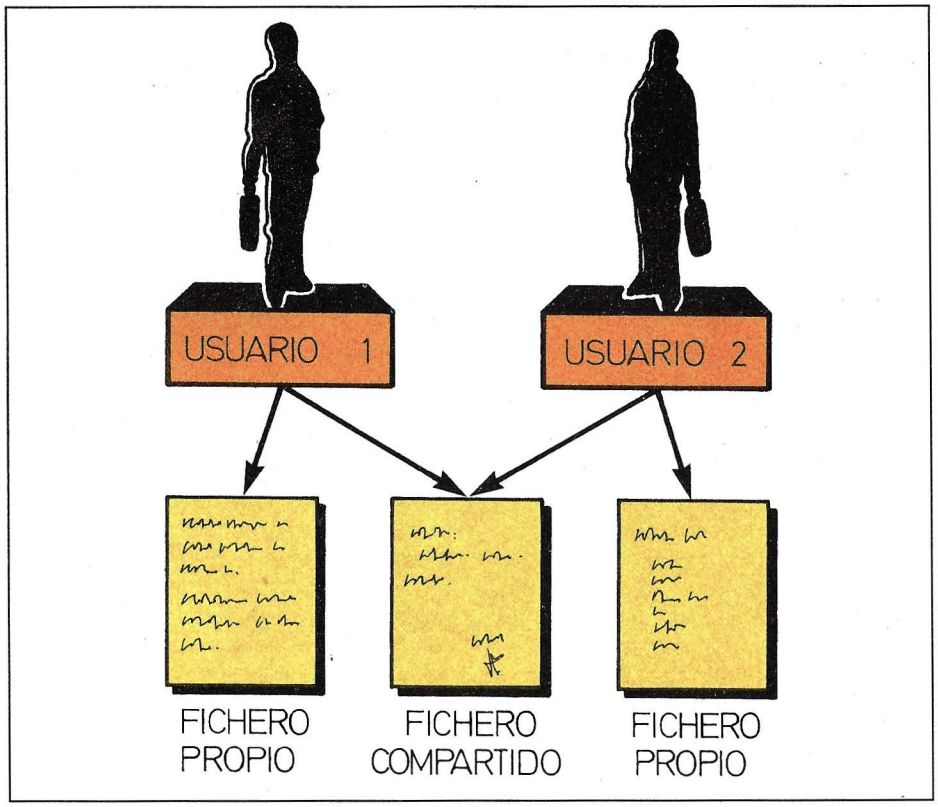
En el sistema operativo OASIS, esta función se lleva a cabo en dos niveles. El primero definiendo una serie de protecciones para los ficheros, de tal forma que cada persona sólo pueda acceder a sus propios ficheros o a otros que se le permitan de forma controlada por el administrador del sistema. El segundo, definiendo una serie de niveles de privilegio o prioridad (hasta cinco niveles), los cuales se asocian al usuario en el instante en el que éste se conecta al sistema. El nivel de prioridad que tenga otorgado permitirá a cada usuario llevar a cabo, o no, una serie de operaciones por medio de los comandos pertinentes. Por lo que respecta al primero de estos métodos, cabe precisar que los ficheros en el sistema operativo OASIS se dividen en tres tipos o categorías: privados, compartidos y del sistema.

Los ficheros privados sólo pueden ser accedidos por la persona que los creó, y siempre de acuerdo con el nivel de privilegio que tenga asignado.

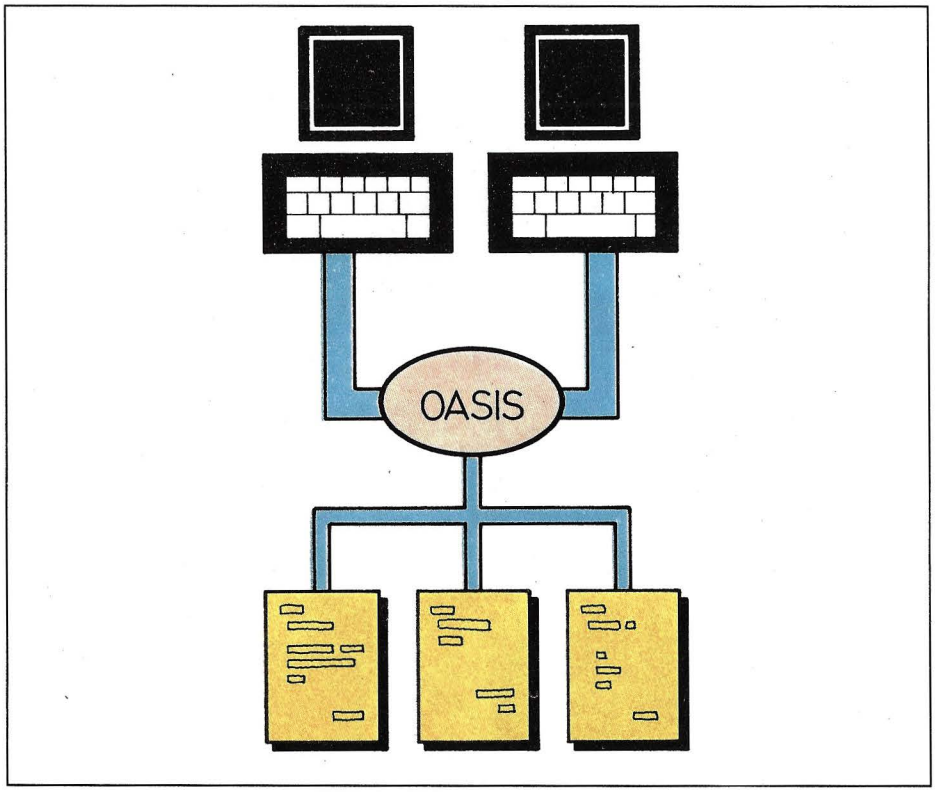
Los ficheros compartidos son creados por un usuario específico; si bien, por necesidad, en función de las características de la información contenida en los mismos, se



El sistema operativo OASIS controla el acceso de los usuarios al ordenador en función de sus respectivos niveles de privilegio.



Una de las peculiaridades del OASIS reside en que admite la existencia de ficheros exclusivos de cada usuario y ficheros compartidos entre varios usuarios.



Los ficheros propios del sistema operativo OASIS resultan accesibles a todos los usuarios conectados al equipo informático.

hace necesario que otras personas tengan acceso a los mismos.

La tercera categoría de ficheros, de acuerdo al criterio establecido, está ocupada por los ficheros del sistema. Estos contienen comandos propios del OASIS y a los cuales acceden los usuarios en función de sus niveles de privilegio.

Para ilustrar este principio de funcionamiento es conveniente proponer un ejemplo. Suponga que existen dos usuarios, inicialmente con el mismo nivel de prioridad, y cuyos ficheros están dirigidos a dos áreas de actividad distintas: Almacén y Ventas, respectivamente. Aparte se encuentran los ficheros propios del sistema. La denominación de los referidos ficheros es la que sigue:

- Ficheros del sistema

BASIC. COMMAND
RUN. COMMAND
EDIT. COMMAND
ERASE. COMMAND
RENAME. COMMAND
FILELIST. COMMAND

- Ficheros usuario 1 (Ventas)

VENTAS. MADRID
VENTAS. MALAGA
VENTAS. BILBAO

- Ficheros usuario 2 (Almacén)

ALMACEN. GENERAL
ALMACEN. PARCIAL 1
ALMACEN. PARCIAL 2

En un principio, los ficheros que corresponden a cada usuario serán de uso exclusivo por cada uno de ellos. No obstante, parece lógico pensar que el usuario 1, a la recepción de un pedido, debe ser capaz de conocer si hay existencias de ese elemento o no; en consecuencia, sería necesario que éste tuviera acceso al fichero del usuario 2 denominado ALMACEN GENERAL. Este fichero sería entonces un fichero compartido por los dos usuarios, mientras que ambos gozarían de idéntico acceso a los ficheros del sistema, puesto que su nivel de privilegio es el mismo.

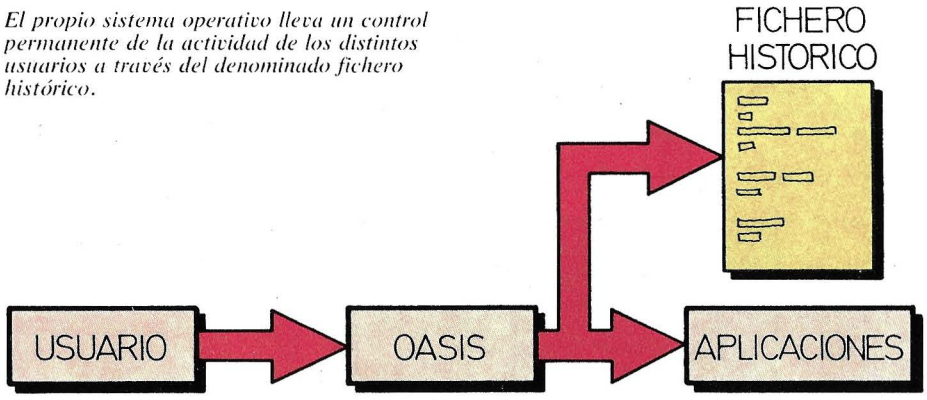
Supongamos ahora que, por cualquier motivo, no es oportuno que el usuario 2 pueda modificar sus ficheros, mientras que el usuario 1 sí ha de poder hacerlo. Ello se consigue con suma facilidad, sin más que modificar los niveles de privilegio de forma que se produzca este hecho. Hay que tener en cuenta que los cinco niveles de privilegio tienen una directa re-

lación con el número y tipo de comandos que el usuario tendrá a su alcance, para hacer uso de los mismos.
Asociadas a todo este proceso de control están las palabras clave que gobiernan el acceso al sistema, y el propio fichero histórico del sistema. Este último, cuyo nombre es SYSTEM HISTORY, almacena en su interior los diferentes registros históricos que se producen cuando un usuario se conecta al sistema; dichos registros almacenan la denominación del usuario que se conectó, así como el tiempo que permaneció conectado; además de otra información adicional y plenamente necesaria, como puede ser la memorización del momento en el que se obtuvo la copia de seguridad de un disco (Backup), o cuando se produjo la carga del sistema.
Normalmente, cuando se llene todo el espacio destinado al fichero histórico, residente en el disco del sistema, el ordenador lo indicará a través de un mensaje al efecto para que los usuarios puedan tomar las medidas oportunas.

EL OASIS COMO SISTEMA MULTIUSUARIO

Se ha mencionado anteriormente que el OASIS puede trabajar como sistema operativo multiusuario, en cuyo caso el núcleo del sistema operativo controla el reloj interno del sistema y asigna tiempo de microprocesador a las diversas tareas a realizar. Ello se realiza de forma tal que, bajo la perspectiva del usuario, el sistema aparece totalmente transparente en sus funciones; como si dicho usuario fuera el único que estuviera accediendo a sus recursos.
El proceso seguido para poner en práctica tal fraccionamiento, consiste en fijar el intervalo de tiempo que se va a destinar a cada tarea, en función de las distintas necesidades de los usuarios. Este cometido lo lleva a cabo el administrador del sistema por medio de la orden SET SLICE.
Una vez precisada semejante división, se inicia el trabajo en modo multiusuario, durante el cual cada usuario accede durante la fracción de tiempo establecida a los recursos de la máquina. Transcurrido este tiempo, el control se transfiere al siguiente usuario, siguiendo un proceso rotativo.

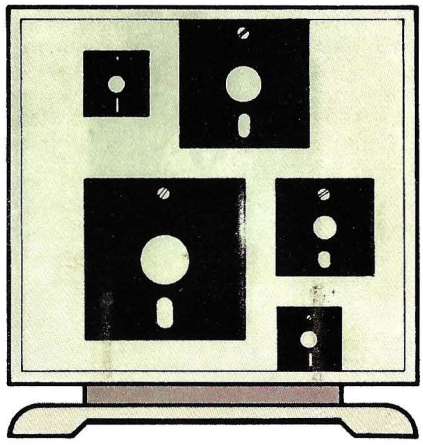
El propio sistema operativo lleva un control permanente de la actividad de los distintos usuarios a través del denominado fichero histórico.

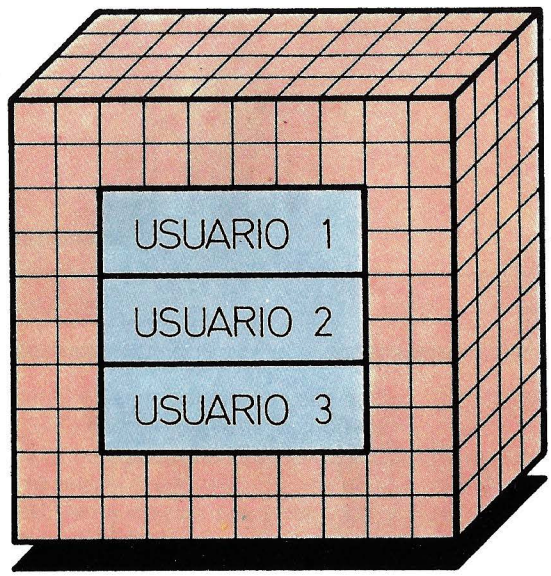


La técnica puesta en práctica garantiza un empleo óptimo de la CPU, toda vez que ésta se encuentra en operación de un modo casi constante. Si en un momento determinado algún usuario está cursando una operación de entrada/salida, el intervalo de tiempo de acceso a CPU que le corresponde sería otorgado a otro usuario asociado al sistema. Hay que tener en cuenta que, por sus características, muchos tipos de operaciones de entrada/salida no consumen tiempo de CPU.
Cuando el ordenador opera en modo multiusuario, su memoria principal se divide en una serie de zonas denominadas "bancos de memoria"; los cuales, a su vez, son subdivididos internamente en particiones de memoria. El espacio en bytes asignado a cada una de estas divisiones es variable y, en cualquier caso, definible por el usuario dentro de unos ciertos valores admisibles por el sistema.

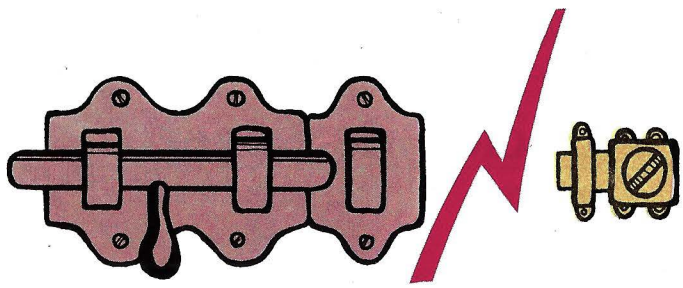
La distribución de ficheros en disquete

Cuando un usuario crea, copia, o borra un fichero en un disquete, se desencadena un proceso variable dependiente de la orden que ejecute. Sin embargo, existe una estrecha relación entre las tres operaciones; una relación fundamentada en el uso que hacen del espacio disponible en el disco.
Cuando se parte de un disquete virgen y se van creando ficheros sobre el mismo, estos ficheros se irán ordenando uno tras otro sin dejar entre ellos espacios vacíos. Ello significa que el tamaño máximo de cualquier fichero que se desee introducir posteriormente, coincidirá con la diferencia entre la capacidad total de almacenamiento del disquete y el espacio ya ocupado por otros ficheros. Cuando se borra un fichero, el sistema operativo toma el área de memoria ocupada por éste y la pone a disposición de otros datos, aunque no procede a ninguna reordenación de los ficheros sobre el disco. Ello se traduce en que el tamaño máximo de cualquier fichero que desee introducirse quedará limitado al espacio de la mayor área libre que se encuentre entre los ficheros residentes, y no por la diferencia entre "área utilizada" y "área total de memoria".
De ello se desprende que un disquete puede contener muy pocos ficheros y de pequeño tamaño, con lo que teóricamente el espacio de memoria disponible será grande y, sin embargo, por estar estos ficheros espaciados por el disquete, el tamaño máximo del fichero almacenable puede resultar muy inferior.

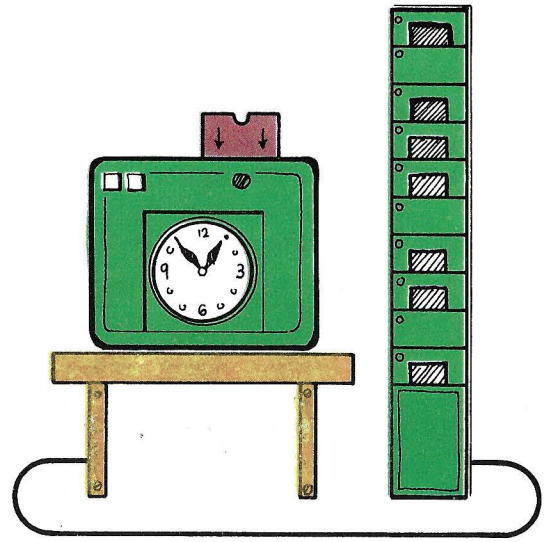




Los recursos de memoria son divididos por el OASIS entre los diversos usuarios conectados. A su vez, las zonas destinadas a los usuarios son fraccionadas por el sistema operativo en particiones de tamaño variable.



El OASIS permite definir distintos niveles de protección de los datos, lo cual limita el acceso a la información puesta en juego en el ordenador.



A través del fichero histórico del sistema, el OASIS mantiene un pleno y detallado control de la actividad de los distintos usuarios con el ordenador.

Un punto fundamental en la protección de cualquier sistema multiusuario reside en la protección de los registros de los ficheros frente al acceso simultáneo por parte de varios usuarios. En efecto, suponga un hipotético caso en el que dos usuarios, sin conocimiento del hecho debido a la transferencia del sistema operativo, acceden a la información contenida en un registro, información que supondremos de tipo numérico. Esta información es transportada a sus respectivas particiones de memoria, y procesada, con lo cual se altera su valor y, finalmente, es devuelta a su registro de origen. Si no existiera ningún tipo de protección, el dato que quedaría almacenado sería el correspondiente al usuario que devolviera en último lugar el dato de los dos que lo leyeron simultáneamente.

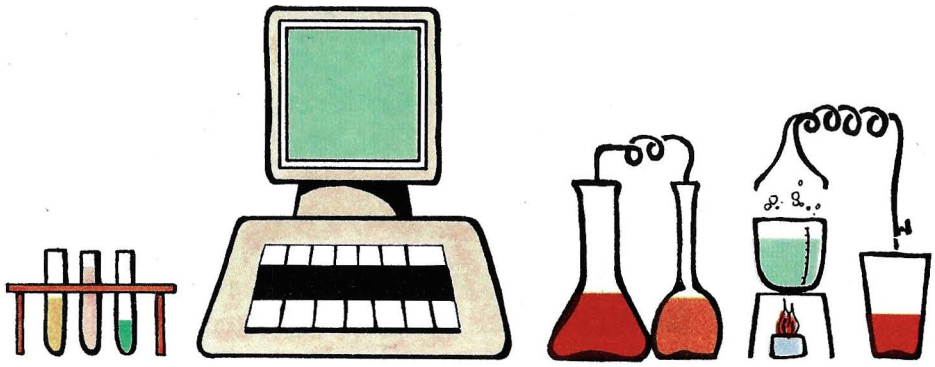
Para obtener una idea de los problemas a que ello puede conducir, cabe imaginar que el dato en cuestión coincide con las existencias de un determinado producto, y las operaciones realizadas por los usuarios son actualizaciones por pedidos. Si el valor inicial del dato es 10 unidades y las operaciones son la sustracción de 8 y 7 unidades por haberse producido pedidos por este valor, puede suceder que el dato actualizado adopte el valor 2 ó 3 unidades, dando la sensación de que todavía hay existencias cuando la realidad es que uno de los dos pedidos no puede cumplimentarse por falta de elementos.

El sistema operativo OASIS solventa este problema por medio de bloqueos de información de dos tipos: uno a nivel de fichero y otro a nivel de registro. En el primero de los casos, el primer usuario que accede a un fichero se adueña del mismo, y en tanto no finaliza sus operaciones con el mismo ninguna otra persona puede tener acceso. Evidentemente, este procedimiento es interesante en el caso de que se trate de ficheros de escasa frecuencia de acceso, pues de lo contrario produciría retrasos inaceptables a los usuarios. El otro método que brinda el sistema operativo OASIS es el bloqueo a nivel de registro, más adecuado para ficheros de acceso frecuente. Se trata de un método por el cual el usuario que primero accede a un registro de un fichero provoca un bloqueo del mismo, de tal forma que el siguiente usuario que efectúe el acceso no podrá modificar la información contenida mientras que el anterior usuario no haya finalizado sus operaciones y libere el registro.

Software científico/técnico

Un abanico de programas especializados

Sin llegar a profundizar en un paquete concreto, a lo largo del presente capítulo vamos a realizar una descripción de las principales características del software científico/técnico. Es muy amplia la variedad de programas para ordenadores personales que pueden clasificarse dentro de esta categoría de aplicaciones. No obstante, por su propia naturaleza y complejidad, suele ser necesario en muchos casos proceder a un desarrollo de los programas «a medida» y específicos para el problema que se pretende mecanizar. En cualquier caso, antes de acometer el desarrollo de una «aplicación privada», es recomendable hacer una incursión en las listas de programas comercializados por si hubiera alguno adecuado a las especificaciones iniciales.



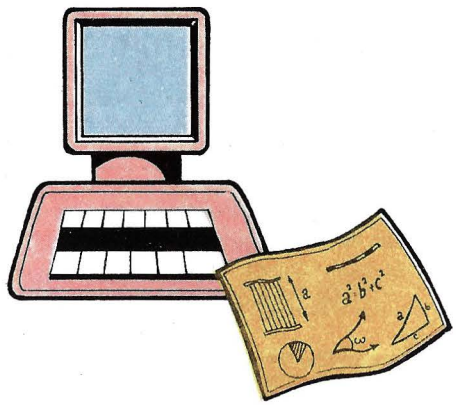
Los programas científicos tienen como principal característica su utilización en proyectos de investigación sobre diversas materias: química, física, matemáticas...

¿QUE SE ENTIENDE POR SOFTWARE CIENTIFICO/TECNICO?

Dentro de la anarquía terminológica que existe en Informática, uno de los conceptos para el que se pueden encontrar las más diversas definiciones es precisamente el que estamos estudiando. Antes de comenzar con la exposición de las características de este tipo de programas, vamos a definir lo que, en este caso, entendemos para cada uno de los dos conceptos manejados:

- *Programa científico*

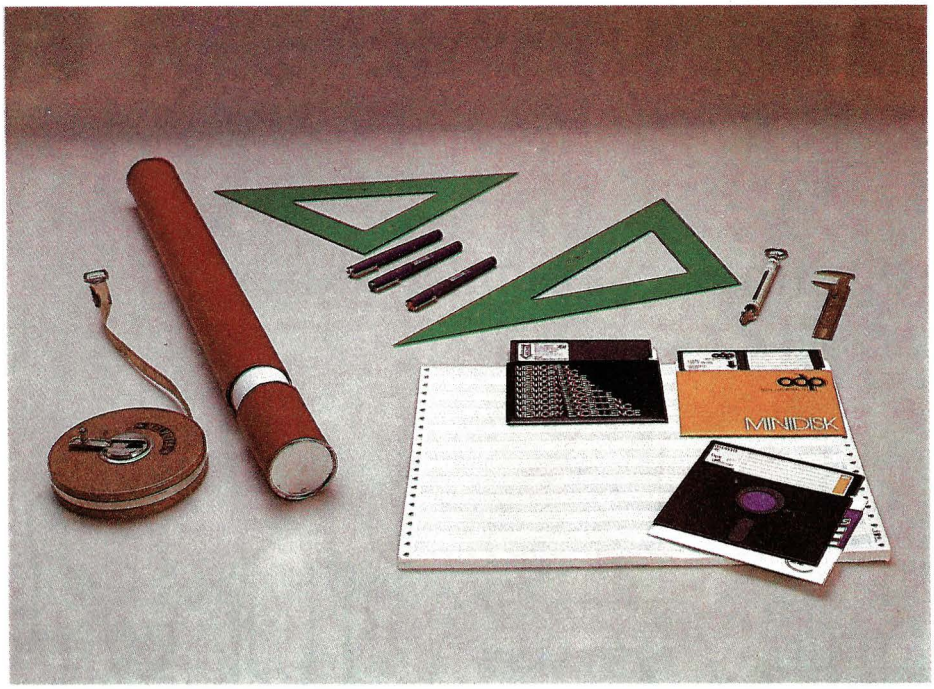
Un programa es científico cuando su empleo está orientado a proyectos de desarrollo o investigación en cualquier disciplina científica. Este tipo de programas no es frecuente encontrarlos comercializados, ya que su uso queda restringido a un reducido número de usuarios; además, la complejidad inherente a la resolución de este tipo de problemas hace que el desarrollo de los programas acostumbre a realizarlo el propio investigador.



... en cambio, los programas técnicos suelen ser utilizados en trabajos mas convencionales y concretos.

- *Programa técnico*

Un programa se califica como técnico cuando su misión consiste en dar servicio a especialistas en diversas profesiones. En este caso, la diversificación de los temas tratados y el gran número de usuarios potenciales, ha incitado a muchas empresas de software a desarrollar programas especializados en cada materia. Para completar esta definición inicial sólo queda por establecer una división dentro de los programas técnicos, adoptando como criterio el tipo de uso que se le dará al programa. Así, podemos hablar de programas técnicos de gestión y de programas técnicos específicos. Los primeros son aquellos cuya misión es de carácter interdisciplinario y, por lo tanto, servirán para cualquier tipo de técnico; por ejemplo: una base de datos puede servir tanto para que un médico mantenga un historial clínico de sus pacientes, como para que un abogado archive normativas y leyes. En cambio, los segundos tienen una utilidad restringida para los profesionales a los que se dedican; así, por ejemplo, un programa para el cálculo de estructuras de edificios sólo será útil a los arquitectos e ingenieros civiles y, por consiguiente, es catalogable como programa técnico específico.



La propia naturaleza y complejidad de esta categoría de aplicaciones obliga, en muchos casos, al desarrollo de los programas científico/técnicos «a medida».

CARACTERISTICAS GENERALES DEL SOFTWARE CIENTIFICO

Como ya adelantábamos, este tipo de programas se utiliza generalmente en trabajos de investigación. En algunos casos, un programa científico, una vez finalizado y probado, puede estandarizarse y generalizarse su uso. Si esto sucede, atendiendo a la calificación establecida en el párrafo anterior, habría que dejar de clasificarlo como científico y trasladarlo al grupo de programas técnicos específicos.
En otros casos, esta circunstancia no ocurre jamás y el programa presta sus servicios mientras dure el proyecto de investigación, dejando de explotarse al finalizar éste. En resumen cabe afirmar que la vida de un programa científico está ligado a la vida de un proyecto de investigación, y sólo sobrevive y se comercializa si, por sus características, su uso es generalizable.
En cuanto a las características técnicas de estos programas podemos citar las siguientes:

- *Presentación*

Dado que su uso se limitará a un reducido número de personas, no suele cuidarse demasiado la calidad estética de los resultados. En cambio, sí resulta de vital importancia garantizar su precisión.

- *Datos manejados*

En cuanto al volumen y naturaleza de los datos manejados por este tipo de programas, no es posible generalizar ninguna propiedad para los datos de entrada. En algunos casos, cuando el proyecto sea de tipo teórico, el número de datos de entrada será muy reducido aunque de naturaleza compleja; en cambio, cuando el proyecto sea de tipo práctico, será necesario tratar grandes masas de datos de tipo convencional.
Donde sí se puede establecer una característica mayoritaria es para los datos de salida: con algunas excepciones, su volumen suele ser reducido.

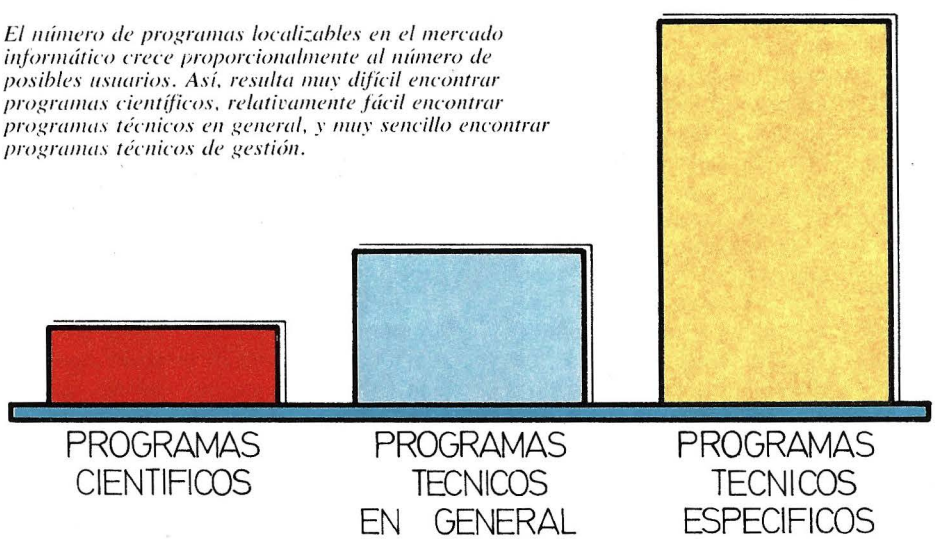
- *Metodología*

Los programas de tipo científico suelen estar sometidos a modificaciones continuas que le vienen impuestas por los resultados parciales del proyecto. En consecuencia, resulta de vital importancia comentarlos convenientemente y mantener una documentación adicional para facilitar las sucesivas revisiones a las que estarán sometidos.

- *Software de apoyo*

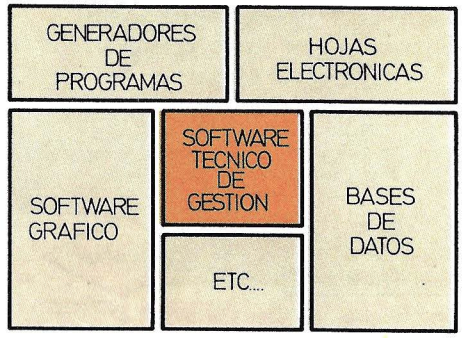
Evidentemente resulta de vital importancia disponer de un compilador de alto nivel que facilite la programación. Los lenguajes utilizados pueden ser varios, y dependerá de los objetivos perseguidos la elección de uno u otro. En general, los más utilizados suelen ser FORTRAN, PASCAL y BASIC; este último más por ser el único lenguaje disponible en algunos casos que por resultar el idóneo.

El número de programas localizables en el mercado informático crece proporcionalmente al número de posibles usuarios. Así, resulta muy difícil encontrar programas científicos, relativamente fácil encontrar programas técnicos en general, y muy sencillo encontrar programas técnicos de gestión.

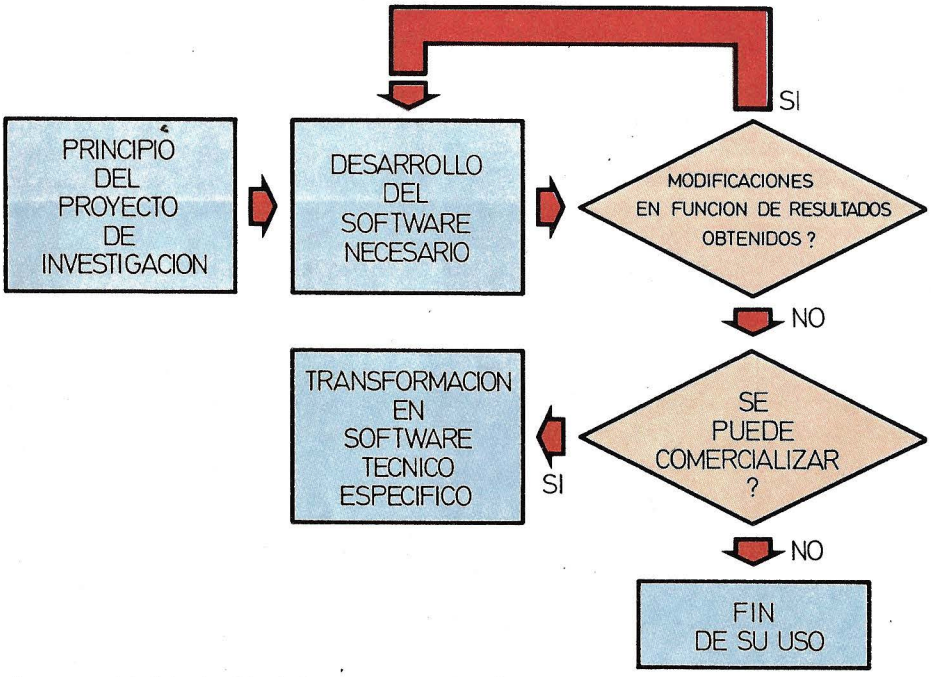


CARACTERISTICAS GENERALES DEL SOFTWARE TECNICO DE GESTION

Acerca de este tipo de programas no es posible establecer ningún tipo de características propias y exclusivas. En realidad, se puede afirmar que la mayor parte de los programas estudiados en esta sección de la obra forman parte del software técnico de gestión: bases de datos, hojas electrónicas, software gráfico, generadores de programas, etc. Por lo tanto, la única condición indispensable para poder afirmar que un programa pertenece a este grupo es que sea susceptible de ser utilizado por un técnico de cualquier especialidad.



Cualquiera de las aplicaciones de tipo horizontal tiene también una gran utilidad en el terreno técnico y científico. Incluso el grupo del software técnico de gestión queda también abierto a innumerables ámbitos de actividad.



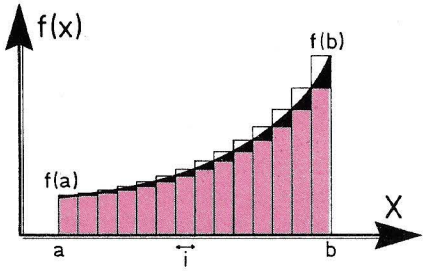
Resumen del ciclo de vida de los programas científicos.

CARACTERISTICAS GENERALES DEL SOFTWARE TECNICO ESPECIFICO

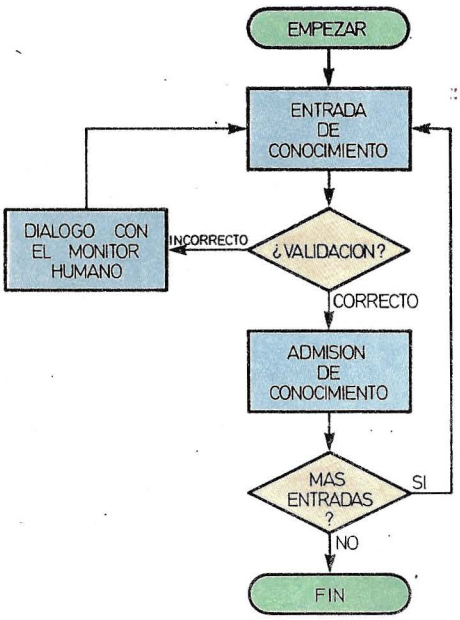
Dentro de este tipo de software, la diversidad de aplicaciones resulta tan grande que no cabe hablar de características comunes a todos. En cambio, sí se encuen-

Integración de funciones

Como ejemplo de sencillo programa científico técnico podemos citar la integración de funciones continuas y definidas. El objetivo del programa es integrar una función f(x) entre dos puntos a y b del eje de abscisas. Dicho de otra forma: calcular la superficie comprendida entre f(x) y el eje de las «equis» desde x = a hasta x = b.
El método más sencillo para resolver informáticamente la resolución de una integral consiste en sustituirla por un sumatorio. Para ello se elegirá una longitud de intervalo «i» y se sumarán las superficies de todos los rectángulos de base «i» y altura según el punto de intersección con f(x). Evidentemente, con este método se perderá precisión, puesto que se desprecian los triángulos situados entre f(x) y la superficie sumada; no obstante, si se toma un valor muy bajo para el intervalo «i», el error cometido será mínimo.
Una última consideración sobre este programa consiste en que los rectángulos pueden tomarse por debajo o por encima de la función. En el primer caso se obtiene lo que podemos denominar integral inferior, mientras que en el segundo se obtiene la integral superior.
En la ilustración que se acompaña se puede apreciar el error cometido en la integral superior sumando los triángulos blancos.
También se incluye un programa escrito en BASIC que resuelve la integral inferior de una función f(x) que, por supuesto, debe ser programada como función o rutina dentro del programa.




```
10 LET SUM = 0
20 INPUT A
30 INPUT B
40 INPUT I
50 LET X1 = A
60 LET X2 = X1 + I
70 IF X2 ≥ B THEN GOTO 110
80 LET SUM = SUM + I * f(X1)
90 LET X1 = X2
100 GOTO 60
110 LET SUM = SUM + (B-X1) * f(X1)
120 PRINT SUM
130 STOP
```



Esquema resumido de la fase de aprendizaje de un programa experto. El propio diagrama evidencia la total importancia que tiene para el programa la presencia de un buen tutor humano.

tran paralelismos cuando nos limitamos al software específico para un determinado grupo de técnicos. A continuación se detallan algunos —todos resultaría imposible— de los más extendidos entre este tipo de programas.

- *Software aplicado a la Ingeniería*

Tal vez sea este uno de los colectivos de técnicos más necesitados de apoyo informático: por un lado sus necesidades de cálculo son muy grandes, y por otro el volumen de datos manejados también es muy elevado. Al asociar esta necesidad de apoyo al hecho de que los procesos de cálculo realizados por un ingeniero suelen ser muy repetitivos y de tipo general, resulta que tenemos las condiciones ideales para el desarrollo de programas técnicos específicos. De hecho, existen programas comercializados para casi todas las especialidades de la Ingeniería.

- *Software aplicado a la Medicina*

Dado que este apartado se centra en el software técnico específico, no cabe incluir aquí programas de gestión que podrían servir de apoyo a un médico, sino tan sólo a aquellos encuadrables en el «software experto».
Cabe definir como programa experto a todo aquel que es capaz de asimilar conocimientos e inferir nuevos conocimientos a partir del mismo. Dentro de este grupo de programas existe uno, especialmente destacado, dedicado a «conversar» con uno o varios médicos sobre un paciente. Su nombre es MYCIN y, muy sucintamente, su funcionamiento se esquematiza en los siguientes pasos:

1. El programa admite que el médico le aporte conocimientos en forma de reglas (por ejemplo: «*si* la fiebre es alta *y* el número de glóbulos blancos es superior a 10.000, *entonces* posible infección»).
2. A continuación MYCIN valida la nueva regla con las que ya conocía y detecta posibles incongruencias. En el caso de que así suceda «dialoga» con el médico para aclarar el mal entendido.
3. Una vez almacenado suficiente conocimiento, el programa admite la introducción de datos relativos a un paciente e intenta diagnosticar en función no sólo de las reglas que conoce, sino también de las consecuencias de dichas reglas.

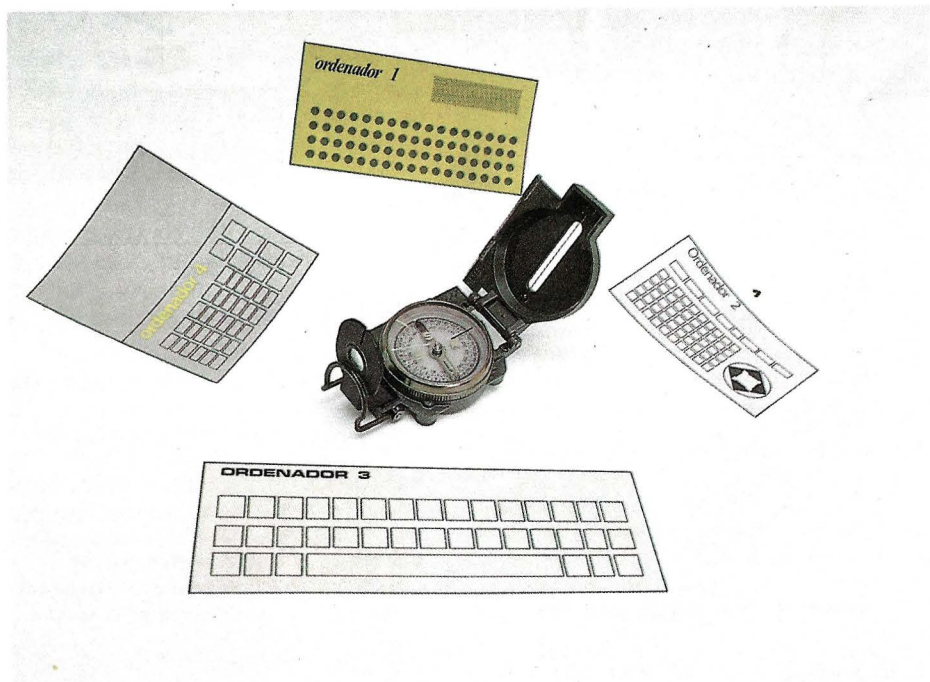
Se ha elegido como ejemplo de sistema experto a MYCIN por ser uno de los más antiguos y contrastados, no obstante se pueden encontrar programas expertos en muchas otras técnicas.

- *Software aplicado a la Economía*

Otro colectivo de profesionales para los que existe una gran variedad de programas técnicos específicos es el de los economistas. Al igual que en el caso de los ingenieros, prácticamente todas las especialidades pueden apoyarse en programas comercializados. Así, es posible encontrar desde sencillos programas destinados a la declaración de la renta de personas físicas o empresas, hasta complicados programas de econometría: sin olvidar a los programas de contabilidad financiera y analítica. Tal vez, los programas sobre contabilidad constituyan el tipo de software con más variedad que existe en el mercado.

- *Software aplicado a la Educación*

Aunque ya ha sido objeto de tratamiento en otros capítulos de la obra, no podemos dejar de citar al software educativo como uno de los grupos de programas técnicos específicos más desarrollados. Existen infinidad de posibilidades para aplicar la informática a la educación y, probablemente, antes de que finalice este siglo, los ordenadores tendrán un papel fundamental en cualquier tipo de enseñanza.



Los ordenadores utilizados para la ejecución de programas de tipo científico pueden ser de muy distinto tipo. En general, un ordenador personal puede dar suficiente «juego», aunque en algunos casos resulta indispensable utilizar grandes ordenadores.

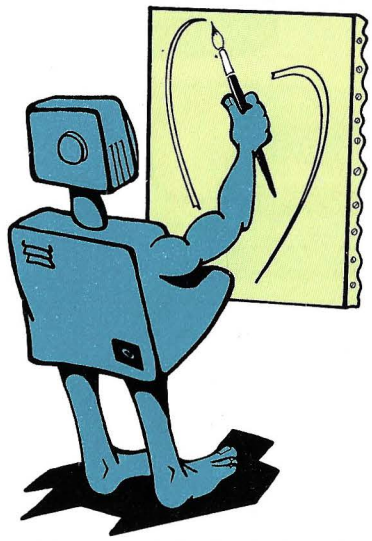
Gráficos en BASIC

Introducción al dibujo en la pantalla desde el BASIC

Una de las características más relevantes de los modernos ordenadores es su capacidad para crear gráficos. La representación gráfica de datos ayuda a su asimilación y a su mejor comprensión. Resulta más adecuado presentar las ventas de una empresa en un gráfico de barras que en una lista de números. De esta forma es mucho más sencillo observar la trayectoria de un simple vistazo.

Los gráficos por ordenador son imprescindibles en una larga serie de actividades: desde el diseño a los juegos, pasando por la educación. En general, una buena presentación hace más agradable el trabajo con el ordenador. Este es el motivo por el cual los fabricantes de ordenadores se preocupan cada vez más de las posibilidades gráficas. Los más novedosos aparatos incorporan funciones impensables hace tan sólo unos años.

Desgraciadamente, la forma de tratar las posibilidades gráficas es muy diferente en cada ordenador. Cada fabricante ha tratado de mejorar las prestaciones que ofrecía la competencia, diversificándose así el modo de acometer esta materia. A lo largo del presente capítulo introductorio se seguirá la norma establecida por Microsoft; filosofía ésta extensiva a un amplio número de aparatos. En los restantes casos, los comandos BASIC se asemejan a los del BASIC-Microsoft, variando en algunas de sus características. Estas diferencias se indicarán a lo largo del texto y en la tabla de compatibilización que lo acompaña.



Los gráficos: una de las facultades más espectaculares del ordenador.

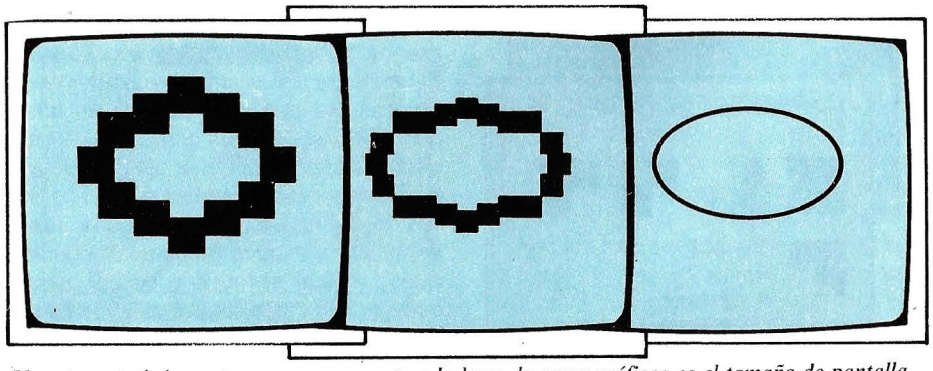
RESOLUCION Y MODOS GRAFICOS

La característica que más influye en la diversidad de formas de crear gráficos es el "tamaño" de la pantalla. Las imágenes que aparecen en ésta, están formadas por un número de líneas horizontales. Cada una de estas líneas, a su vez, agrupa un número limitado de puntos. La imagen total se compone por la iluminación de los puntos adecuados. De ello se deduce que, a mayor cantidad de puntos, mejor definición de la imagen.

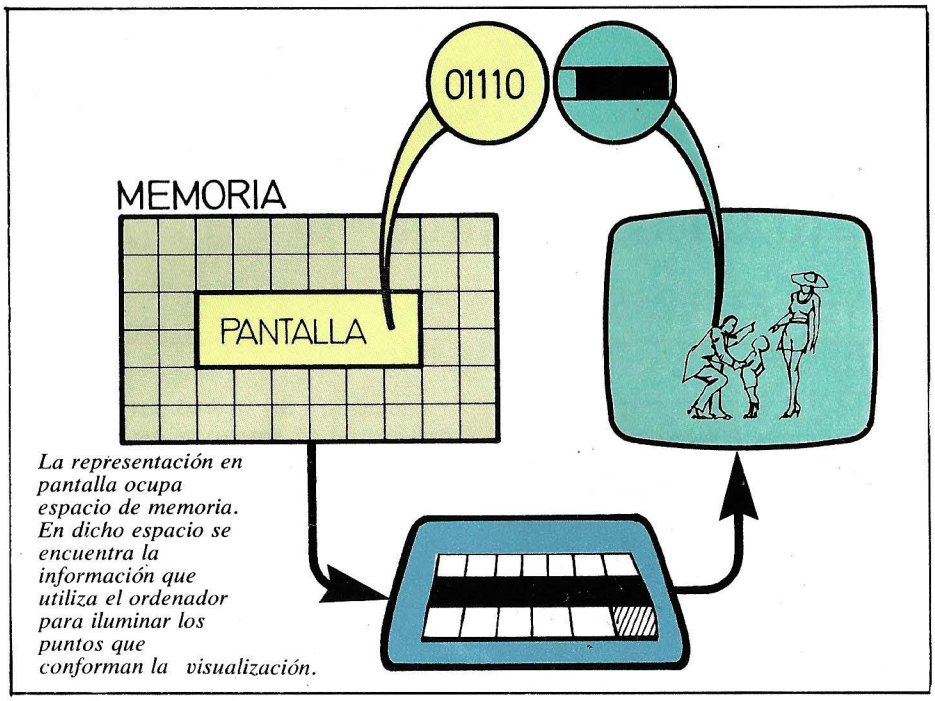
El número de puntos que es capaz de representar un ordenador se denomina "resolución" del mismo. La resolución máxima de un aparato se especifica mediante dos cantidades: una representa el número de líneas y la otra indica los puntos de cada línea. Así, una resolución de 192×256 significa que se tienen 192 líneas de 256 puntos. En ese caso se dispondría de un número total de puntos igual a 49152 (192*256=49152). Estos puntos suelen denominarse también "pixels" (contracción de PIcture ELementS).

Para la representación en pantalla, el ordenador hace uso de una parte de la memoria en la que se almacena el estado actual de cada punto. Si la imagen se va a representar en blanco y negro, se necesitará un bit por pixel. De esta forma, el bit a uno indica que el correspondiente punto está encendido. Un circuito específico del ordenador se encarga de leer la referida memoria de pantalla y trasportar su contenido a la pantalla. Conociendo la ubicación exacta de la zona de memoria de pantalla, ésta puede ser alterada con el uso de comandos POKE. Sin embargo, éste es un método definitivamente arduo y tedioso. En la mayoría de los casos, dicha alteración se ve facilitada por el empleo de los comandos gráficos del BASIC. Estos comandos son los que se describen en las páginas de este capítulo.

La necesidad de una zona de memoria



Una característica a tener muy en cuenta a la hora de crear gráficos es el tamaño de pantalla, medido en puntos luminosos, con el que trabaja el ordenador.



La representación en pantalla ocupa espacio de memoria. En dicho espacio se encuentra la información que utiliza el ordenador para iluminar los puntos que conforman la visualización.

dedicada a la representación en pantalla tiene otras implicaciones. Volvamos al ejemplo anterior. El tamaño de pantalla mencionado exige contar con 49152 bits. Un sencillo cálculo revela que ello representa un total de 6144 bytes (49152/8= =6144). O lo que es lo mismo, 6144/1024=6 Kbytes. Esta es la ocupación de memoria en el caso de utilizar tan sólo dos colores (blanco=1, negro=0). Si se desean representar más colores será necesario contar con más memoria. Para cuatro colores no bastará con un bit por pixel; en tal caso serán necesarios al menos dos bits por cada punto a representar (00=negro, 01=rojo, 10=azul y 11=amarillo, por ejemplo). El resultado supone duplicar el espacio de memoria de pantalla para conservar la resolución. Las soluciones tomadas por los diseñadores ante este problema son diversas.

En algunos aparatos la resolución y la memoria de pantalla son fijas. En otros se flexibiliza una de estas características para hacer un mejor uso de los recursos del aparato. En estos últimos casos se puede optar por dos métodos. Por una parte se puede aumentar la ocupación de memoria cuando se necesite más resolución o más colores. El otro método consiste en asignar un espacio fijo de memoria y jugar con la resolución y el número de colores. Este último implica una relación inversa entre resolución y colores: a mayor resolución menos colores disponibles, y viceversa.

El hecho de poder variar las características gráficas del aparato introduce un nuevo concepto: los modos gráficos. Los ordenadores que permiten al usuario elegir la resolución adecuada (dentro de unos límites) disponen de varios modos de representación o modos gráficos. El operador puede indicar el modo adecuado para la actividad en curso mediante un comando; comando que adopta formatos muy diferentes dependiendo del ordenador en cuestión. De ello se hablará más adelante.

Algunos ordenadores poseen un repertorio de caracteres gráficos. Estos servirán para crear algunos dibujos elementales.

BAJA RESOLUCION, CARACTERES GRAFICOS

Se puede considerar como baja resolución la obtenida mediante el uso de caracteres. Al hablar del comando PRINT y sus variantes, se subdividió a la pantalla en un determinado número de líneas y columnas. Estas indican la cantidad de texto representable en pantalla a un mismo tiempo. Ambos datos (número de líneas y de columnas) indican la "resolución en modo texto". Utilizando algunos caracteres especiales (como puede ser el asterisco o los signos + y −) es posible llegar a construir gráficos elementales. Para ello se hará uso del comando PRINT AT (LOCATE, u otro similar). De esta forma se pueden obtener gráficos sin mucho detalle.

Algunos aparatos brindan un repertorio de caracteres denominados "caracteres gráficos". Estos son bloques especiales como cuadrados, líneas, arcos de circunferencia... con los que se pueden realizar algunos gráficos utilizando el mismo método que con los caracteres normales. La creación de dibujos con estos caracteres gráficos es similar a la construcción de una maqueta con piezas o bloques elementales. Con un poco de práctica se pueden realizar algunos dibujos, aunque con limitaciones.

PRIMEROS PASOS EN ALTA RESOLUCION

La pantalla de alta resolución se puede concebir como un mosaico de puntos. Cada uno de esos diminutos puntos es un pixel. Y cada pixel puede ser identificado por sus dos coordenadas: vertical y horizontal.

El origen de estas coordenadas (el punto 0,0) variará de unos aparatos a otros. Por lo general, el origen se sitúa en una de las esquinas de la parte izquierda de la pantalla. El BASIC de Microsoft utiliza el ángulo superior izquierdo (éste será el que se emplee aquí); no obstante, será conveniente revisar en el manual de cada ordenador para la correcta situación de este punto.

Una vez identificado el origen, el primer

paso consiste en iluminar uno de los pixels. A tal efecto se hace uso del comando PSET (en otros dialectos se emplea PLOT). Dicho comando necesitará las coordenadas para identificar el punto a iluminar. Estas se proporcionan en su argumento: primero la coordenada horizontal (X) y luego la vertical (Y). Este es el aspecto que muestra su formato:

(Número de línea) PSET (<X>,<Y>) [,<color>]

El tercer dato, suministrado opcionalmente, indica el color que ha de tomar el punto en cuestión. En aquellos ordenadores que disponen de varios modos de representación en pantalla, será necesario indicar el modo antes de ejecutar el comando PSET. Si no se hace así, se estará tratando de dibujar en la pantalla de texto, lo que puede producir un error de sintaxis. Si al ejecutar un comando PSET (o PLOT) aparece un mensaje de error, ello indicará que hay que ejecutar un comando de cambio de modo.
Variando el argumento de PSET se consiguen dibujar diferentes puntos. Siempre hay que prestar atención para no exceder los límites de la pantalla. En el caso de una pantalla de 192×256 pixels, la coordenada X puede variar entre 0 y 255, mientras que la vertical no debe exceder del valor 191. Con la resolución mencionada se puede crear el primer programa gráfico.

```
10 CLS
20 FOR I=1 TO 1000
30 PSET (RND*191,RND*255)
40 NEXT I
■
```

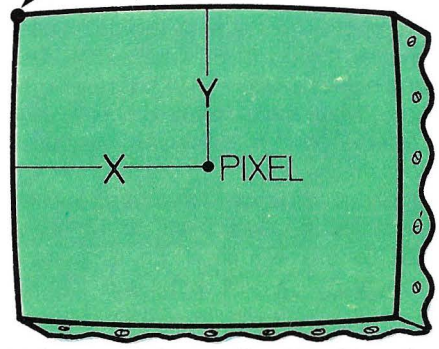
Este pequeño programa enciende puntos en lugares aleatorios de la pantalla. Al ejecutarlo se consigue un bonito cielo estrellado (si el fondo está en negro). Los ordenadores que utilizan varios modos de presentación gráfica necesitarán una línea previa en la que se especifique el modo gráfico adecuado. En algunos de estos aparatos se vuelve a la pantalla de texto al finalizar la ejecución del programa. Ello

PSET

Ilumina el punto de la pantalla indicado por sus coordenadas.

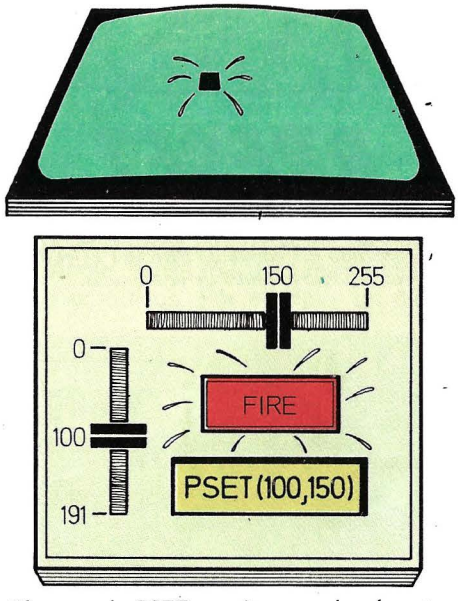
Formato: PSET [STEP] (<X>, <Y>) [,<color]

Ejemplos: 10 PSET (100,150)
50 PSET STEP (10,−10)



Cada punto de la pantalla o "pixel" queda plenamente identificado por medio de sus coordenadas.

hará que desaparezcan los dibujos y se permita la visualización de caracteres. El programa anterior tomaría la siguiente forma en el BASIC de Microsoft.



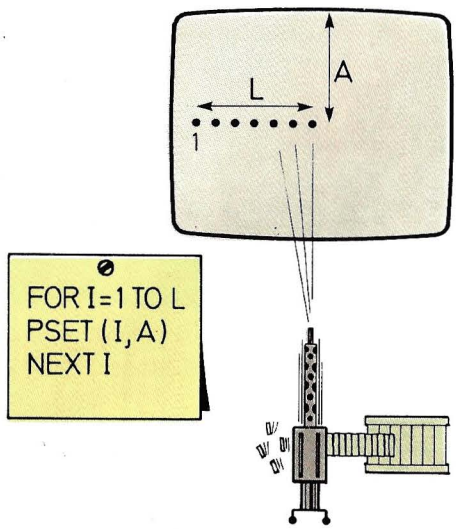
El comando PSET permite encender el punto de la pantalla que desee el usuario, sin más que especificar las coordenadas del mismo.

```
 5 SCREEN 2
10 CLS
20 FOR I=1 TO 1000
30 PSET (RND*191,RND*255)
40 NEXT I
50 GOTO 50
```

La línea 5 especifica el modo gráfico. En este caso se trata del modo de alta resolución. El comando empleado es SCREEN, el propio de Microsoft. Para mantener la imagen en pantalla se hace uso de la línea 50. Esta evita que se detenga la ejecución y se pierda el dibujo.
Con la ayuda de PSET (Point SET, poner punto) se pueden crear dibujos más complejos. Por ejemplo, se podría dibujar una línea; a fin de cuentas, una línea no es más que una sucesión de puntos.
El siguiente ejemplo muestra cómo trazar una línea horizontal.

```
10 INPUT "ALTURA";A
20 INPUT "LONGITUD";L
30 SCREEN 2
40 FOR I=1 TO L
50 PSET (I,A)
60 NEXT I
70 GOTO 70
■
```

Para dibujar la línea se ha tomado un dato fijo para la altura. Esta altura indica la coordenada Y de los puntos de la recta. Al tratarse de una línea horizontal, todos sus puntos tendrán la misma coordenada vertical; esta coordenada la introduce el operador en la línea 10. El otro dato que se le pide al usuario es la longitud del segmento a trazar. Este segundo dato se almacena en la variable L (línea 20). Ambos datos han de ser enteros, ya que el orde-



Con el uso reiterado de PSET se pueden trazar líneas. Hay que recordar que, a fin de cuentas, una línea no es más que una sucesión de puntos.

nador no puede trabajar con fracciones de pixel. Además, los datos proporcionados han de estar comprendidos dentro de los límites de la pantalla. De no ser así se produciría un error.

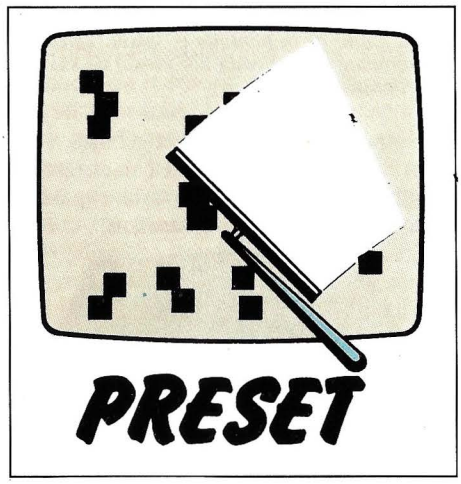
Las líneas de programa 40, 50 y 60 son las encargadas de representar la línea, cuyos extremos coincidirán con los puntos de coordenadas (1, A) y (L, A). El bucle FOR es el que se ocupa de incrementar la coordenada X desde el valor inicial al final.

Siguiendo este mismo método se pueden crear todo tipo de líneas. El trazado de líneas oblicuas (que no sean horizontales ni verticales) resulta un poco más complejo. Veamos un ejemplo:

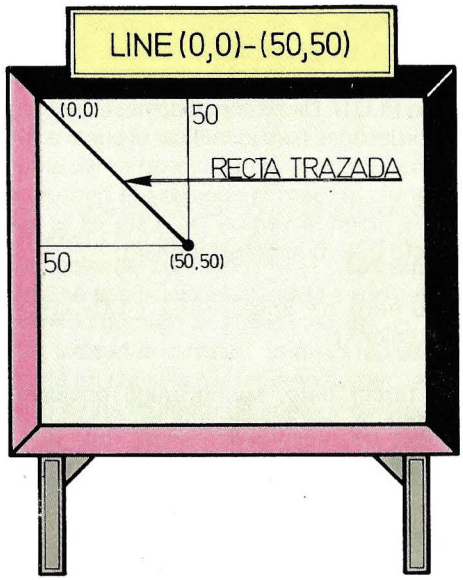
```
10 INPUT "X1";X1
20 INPUT "Y1";Y1
30 INPUT "X2";X2
40 INPUT "Y2";Y2
50 SCREEN 2
60 IF Y2-Y1>X2-X1 THEN M=Y2-Y1 ELSE
   M=X2-X1
70 FOR I=0 TO M
80 PSET (X1+I*(X2-X1)/M,Y1+I*(Y2-Y1)/M)
90 NEXT I
100 GOTO 100
```

Esta rutina es capaz de trazar una línea recta entre los puntos X1, Y1 y X2, Y2. El primer requisito que se debe cumplir es que las coordenadas del primer punto sean menores que las del segundo (X1<X2 y Y1<Y2). De no ser así, el bucle necesitaría incluir la cláusula STEP-1 en la línea 70.

El trazado se realiza por medio de los puntos creados en la línea 80. El primero de ellos corresponde a X1, Y1, ya que I vale cero. Los siguientes puntos se sitúan a incrementos fijos de cada una de las coordenadas. Dicho incremento viene dado por la expresión (X2-X1)/M o (Y2-Y1)/M. El dato M se calcula en la línea 60 y viene a coincidir con la diferencia máxima de coordenadas entre los puntos inicial y final. Así, si la mayor diferencia se encuentra en las coordenadas X, M tomará el valor de esa diferencia. En tal caso, el cociente (X2-X1)/M valdrá uno. Ello significa que el incremento en X será de un pixel por cada ejecución del bucle. Al repetirse éste M veces, se recorre la distancia precisa.



El cometido del comando PRESET es el de borrar (apagar) puntos de la pantalla.



Para el trazado rápido de rectas se hace uso del comando LINE. En su argumento se indican los puntos extremos del segmento a dibujar.

Por lo que respecta a la otra coordenada, la cosa se complica. El incremento es ahora un número no entero de pixels; número que, realmente, corresponderá a una fracción de pixel, puesto que M es mayor que Y2-Y1. En todo caso, ello no plantea ningún problema, ya que el ordenador toma automáticamente el entero más cercano para esa coordenada. De esta forma, el incremento en Y irá aumentando hasta completar una unidad. En ese preciso momento se iluminará el siguiente pixel, el otro, y así hasta terminar. Se puede comprobar que al repetir el bucle M veces, el punto final será, en cualquier caso: X2,Y2.

BORRADO DE PUNTOS

Se ha visto la forma de iluminar puntos con PSET (Point SET). Pues bien, también existe la posibilidad de borrarlos. El comando encargado de esta función es PRESET (Point RESET, borrar punto). Su formato es muy parecido al de PSET, como puede apreciarse a continuación.

PRESET

Borra el punto de la pantalla que corresponde a las coordenadas indicadas en su argumento.

Formato: PRESET [STEP] (<X>,<Y>) [,<color>]

Ejemplos: 20 PRESET (125,125)
70 PRESET STEP (A,A/2)

(Número de línea) PRESET (<X>,<Y>) [,<color>]

De utilizar la opción de color, PRESET se equipara a PSET. En realidad, utilizando el color no es necesaria la presencia de PRESET. Con el mismo comando PSET se puede borrar un punto sin más que ponerlo del color del fondo de la pantalla. Este nuevo comando, pues, no introduce ninguna acción nueva. Como ejemplo se muestra, a continuación, una rutina que borra una zona rectangular de la pantalla:

```
10 SCREEN 2
20 FOR X=0 TO 100
30 FOR Y=0 TO 100
40 PRESENT (X,Y)
50 NEXT Y
60 NEXT X
■
```

En el ejemplo se han tomado como límites del rectángulo de trabajo los puntos de coordenada X entre 0 y 100. Los mismos límites han sido tomados para la coordenada Y. Variando los datos de los bucles FOR se puede borrar cualquier otra zona de la pantalla.

MAS LINEAS...

Más arriba se ha indicado un método para trazar líneas haciendo uso del comando PSET. Realmente, este método resulta superfluo en la mayoría de los ordenadores. Por regla general, se dispone de un comando BASIC que realiza ese mismo cometido. En el caso del BASIC de Microsoft, dicho comando es LINE (en otros intérpretes se identifica con DRAW). Por medio de LINE se pueden trazar líneas con suma facilidad:

(Número de línea) LINE (<punto inicial>)–(<punto final>) [,[<color>], [B][F]]

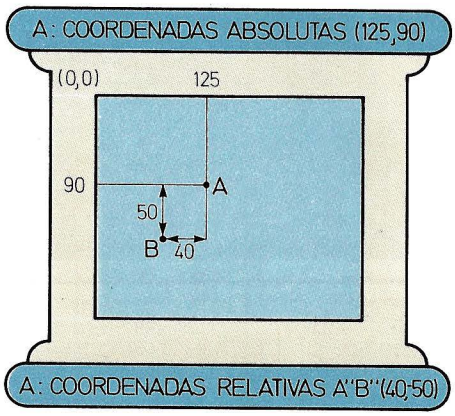
En las zonas de su formato identificadas

LINE

Traza una línea recta o un rectángulo entre los puntos especificados.

Formato: LINE [(<X1>,<Y1>)]–[STEP](<X2>,<Y2>)[,<color>[,B[F]]]

Ejemplos: 20 LINE (0,0)–(50,50)
70 LINE – STEP(50,50),,BF



La identificación de un punto puede realizarse especificando sus coordenadas absolutas (referidas al origen de coordenadas) o sus coordenadas relativas a otro punto de la pantalla.

como <punto inicial> y <punto final> se especificarán las coordenadas absolutas de ambos puntos. Dichas coordenadas se introducen separadas por una coma y en el orden habitual: primero X y luego Y. Así pues, para trazar una línea desde el punto 0,0 al 100,150 bastará con teclear lo siguiente:

Entre los restantes parámetros se encuentra el de color, el cual define el color que ha de tomar la línea a trazar. El color, como se verá más adelante, se especifica mediante un número o código.

LINE es un comando que aporta una mayor potencia a la programación de gráficos. Ahora resulta muy fácil realizar dibujos geométricos. Por ejemplo, un cuadrado de vértices en los puntos (0,0), (0, 50), (50,0) y (50,50); figura que se consigue con el siguiente programa.

```
10 SCREEN 2
20 LINE (0,0)–(0,50)
30 LINE (0,50)–(50,50)
40 LINE (50,50)–(50,0)
50 LINE (50,0)–(0,0)
100 GOTO 100
■
```

Otro ejemplo de la puesta en práctica de LINE consiste en el trazado de rectas aleatorias. Al igual que los comandos PSET y PRESET, LINE admite variables e incluso expresiones en su argumento. Con esta facilidad se puede hacer algo parecido al "cielo estrellado" de párrafos anteriores. El programa que se muestra a continuación traza líneas desde un punto central a otros situados aleatoriamente. Se supondrá el tamaño de pantalla utilizado hasta ahora (192×256). Si usted dispone de una pantalla de otro tamaño, puede adecuar el programa a su aparato cambiando sencillamente esos valores.

```
10 SCREEN 2
20 CLS
```

DRAW

Traza una línea desde el anterior punto trazado al que se indica en su argumento.

Formato: DRAW <increm. X>,<increm. Y>

Ejemplos: 20 DRAW 50,50
70 DRAW LX,LY

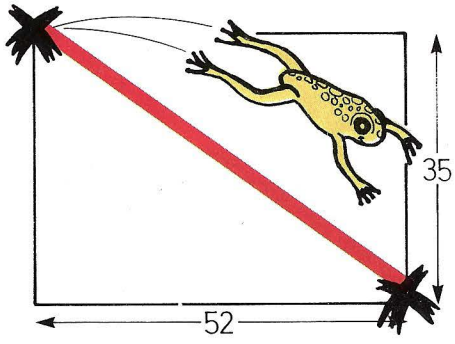
```
30 LINE (125,100)-(RND*255,RND*191)
40 GOTO 30
```

COORDENADAS ABSOLUTAS Y RELATIVAS

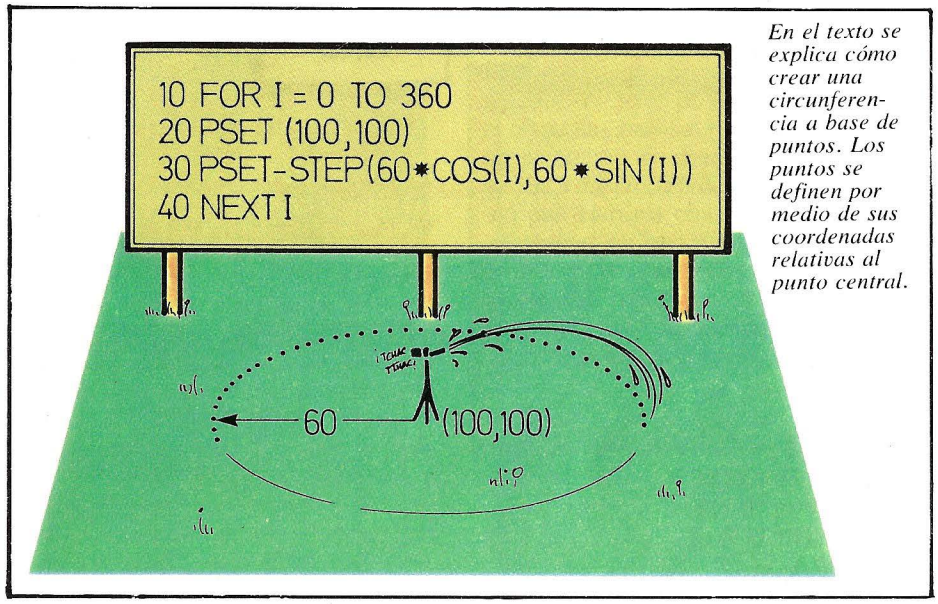
Se ha mencionado que en otros dialectos BASIC el trazado de rectas viene encomendado al comando DRAW. En la mayor parte de los casos éste utiliza un formato similar al siguiente.

(Número de línea) DRAW <incremento de X>, <incremento de Y>

En dicho formato llama la atención la presencia de tan sólo dos coordenadas. Con dos datos, tan sólo se puede especificar un punto. El otro punto que define la recta vendrá determinado por el comando que se haya ejecutado anteriormente. La filosofía es la siguiente: el último punto trazado servirá de punto de partida para el siguiente comando. En efecto, el punto siguiente se da especificando sus coordenadas relativas al punto actual. Ello significa que para determinar el punto final no es necesario conocer las coordenadas absolutas del mismo (las referidas al origen). Los datos que se han de suministrar dependen, pues, de la distancia al último punto. Como ejemplo, se muestra la realización del mismo cuadrado del apartado anterior. En este caso se supone que el último punto trazado corresponde al de coordenadas 0,0.



Para dibujar una línea con el comando DRAW, hay que indicar las coordenadas relativas al último punto trazado.



En el texto se explica cómo crear una circunferencia a base de puntos. Los puntos se definen por medio de sus coordenadas relativas al punto central.

```
10 DRAW 50,0
20 DRAW 0,50
30 DRAW -50,0
40 DRAW 0,-50
■
```

Partiendo del punto 0,0 se traza una línea horizontal hasta el punto 50,0. Para ello se ha de incrementar la coordenada X en cincuenta unidades, dejando intacta la coordenada Y. Esta es, precisamente, la misión de la línea 10. Para trazar el lado que va hasta el punto 50,50 ha de incrementarse la coordenada Y (que estaba en cero) y mantener la X (que se ha actualizado a 50). Los restantes lados se trazan aplicando el mismo método. En ellos se aprecia la posibilidad de dar incrementos negativos, lo cual permite acceder a puntos cuyas coordenadas sean inferiores a las del último punto referenciado.

A la vista del funcionamiento de las referencias relativas, cabe deducir que dicho método sólo permite trazar figuras conexas. Nada más lejos de la realidad. Se puede modificar la posición del último punto referenciado sin por ello trazar una recta; tal es el cometido del comando PSET (o PLOT en otros dialectos). Incluso, es posible indicar la situación del siguiente punto mediante sus coordenadas relativas a la posición actual.

Todo lo comentado es factible en el BASIC de Microsoft. La especificación de que el punto se expresa en sus coordenadas relativas se hace utilizando la cláusula STEP. Véase un ejemplo:

```
10 LINE-STEP (50,0)
20 LINE-STEP (0,50)
30 LINE-STEP (-50,0)
40 LINE-STEP (0,-50)
```

La presencia de la palabra STEP delante de los paréntesis indica que las coordenadas que se adjuntan están dadas en forma relativa. En el último ejemplo propuesto se vuelve a trazar el mismo cuadrado confeccionado con las rutinas anteriores.

La partícula STEP se puede emplear también con el comando PSET; en cuyo caso se tomará como origen el último punto trazado, siendo las coordenadas que se indiquen relativas a dicho punto.

Como ejemplo se trazará una circunferencia; para lo cual es preciso conocer la expresión de los puntos de la circunferencia en función de las coordenadas de su centro. Las fórmulas son las siguientes:

TABLA DE CONVERSION

ORDENADOR	PUNTOS (coordenadas absolutas)	PUNTOS (coordenadas relativas)	BORRADO DE PUNTOS	RECTAS (coordenadas absolutas)	RECTAN-GULOS (contorno)	RECTAN-GULOS (sólidos)	RECTAS (coordenadas relativas)
	PSET (<X>,<Y>)	PSET STEP (<X>,<Y>)	PRESET (<X>,<Y>)	LINE (<P_1>)–(<P_2>)	LINE (<P_1>)–(<P_2>),,B	LINE (<P_1>)–(<P_2>),,BF	DRAW (<X>,<Y>)
APPLE II (APPLESOFT)	HPLOT <X>,<Y>	—	—	HPLOT <P_1> TO<P_2>	HPLOT <P_1> TO<P_2>TO... (1)	—	—
APRICOT (M-BASIC)	—	—	—	—	—	—	—
ATARI	PLOT <X>,<Y>	—	—	DRAWTO <P_2> (2)	—	—	—
CBM 64	—	—	—	—	—	—	—
DRAGON	PSET (<X>, <Y>,<C>)	—	PRESET (<X>,<Y>)	LINE (<P_1>)–(<P_2>),<a>	LINE (<P_1>)–(<P_2>),<a>,B	LINE (<P_1>)–(<P_2>),<a>,BF	—
EQUIPOS MSX	PSET (<X>,<Y>)	PSET STEP (<X>,<Y>)	PRESET (<X>,<Y>)	LINE (<P_1>)–(<P_2>)	LINE (<P_1>)–(<P_2>),,B	LINE (<P_1>)–(<P_2>),,BF	—
HP-150	—	—	—	—	—	—	—
IBM PC	PSET (<X>,<Y>)	PSET STEP (<X>,<Y>)	PRESET (<X>,<Y>)	LINE (<P_1>)–(<P_2>)	LINE (<P_1>)–(<P_2>),,B	LINE (<P_1>)–(<P_2>),,BF	—
MPF	HPLOT <X>,<Y>	—	—	HPLOT <P_1> TO <P_2>	HPLOT <P_1> TO <P_2> TO... (1)	—	—
NCR DM-V (MS-BASIC)	—	—	—	—	—	—	—
NEW BRAIN	—	—	—	—	—	—	—
ORIC	—	—	—	—	—	—	DRAW <X>,<Y>,<C>
SHARP MZ-700 (MZ-BASIC)	SET <X>,<Y>	—	RESET <X>,<Y>	—	—	—	—
SINCLAIR QL	POINT <X>,<Y>	POINT_R <X>,<Y>	—	LINE <P_1> TO <P_2>	LINE <P_1> TO <P_2> TO... (1)	—	LINE_R TO <X>,<Y>
SPECTRAVIDEO	PSET (<X>,<Y>)	PSET STEP (<X>,<Y>)	PRESET (<X>,<Y>)	LINE (<P_1>)–(<P_2>)	LINE (<P_1>)–(<P_2>),,B	LINE (<P_1>)–(<P_2>),,BF	—
ZX-SPECTRUM	PLOT <X>,<Y>	—	—	—	—	—	DRAW <X>,<Y>

<X>: Coordenada horizontal del punto a trazar. <Y>: Coordenada vertical del punto a trazar. <C>: Código del color que ha de tomar el punto. <a>: Adoptará los valores PSET (para trazar) o PRESET (para borrar). <P_1> y <P_2>: Puntos inicial y final de la recta, dados en forma <X>, <Y>

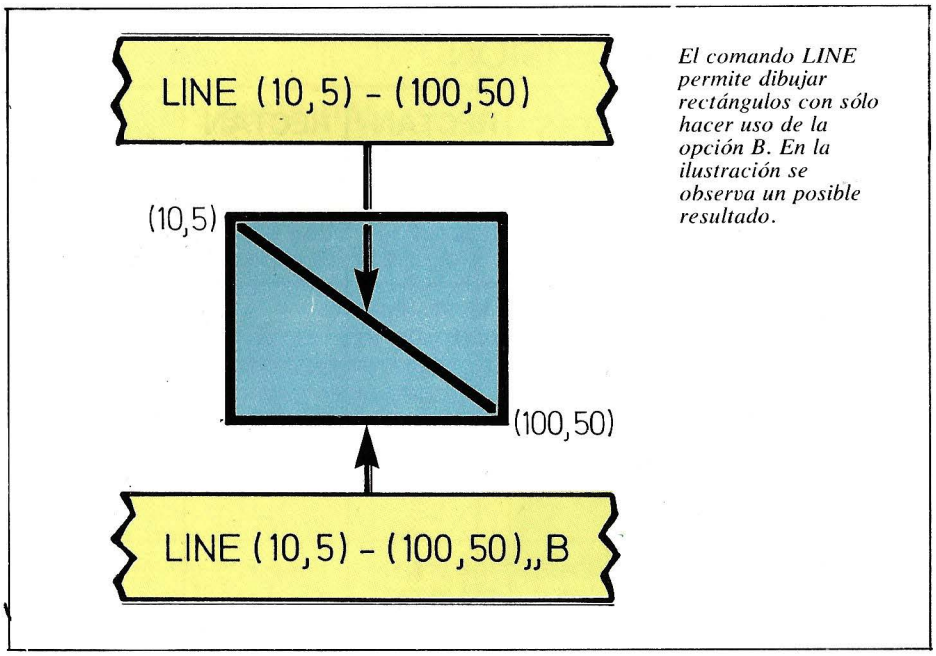
(1): El trazado de rectángulos se realiza por la concatenación de rectas en un mismo comando.

(2): La línea se traza desde el último punto referenciado hasta <P_2>. Este último se da en coordenadas absolutas.

$X=X_0+R$ *cos (a)
$Y=Y_0+R$ *sen (a)

En donde X_0 e Y_0 son las coordenadas del centro, R es la longitud del radio y "a" es el ángulo que forma el radio de cada punto con la horizontal. Como se trata de dibujar la circunferencia completa, se hará variar el ángulo de 0 a 360 grados (o de 0 a 2*PI radianes). He aquí el programa.

```
10 FOR I=0 TO 360
20 PSET (100,100)
30 PSET–STEP (60*COS(I),60*SIN(I))
40 NEXT I
```



El comando LINE permite dibujar rectángulos con sólo hacer uso de la opción B. En la ilustración se observa un posible resultado.

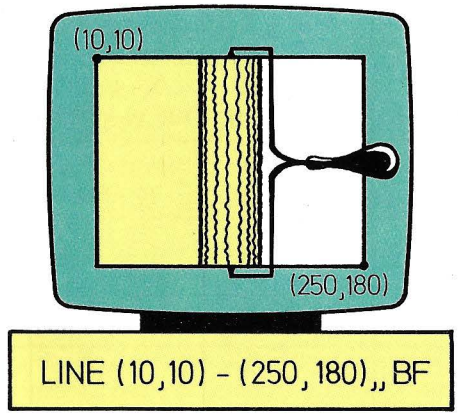
Si el aparato procesa los ángulos medidos en radianes, la línea 10 habrá de sustituirse por la siguiente:

```
10 FOR I=0 TO 6.28 STEP 0.01
```

En el ejemplo se ha situado el centro de la circunferencia en el punto 100,100. El radio tomado en este caso es de 60 unidades (60 pixels). El comando de la línea 20 ilumina una y otra vez el centro de la circunferencia, lo cual permite al siguiente comando apoyarse en dicho punto. De esta forma, los puntos trazados con el comando de la línea 30 vienen referidos al punto central.

CUADRADOS Y RECTANGULOS

Anteriormente se ha utilizado el comando LINE para dibujar un cuadrado de dos formas diferentes. En realidad, LINE tiene más posibilidades: las relacionadas con los parámetros opcionales B y F. El primero de ellos permite dibujar rectángulos que tendrán siempre sus lados paralelos a los límites de la pantalla; esto es, verticales y horizontales, pero nunca oblicuos. Para trazar una de estas figuras basta con incluir una B en el argumento del comando LINE. El rectángulo vendrá definido por los dos puntos que se especifiquen; puntos que corresponderán a dos vértices opuestos de dicho rectángulo. Utilizando esta opción, el cuadrado de los ejemplos anteriores se obtiene con una sola línea de programa:



El empleo conjunto de las opciones B y F permite crear rectángulos sólidos; esto es, rectángulos con toda la superficie interna coloreada.

```
10 LINE (0,50)-(50,0),,B
```

Es más, incluso se puede indicar el segundo punto referido al primero en modo relativo. Esto se consigue con la comentada partícula STEP. A continuación se muestra cómo emplear esta posibilidad para la generación del mismo cuadrado:

```
10 LINE (0,0)-STEP (50,50),,B
```

En principio, esta posibilidad no parece de mucha aplicación. Sin embargo, la tiene, tal y como pone de manifiesto el siguiente programa:

```
10 INPUT "COORDENADA X";X
20 INPUT "COORDENADA Y";Y
30 SCREEN 2
40 LINE (X,Y)-STEP (50,50),,B
100 GOTO 100
```

Las cinco líneas de programa trazan un cuadrado de lado 50, en el que la situación de su vértice superior izquierdo puede ser elegida por el operador. Al estar el segundo punto referido al primero, bastará con determinar ese primer punto.
Sea cual fuere el punto de partida seleccionado, el cuadrado siempre tendrá 50 pixels de lado, lo cual permite crear la figura con independencia de la posición de pantalla en la que se sitúe.
Existe una posibilidad adicional especificada con la opción F. La anterior hacía referencia a rectángulos (B de Box, caja). Esta otra permite rellenarlos (F de Fill, rellenar). Para utilizar la opción F es necesario trazar un rectángulo, no una recta. Con el empleo de BF aparecerá en pantalla un rectángulo sólido. Es decir, no sólo se trazará su contorno, sino que se rellenará toda su superficie con el color indicado. Así pues, el mismo cuadrado, aunque esta vez relleno, se puede crear con la rutina que sigue:

```
10 INPUT "COORDENADA X";X
20 INPUT "COORDENADA Y";Y
30 SCREEN 2
40 LINE (X,Y)-STEP (50,50),,BF
100 GOTO 100
```

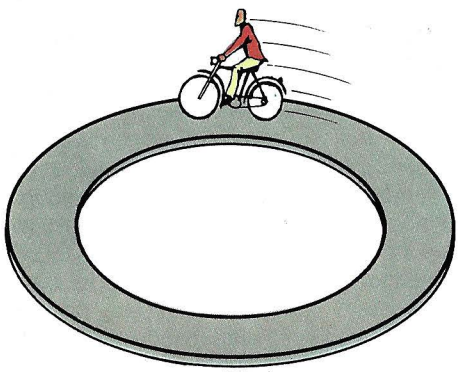
Forth (6)

Bucles anidados e indefinidos

En el anterior capítulo dedicado al lenguaje FORTH tuvo lugar una primera introducción al manejo de bucles o estructuras cíclicas. En este nuevo capítulo, volcado en el mismo tema, se analizan más posibilidades relativas a la programación de bucles en FORTH.

MAS SOBRE BUCLES CONTROLADOS

La técnica más sencilla para la creación de un bucle se concreta en partir de un valor de referencia prefijado, valor que debe incrementarse en una unidad cada vez que sea recorrido el bloque de instrucciones que dan cuerpo al bucle; la secuencia de ejecución cíclica terminará, saliendo del bucle, cuando el valor de referencia —contador de ciclos— alcance el valor final, también prefijado. En ciertas ocasiones, es necesario crear un bucle en el que, en cada pasada, el incremento del valor de referencia no sea unitario, sino de varias unidades; e incluso puede interesar que el valor del incremento sea positivo o negativo.



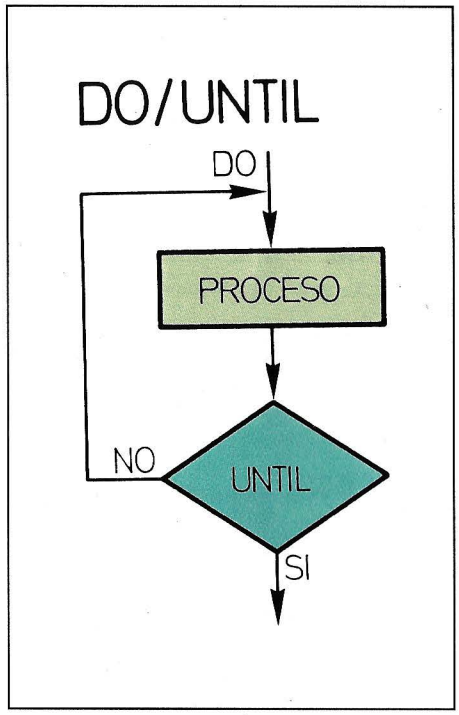
La ejecución de un bucle DO/LOOP puede compararse con una carrera de velocidad que termina una vez que se han completado un determinado número de vueltas al circuito. A su vez, los bucles de tipo DO/UNTIL son equiparables a las "24 horas de Le Mans": hay que correr durante 24 horas sin importar el número de vueltas que se den al circuito.

Para contemplar esta posibilidad, el lenguaje FORTH dispone de una palabra adecuada para sustituir a la palabra LOOP que encontrábamos anteriormente. Esta nueva palabra (+LOOP) incrementará el valor del índice en tantas unidades como indique el número que se encuentre en la cima de la pila.
Veamos un sencillo ejemplo:

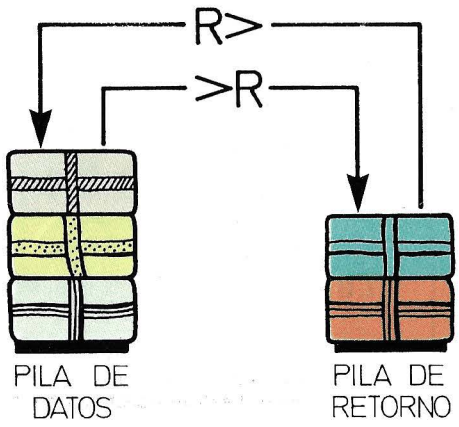
```
: CANGREJO <CR>
-1 10 <CR>
DO <CR>
-1 <CR>
+LOOP <CR>
; <CR>
```

La ejecución de la palabra que acabamos de crear, CANGREJO, no produce ningún resultado en la pantalla. Para observar algún efecto en la visualización vamos a introducir una nueva palabra FORTH. Se trata de la palabra I, cuya función consiste en copiar el valor del índice, que se encuentra en la pila de retorno, a la pila de datos, permitiendo de este modo realizar alguna operación con el referido índice.
En este caso, el índice utilizará para examinar la forma en la que evoluciona el recorrido del bucle. Al efecto, será preciso editar la palabra CANGREJO, previamente definida, y modificarla hasta que adopte la siguiente forma:

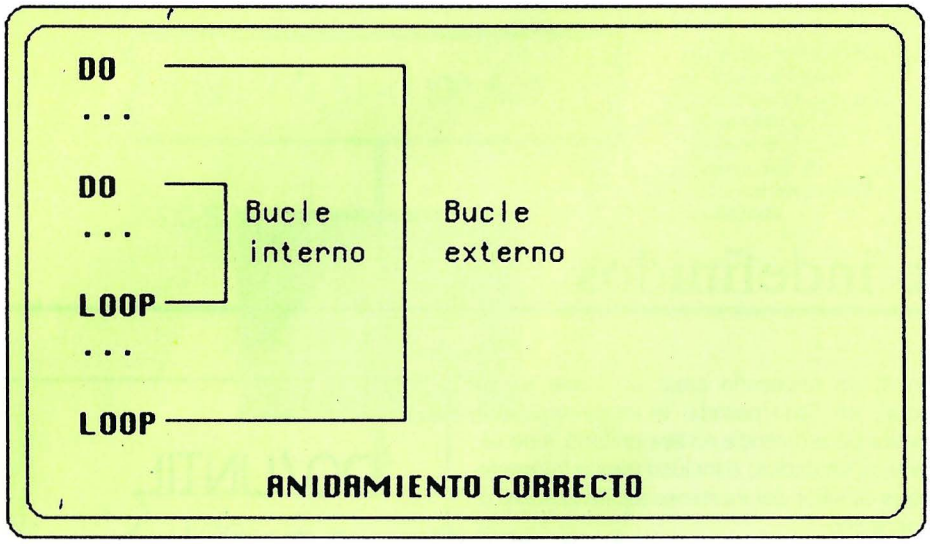
```
: CANGREJO
-1 10
DO
I . -1
+LOOP
;
```

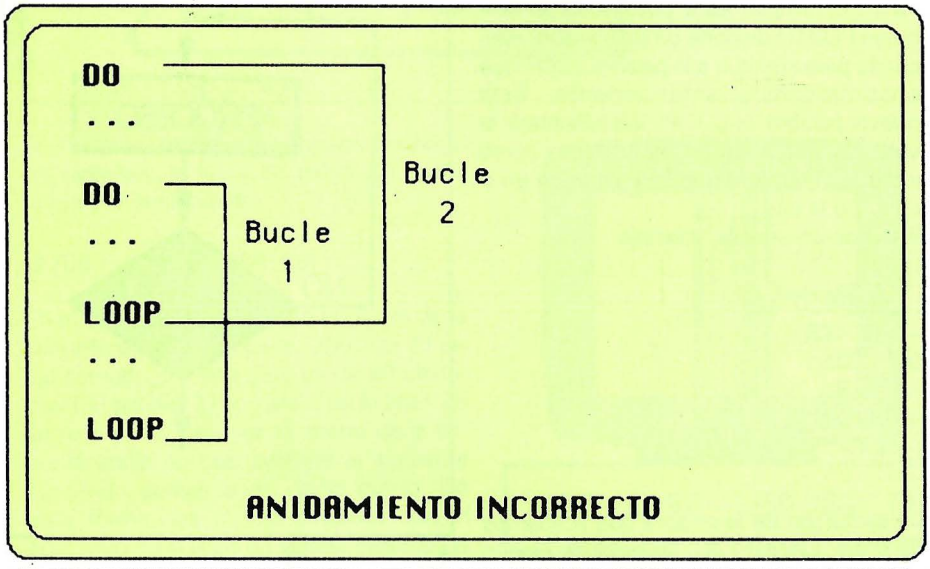


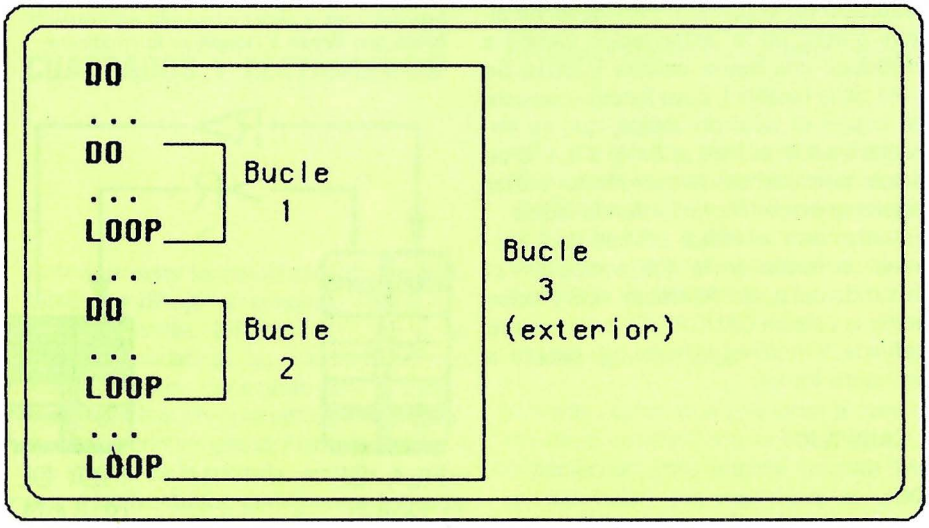
La estructura DO/UNTIL permite repetir un proceso tantas veces como sea necesario hasta que llegue a cumplirse la condición impuesta.



Las palabras R y R permiten transportar datos de la pila de datos a la de retorno, y viceversa.







Ejemplos de anidamiento correcto e incorrecto de bucles DO/LOOP. En la ilustración inferior se observa una estructura algo más compleja, en la que un bucle externo engloba a dos bucles independientes.

Ahora, al invocar la palabra CANGREJO, la respuesta en pantalla será la siguiente:

```
CANGREJO <CR>

CANGREJO 10 9 8 7 6 5 4 3 2 1 0 OK
```

En efecto, se observa que tal y como se ha previsto, el índice evoluciona en sucesivos saltos de −1 unidad por cada pasada que se da al bucle.

Además de la palabra I, cuya misión es pasar un valor de la pila de retorno a la de datos, existen más palabras FORTH que permiten operar pasos de una a otra pila. Por ejemplo: la palabra >R, la cual permite pasar valores de la pila de datos a la de retorno, y la palabra R> que los transfiere de la pila de retorno a la pila de datos.

BUCLES ANIDADOS

Al igual que la mayor parte de los lenguajes de alto nivel, también en el FORTH es posible construir bucles anidados. El anidamiento de bucles consiste, sencillamente, en introducir nuevos bucles dentro de otros más generales. La única condición que se impone al anidamiento de bucles es, precisamente, que tal anidamiento sea correcto. Es decir, si un bucle está incluido dentro de otro más amplio, es necesario que el bucle anidado comience y termine en el interior del bucle externo.

En el cuadro adjunto puede observarse algún ejemplo gráfico de anidamiento correcto e incorrecto.

En la figura adjunta, se observa que también es posible realizar anidamientos más complejos. El ilustrado corresponde al anidamiento de tres bucles en dos niveles: el bucle externo incluye a dos bucles elementales y encadenador en sucesión.

Veamos un ejemplo programado de anidamiento de bucles:

```
: BUCLE <CR>
5 1 <CR>
DO <CR>
I . 5 1 <CR>
DO <CR>
I . <CR>
LOOP <CR>
CR <CR>
LOOP <CR>
; <CR>
```

En el ejemplo aparece una nueva palabra FORTH: CR, la cual ordena un retorno de carro en la pantalla; esto es: al ejecutar la palabra CR cambia la posición del cursor, pasando del punto en el que la dejó la anterior operación realizada en la pantalla al comienzo de la siguiente línea.
Al invocar a la palabra BUCLE:

```
BUCLE <CR>
```

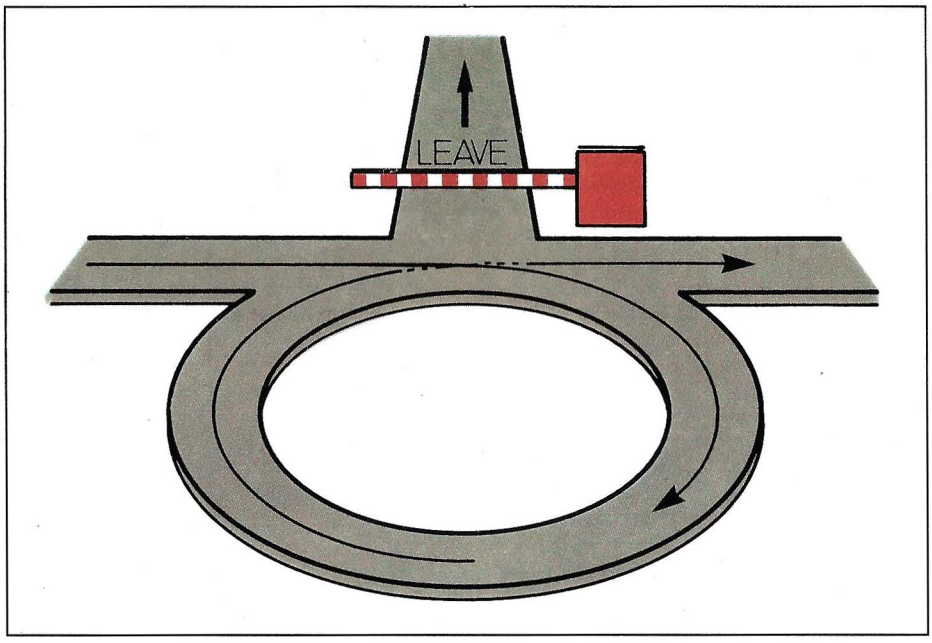
se observará en la pantalla conectada al ordenador el resultado que sigue:

```
BUCLE 1 1 2 3 4 5
2 1 2 3 4 5
3 1 2 3 4 5
4 1 2 3 4 5
5 1 2 3 4 5 OK
```

Veamos un nuevo ejemplo:

```
: FILAS <CR>
CR 1+ 1 <CR>
DO <CR>
I 0 <CR>
DO <CR>
." *" <CR>
LOOP <CR>
CR <CR>
LOOP <CR>
CR <CR>
; <CR>
```

Para invocar a la nueva palabra definida (FILAS), es necesario dejar en la pila un valor que indique el número de filas de asteriscos que se desea dibujar en la pantalla.



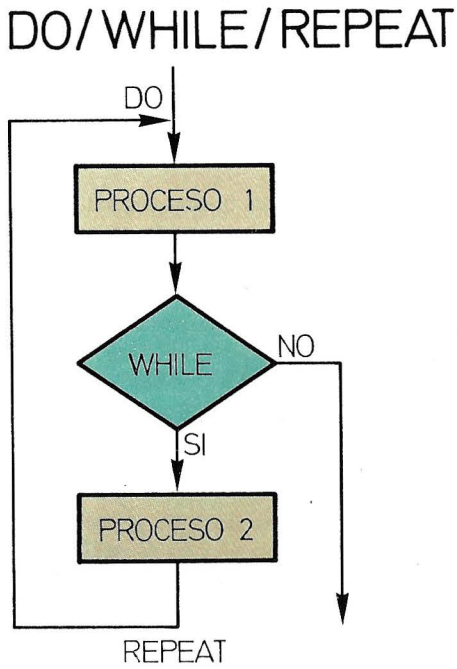
LEAVE es una palabra del vocabulario FORTH cuya llamada permite abandonar la ejecución de un bucle.

```
4 FILAS <CR>

4 FILAS
*
**
***
****
OK
```

Para abandonar un bucle durante su ejecución, es necesario ejecutar una instrucción LEAVE, la cual permite abandonar el ciclo en curso. Para ello, llevará el valor del índice al límite y originará la salida del bucle tras completar la pasada que se esté efectuando en ese preciso momento.

BUCLES INDEFINIDOS

Los bucles que se han utilizado hasta el momento eran bucles controlados. Ello significa que el bucle se ejecuta un número de veces previamente establecido, antes de que dé comienzo la ejecución del bucle (excepto en el caso de que se utilice la palabra LEAVE).



Las palabras DO/WHILE/REPEAT actúan sobre dos procesos que se ejecutarán cíclicamente hasta que deje de cumplirse una determinada condición. Tal como muestra el diagrama de flujo, el proceso 2 no se ejecutará al salir de la estructura.

Existen otros dos tipos de bucles cuyo funcionamiento difiere del descrito hasta ahora. Se trata de los bucles creados con el apoyo de las palabras BEGIN/UNTIL y BEGIN/WHILE/REPEAT. En esencia, estos bucles se ejecutan tantas veces como

TABLA DE ORDENES FORTH

PALABRA	DESCRIPCION	TIPO
DO/+LOOP	Creación de un bucle, con un incremento sucesivo del índice distinto a la unidad	Palabras de control
LEAVE	Permite abandonar un bucle antes de que éste sea completado	Palabras de control
DO/UNTIL	Ejecuta un proceso hasta que se cumple la condición impuesta	Palabras de control
DO/WHILE/REPEAT	Ejecuta un proceso hasta que deje de cumplirse la condición establecida	Palabras de control
EXIT	Permite abandonar un bucle DO/UNTIL o DO/WHILE sin necesidad de que se satisfaga la condición impuesta	Palabras de control
I	Toma el valor del índice de la pila de retorno y los pasa a la pila de datos	Transferencia de datos
CR	Genera un retorno de carro	Impresión en pantalla
R>	Transfiere valores de la pila de retorno a la de datos	Transferencia de datos
>R	Transfiere valores de la pila de datos a la de retorno	Transferencia de datos

sea necesario hasta que se vea satisfecha una condición. Veamos cada uno de ambos casos por separado.

- Bucles BEGIN/UNTIL

Se ejecutarán las instrucciones encerradas entre las palabras BEGIN y UNTIL las veces que sea necesario hasta lograr que se verifique la condición impuesta. La evaluación del referido condicionante ha de realizarse antes de llegar a la instrucción UNTIL, de tal forma que al llegar al nivel de UNTIL, se encuentre en la cima de la pila el valor «verdadero» o «falso». Veamos un ejemplo:

```
: +- <CR>
BEGIN <CR>
1 + DUP . DUP 8 <CR>
> <CR>
UNTIL <CR>
; <CR>
```

Al invocar la ejecución de la palabra definida se obtiene la siguiente respuesta en pantalla:

```
1 +- 2 3 4 5 6 7 8 9 OK
```

La instrucción BEGIN/WHILE/REPEAT presenta algunas diferencias esenciales respecto a la instrucción UNTIL. Para empezar, en este nuevo caso la ejecución se detendrá precisamente cuando la condición deje de cumplirse. Además, hay dos secciones dentro del bucle: una de ellas se ejecuta antes de examinar la condición (la zona comprendida entre BEGIN y WHILE), mientras que la otra sección se ejecutará tan sólo si la condición se cumple. En caso de que no se verifique la condición impuesta, la secuencia saltará directamente a la instrucción que sigue a la palabra REPEAT.
El siguiente ejemplo ilustra alguno de estos conceptos:

```
: -- <CR>
BEGIN <CR>
2+ DUP 16 < <CR>
WHILE <CR>
DUP . <CR>
REPEAT <CR>
; <CR>
```

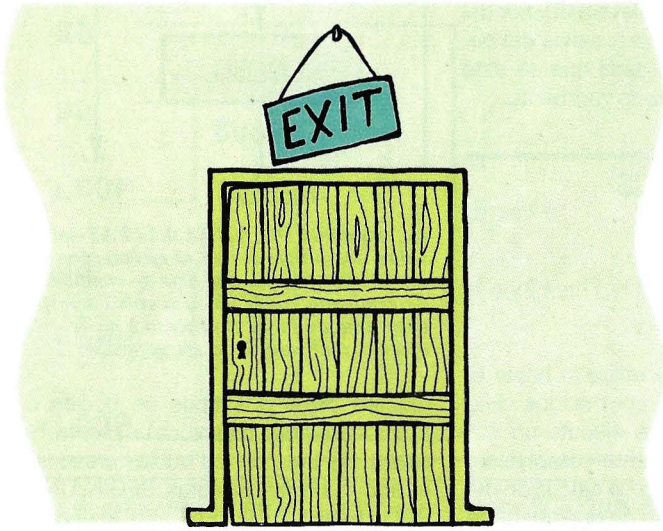
Veamos cuál será el resultado de ejecutar la nueva palabra:

```
1 -- <CR>
```

Y la respuesta que se obtiene en pantalla es la siguiente:

```
1 -- 3 5 7 9 11 13 15 OK
```

De estos bucles también es posible salir por la «puerta falsa» empleando la instrucción EXIT; al igual que ocurría con los bucles controlados en el caso de ejecutar la instrucción LEAVE.



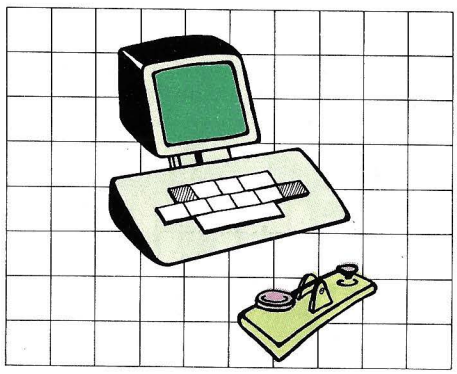
EXIT permite abandonar la ejecución de un bucle por "la puerta falsa".

Oasis (3)

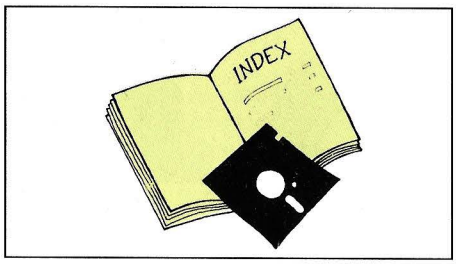
Gestión de los ficheros en disco

Un sistema operativo es, esencialmente, un surtido de herramientas para gestionar la información. Información que es almacenada y gestionada de una forma variable, dependiendo de las características de los datos, de su estado de proceso, así como la posible previsión de uso que se va a hacer de los mismos. Resulta obvio que no serán tratados de igual forma los datos residentes en un disco rígido, que los contenidos en un disquete o en una unidad de cinta magnética. Igual de importantes son los métodos empleados para la protección de la información, puesto que en un sistema multiusuario es evidente que existen posibilidades no factibles en un sistema monousuario.

En el sistema operativo OASIS se ha prestado una especial atención a estos aspectos. Especialmente, al hecho de facilitar una gestión de los ficheros lo más transparente posible, para que los usuarios con poca experiencia no se vean desbordados ante complicados comandos de operación con ficheros, tales como los empleados en sistemas operativos como el CP/M, en sus distintas versiones, o el MS/DOS. Por lo demás, el OASIS presenta toda una serie de posibilidades para el bloqueo de la información ante posibles accesos indeseables en un ambiente de operación multiusuario.



El tratamiento que deben recibir los datos difiere según sea el origen de los mismos. Este es un hecho que debe contemplar el sistema operativo y, en consecuencia, gestionar según corresponda los datos residentes en disquete, disco rígido o cinta magnética.



La función del directorio es la de actuar a modo de índice o agenda de información que facilite la tarea del sistema operativo cuando éste deba acceder al disco.

ALMACENAMIENTO Y CRITERIOS DE NOMENCLATURA

El sistema operativo OASIS está configurado de tal forma que admite la operación de hasta ocho unidades de disco, lo que supone una respetable capacidad de almacenamiento. El sistema gestiona cada una de las unidades de disco por separado, con lo cual se inhibe la posibilidad de que un fichero pueda comenzar en un disco y continuar en otro, obligando a que cada fichero esté contenido en un solo disco. Como consideración adicional, cabe indicar que un fichero sólo podrá tener como tamaño máximo el del disco en el que reside.

Todo disco que vaya a ser utilizado con el sistema operativo OASIS ha de ser inicializado previamente; de lo contrario no será posible su empleo como medio de almacenamiento. Esta operación la lleva a cabo un programa que forma parte del propio sistema operativo y que es invocado a modo de comando. Se trata del comando INITDISK, el cual hace uso en su interior de la opción FORMAT.

Desde luego, un disco puede ser inicializado cuantas veces se desee, aunque hay que tener en cuenta que este proceso elimina la información almacenada en el mismo, la cual no puede ser recuperada posteriormente.

La información que se graba en el disco durante el proceso de inicialización o formateado se divide en tres bloques:

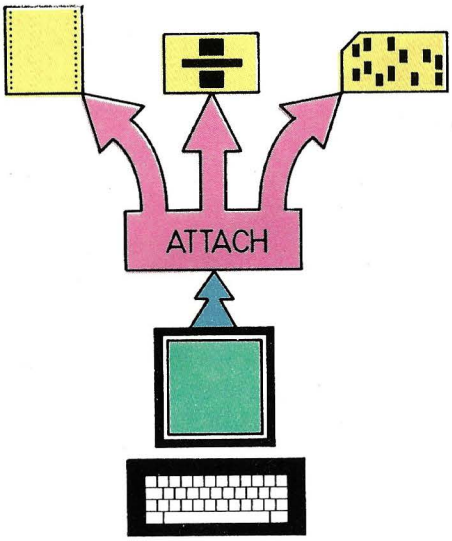
— Directorio
— Etiqueta del disco
— Mapa del disco

La función del directorio es la de actuar a modo de índice o agenda de información para el sistema operativo cuando este accede al disco. En efecto, el directorio contiene información sobre el contenido del disco; esto es: acerca de sus diferentes ficheros, así como los punteros que indican la situación de los ficheros. Resulta evidente que sin esta información el sistema operativo se encontraría impotente a la hora de extraer información de los periféricos de almacenamiento masivo.

El mapa del disco tiene por objeto llevar un control de las áreas del disco empleadas para el almacenamiento de información; estas zonas se dividen en una serie de bloques, que en el caso del OASIS son de 1K, es decir, 1.024 bytes de información.

CRITERIOS DE BUSQUEDA DE FICHEROS

Cuando el usuario de un ordenador equipado con el sistema operativo OASIS co-



El sistema operativo OASIS permite asignar a un dispositivo lógico diferentes dispositivos físicos. De este cometido se encarga el comando ATTACH.

municа su deseo de acceder a un fichero determinado, ha de especificar los siguientes datos:

— Nombre del fichero
— Tipo de fichero
— Disco en que está almacenado

Del nombre del fichero y de su tipo hablaremos posteriormente. Vamos ahora a ocuparnos del disco. Este puede ser identificado de dos formas. La primera de ellas especificando la etiqueta del disco, la cual, como se recordará, fue generada por medio del comando INITDISK. La segunda, a través de una etiqueta de directorio de un carácter, la cual se establece por medio del comando ATTACH. La función de este comando es crear una asociación entre un dispositivo físico y un dispositivo lógico, permitiendo así que el operador se adapte a los periféricos de que dispone, o cambie de forma dinámica el periférico de entrada/salida sin necesidad de efectuar ninguna modificación en los programas que está manejando. Como se ha mencionado, el sistema operativo OASIS puede controlar hasta ocho unidades de disco. Estas unidades reciben una denominación específica que es:

S, A,B,C,D,E,F,G

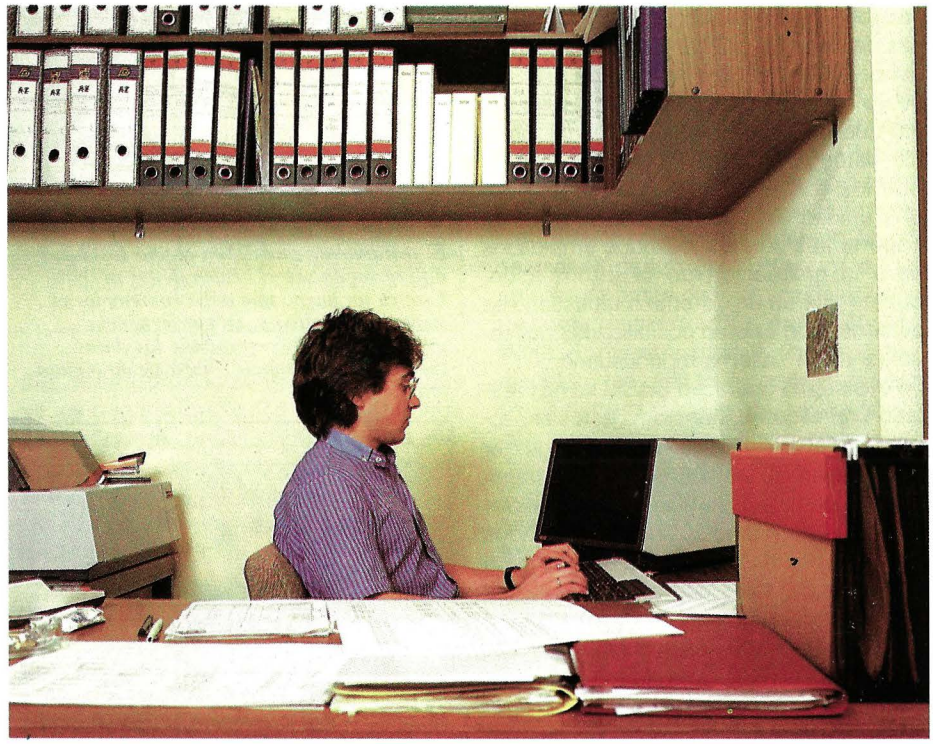
donde S es el disco del sistema operativo. La letra que se designa a la unidad de disco en la que se encuentra almacenado el fichero al que se desea acceder, es la etiqueta de directorio a especificar en el caso de no haber indicado la etiqueta del disco.

Puede darse el caso de que, por cualquier motivo, no sea posible especificar el disco en el cual se encuentra almacenado un determinado fichero. En tal situación, el sistema operativo OASIS inicia un proceso de rastreo a través de las diversas unidades de almacenamiento masivo. Este proceso es diferente según lo que el sistema esté buscando. Si se trata de un programa de usuario, los discos son rastreados, uno a uno, y por el orden alfabético de sus referencias, es decir:

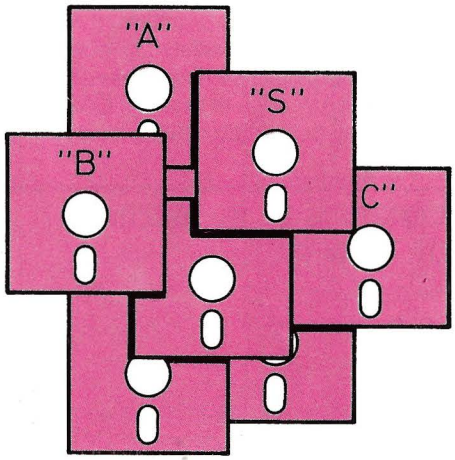
A →, B →, C →, D →, E →, F →, G →, S

Sin embargo, si lo que se especifica es un comando, que por su definición en el sistema operativo deba estar almacenado en disco, éste será buscado inicialmente en el disco del sistema (S), y a continuación en los restantes discos, siguiendo el orden alfabético.

Igual procedimiento se emplearía con un programa EXEC. Como ya sabemos, este es el lenguaje de comandos propios del OASIS, a través del cual pueden encadenarse una serie de operaciones definidas por medio de distintos comandos del sistema.



El OASIS presta un especial cuidado a la protección y al bloqueo de los datos frente a posibles accesos no deseables. Hay que tener en cuenta que su ámbito de explotación suele coincidir con un entorno multiusuario.



Para facilitar la gestión de las memorias de masa asociadas al sistema, el OASIS permite asignar a las unidades de disco claves distintas para su identificación.

CRITERIOS DE DENOMINACION DE FICHEROS

Cualquier fichero que se cree en el transcurso de las operaciones con el ordenador tiene únicamente dos caminos de generación: el propio sistema operativo del equipo, o el programa de usuario que esté trabajando en la actualidad sobre el ordenador. En cualquier caso, el fichero ha de ser denominado de forma tal que se excluya la confusión con otros ficheros. De ahí que sea necesario seguir unos criterios de nomenclatura, similares en la mayoría de los sistemas operativos, aunque con peculiaridades propias en cada uno.
En el caso del sistema operativo OASIS, la denominación de un fichero consta de tres elementos:
NOMBRE del fichero. *Tipo* del fichero: *Disco* en el que se almacena.
El significado de cada uno de estos elementos es:

— Nombre del fichero: incluye de uno a ocho caracteres de longitud, siendo el primero de estos caracteres obligatoriamente una letra, mientras que los restantes pueden contener otras letras, números o el signo «$», no admitiéndose otro tipo de caracteres en esta nomenclatura.

— Tipo de fichero: también debe incluir de uno a ocho caracteres y se utiliza como un caracterizador del nombre del fichero. Sin embargo, hay que tener en cuenta que algunos de estos tipos de fichero tienen un significado particular para el sistema operativo OASIS y, por lo tanto, han de ser empleados en consecuencia. Por lo que se refiere al tipo de caracteres, se siguen en este caso las mismas reglas que las aplicadas al nombre del fichero.

— Disco en el que está almacenado. Esto

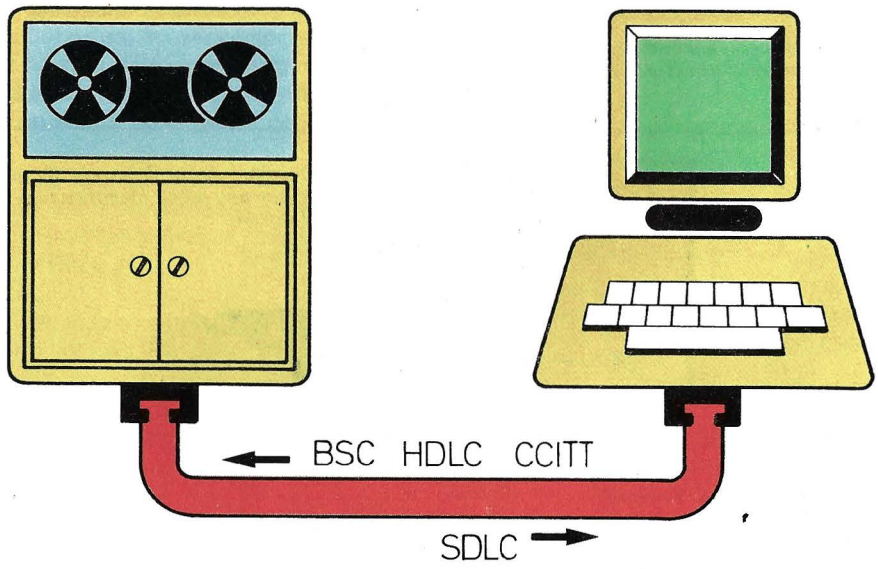
El acceso a los grandes ordenadores

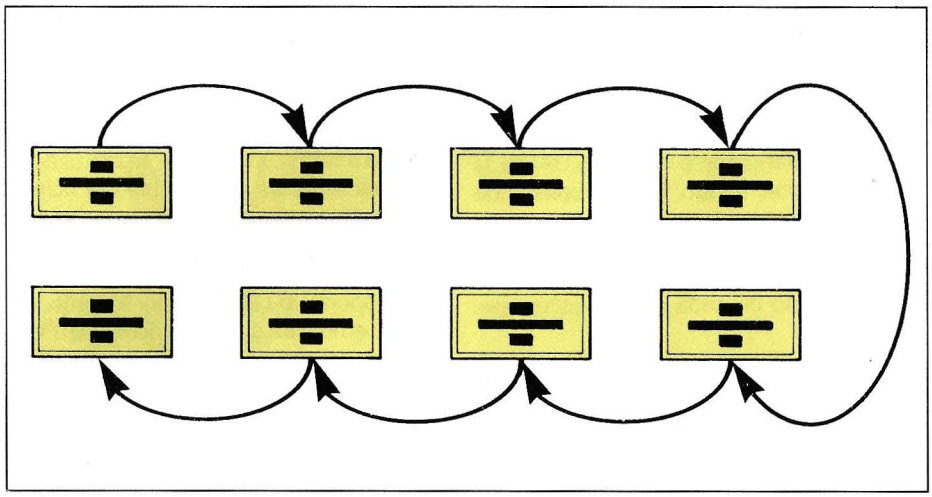
Cada vez es más frecuente el uso de los microordenadores como nexo de unión con grandes ordenadores o «main frames», de modo que los usuarios modestos tengan acceso a la gran velocidad de proceso de que disfrutan los grandes ordenadores. En tal situación, los microordenadores se destinan principalmente a la preparación de datos y a la recogida de resultados, ya que el tratamiento es bastante más fácil en estos últimos debido a la gran cantidad de software existente en el mercado orientado hacia este objetivo (hojas electrónicas, paquetes de gestión, gráficos, etc.).
Para la comunicación con el gran ordenador es necesario disponer de una conexión física (un dispositivo de interface serie de tipo RS232-C ó RS422 y cable coaxial es suficiente en la mayoría de los casos), además del software adecuado para emular los diversos protocolos utilizados en la transmisión de información.
La forma en que se transfiere la información de un sistema a otro es puramente secuencial, un bit a continuación de otro, y tiene dos variantes básicas; síncrona y asíncrona. El nombre de transmisión síncrona procede de que la información es enviada en paquetes discretos, con intervalos de tiempo constantes entre paquetes. Una velocidad de transmisión típica es, por ejemplo, la de 9.600 baudios.
Por contra, la transmisión asíncrona se realiza en intervalos de tiempo no constantes. Cada paquete de bits es identificado por ciertos bits especiales, de control, que señalan el comienzo y el fin de la transmisión. Este segundo método presenta la ventaja de que no requiere software ni equipos tan sofisticados como los necesarios para la transmisión síncrona; aunque tiene el inconveniente de que la comunicación es más lenta debido a que un extremo no sabe cuándo va a recibir información del otro.

La información puede ir de un extremo a otro de la línea en un solo sentido (half-duplex) o en ambos (full-duplex).

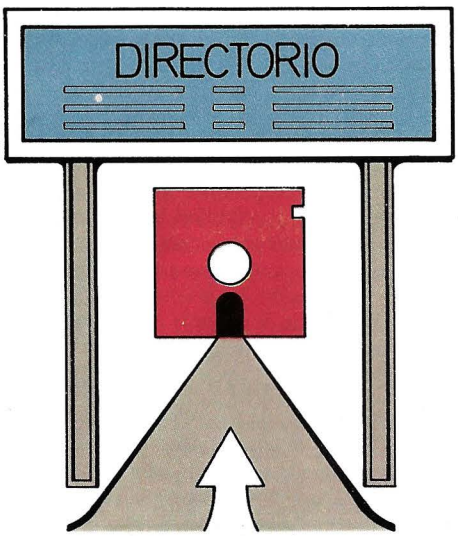
Existen métodos de control para la detección de errores, como por ejemplo, el derivado de utilizar 7 bits para la transmisión y un octavo para reflejar su paridad (si la paridad se define como par y el número de bits cuyo valor es uno es impar, este último bit será 1, y 0 en caso contrario).

La norma más extendida para la transmisión de datos es el SNA (System Network Architecture) de IBM, con sus protocolos BSC (Bysinchronous System Communication), SDLC (Sinchronous Data Link Control) y HDLC (High-level Data Link Control), así como las normas CCITT.
La puesta a punto de los diversos sistemas y equipos para llevar a cabo la transmisión de datos a través de una línea (línea de transmisión, controladores, emuladores de protocolo, etc.), puede conducir a gastos de hasta 6 y 7 veces el valor del microordenador con el que se pretendía realizar la conexión. De ahí que también se esté impulsando la vía de la conversión de terminales clásicos de gran ordenador en terminales inteligentes, con las mismas propiedades y periféricos que los microordenadores, evitando así la parte más cara costituida por las comunicaciones.





En el caso de que no se especifique la unidad en la que reside el fichero con el que hay que operar, el sistema operativo inicia una búsqueda secuencial a través de los discos alojados en las distintas unidades.



El acceso al contenido de cualquier disco se realiza siempre a través del directorio del mismo.

se indica por medio de uno cualquiera de los dos métodos expuestos al hablar de la localización de un fichero; bien por medio de la etiqueta del directorio, o bien por medio de la etiqueta del disco.
En la mayoría de los casos no será preciso especificar este último elemento de identificación, ya que, como se ha visto, el sistema operativo realizará una búsqueda secuencial por los diferentes discos conectados al ordenador.

FORMATOS DE LOS FICHEROS

El sistema operativo OASIS emplea seis formatos distintos de ficheros; estos son:

El nombre de un fichero ha de aportar una plena identificación del mismo.

- Ficheros secuenciales ASCII

Estos son ficheros conformes al American National Standard Code for Information Interchange; en ellos, cada carácter está representado por un código de 8 bits. Los ficheros de este tipo son los creados por medio del Editor, programas del MACRO-ensamblador, o ficheros de datos creados por el usuario. Este tipo de ficheros son los únicos que soportan registros de longitud variable; aunque, por su carácter secuencial, no pueden ser actualizados registro a registro, sino que han de añadirse los nuevos registros al final del fichero.

- Ficheros de acceso directo

Contienen datos binarios sobre registros de longitud fija y con un número determinado de registros. El acceso a cada uno de los registros se efectúa especificando un número coincidente con el identificador de dicho registro.

- Ficheros indexados

El sistema operativo OASIS soporta un tipo de fichero de una sola clave, siendo, al igual que los ficheros de acceso directo, de longitud fija y con un número de registros predeterminado. La clave es un campo del registro que actúa como identificador del mismo y que permite acceder a él con independencia de los demás. El OASIS mantiene siempre este tipo de ficheros ordenados por clave.

- Ficheros en formato de clave

Son idénticos en su formato a los ficheros indexados, sólo que no se mantienen ordenados por clave.

- Ficheros en formato absoluto

Son ficheros generados por el programa LINK-editor y están restringidos a los programas en código máquina. Contienen a los programas en un formato idéntico al que tendrán en la memoria del ordenador una vez cargados.

- Ficheros en formato reubicable

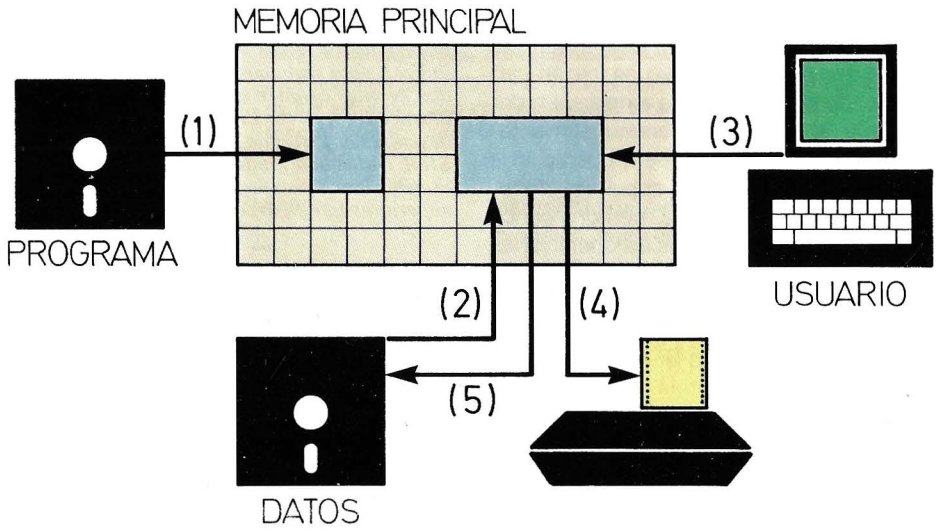
Son ficheros también reservados a los programas en código máquina, pero en este caso son originados con una dirección de memoria cero, siendo reubicado el programa en memoria a medida que es cargado.

Easy Writer (1)

La sencillez de un procesador de textos

El programa EASY WRITER es uno de los paquetes estandarizados para el «tratamiento de textos» de mayor popularidad y difusión. Como su propio nombre indica, el objetivo de EASY WRITER es permitir al usuario «una escritura fácil». Dentro de los paquetes estandarizados para la explotación de ordenadores personales, los programas destinados al tratamiento de textos ocupan un papel muy destacado; su facilidad de uso y gran utilidad les hacen candidatos a dar servicio a usuarios de muy distinto tipo.

Dentro de este grupo de programas, uno de los más consolidados es el que nos ocupa: el paquete EASY WRITER. Dos de las cualidades que le han servido para situarse en las posiciones de liderazgo en el terreno de los procesadores de texto, son su sencillez de uso y su gran capacidad para el proceso de palabras.



Las operaciones habituales en una sesión con el paquete EASY WRITER son, por este orden, las siguientes: (1) cargar el programa en memoria, (2) cargar datos en memoria, (3) edición de texto por parte del usuario, (4) producción del documento escrito, y (5) almacenamiento de los datos para futuras sesiones.

CARACTERISTICAS Y REQUISITOS PARA UTILIZAR EASY WRITER

El ámbito de funcionamiento de EASY WRITER se puede resumir en un ordenador personal, con una memoria principal de al menos 64 Kbytes. En cuanto a la memoria auxiliar o de masa, resulta imprescindible disponer al menos de una unidad para disco flexible; aunque, por supuesto, se pueden conseguir capacidades más amplias utilizando un disco rígido. El carácter de la aplicación hace que también resulte importante disponer de una impresora de cierta calidad.

Las características lógicas más relevantes del programa EASY WRITER son las siguientes:

- Editor de pantalla completa, que permite al usuario desplazarse a través de la misma por medio del cursor, sin necesidad de ejecutar ningún tipo de comando. Ello permitirá escribir con completa libertad sin preocuparse del fin de línea.

- Posibilidad de realizar modificaciones en un texto previamente escrito, permitiendo insertar o eliminar caracteres, palabras y líneas.

- Búsqueda y modificación de palabras dentro del texto; de tal forma que sin más que indicar al programa la palabra a sustituir y la palabra que debe reemplazarse, se operará automáticamente la sustitución en todas sus ocurrencias dentro del documento.

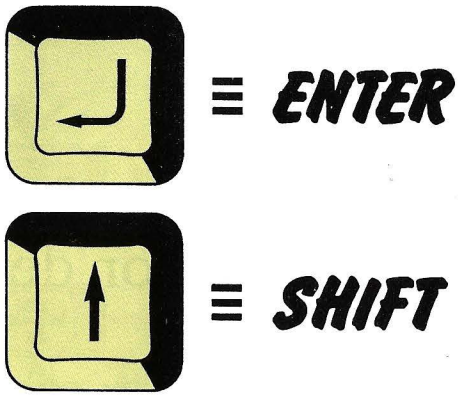
- Gestión del tipo de documento a producir. Dentro de este apartado cabe citar la posibilidad de obtener un paginado automático, inclusión de títulos y cabeceras, justificación de márgenes, centrado de líneas, etc.

- Posibilidad de definir bloques de texto, lo que facilita las operaciones de copia y desplazamiento de varias líneas simultáneamente.

- Sistema de ayuda (help) como soporte para el usuario final, encargado de resolver, de forma interactiva, las dudas que se le puedan plantear.

- Utilidades para la explotación del programa, destinadas a dar información sobre las características de la sesión de trabajo en curso. Estas permiten, entre otras cosas, contar el número de palabras contenidas por un documento.

CARACTERISTICAS DEL PROGRAMA EASY WRITER



Dentro de las teclas especiales que se utilizan para trabajar con EASY WRITER, dos de ellas asumen un papel muy destacado: la tecla ENTER, que puede desempeñar distintas funciones, y la tecla SHIFT utilizada fundamentalmente para escribir letras mayúsculas o los signos superiores que figuran en las diversas teclas.

Al cargar el programa en la memoria principal del ordenador, EASY WRITER mostrará un pantallazo inicial de presentación y, a continuación, un menú para la gestión de ficheros. El motivo de empezar con esta secuencia se debe a que la mayoría de las sesiones de trabajo se inician cargando, desde la memoria externa, un documento producido en una sesión anterior. Por lo tanto, la primera operación a efectuar consistirá en copiar el documento desde el disco a la memoria principal. Una vez en ella, todas las operaciones realizadas sobre el documento quedarán reflejadas únicamente en la memoria principal y, al finalizar la sesión, será preciso recopiar el documento desde la memoria principal hacia la memoria auxiliar, con objeto de poder utilizarlo en futuras sesiones de trabajo.

Por supuesto, la ejecución de esta carga y descarga inicial y final no es imprescindible. Una sesión de trabajo puede iniciarse perfectamente sin cargar ningún documento anterior; en este caso se tratará de producir un documento completamente nuevo. Tampoco es obligatorio finalizar la sesión de trabajo salvando el contenido de la memoria principal en disco o disquete; aunque, en este caso, si no existía una versión anterior del documento en cuestión, este no podrá volverse a utilizar. En todo caso, de existir una versión anterior, su próxima utilización no incluirá las modificaciones de la última sesión.

En definitiva, el programa EASY WRITER permite al usuario realizar tratamiento de textos, a la vez que facilita el almacenamiento y recuperación de los documentos producidos, en una memoria auxiliar.

Dada la naturaleza del programa, resulta de vital importancia para el usuario conocer perfectamente el teclado del ordenador. En una primera clasificación podemos establecer tres grandes grupos de teclas:

- CONVENCIONALES, que permiten realizar las típicas labores de escritura de un texto. Este grupo de teclas coincide con las existentes en cualquier máquina de escribir convencional.

- ESPECIALES, cuya misión no es la escritura, sino que tienen asociadas propiedades especiales.

- FUNCIONALES que al ser pulsadas desencadenan la ejecución de determinadas funciones; por lo tanto, cabe hablar de ellas como de teclas programadas (se suele identificar mediante la letra «F» seguida por un número, p.e. F1, F2, ...).

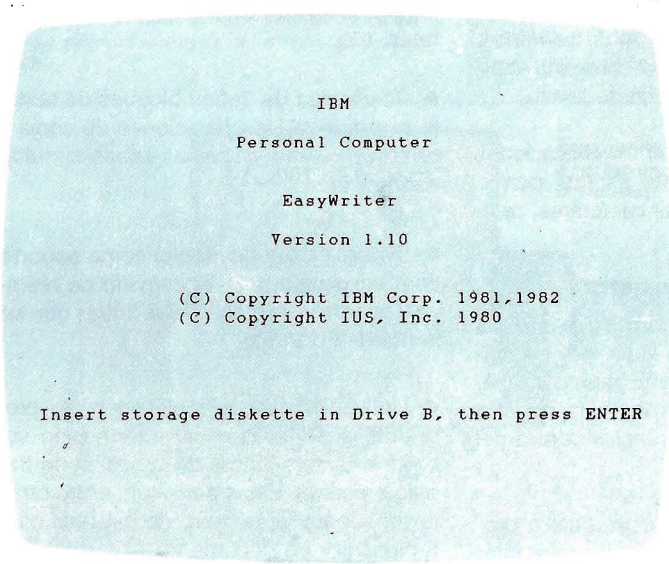
Dentro de las teclas especiales, dos de las más utilizadas en una sesión de trabajo con este programa son: [ENTER] y [SHIFT]. La primera se utiliza para indicar el fin de párrafo, aunque en determinados momentos puede tener otros efectos. La segunda, [SHIFT], tiene dos misiones fundamentales:

1. Permitir al usuario introducir la letras mayúsculas dentro del texto.
2. Seleccionar el carácter alto de una tecla con más de una utilidad. En el caso de que la tecla incluya además un tercer significado, representado en la parte inferior de la misma, para poder utilizarlo habrá que pulsar simultáneamente la tecla [ALTER] y la deseada.

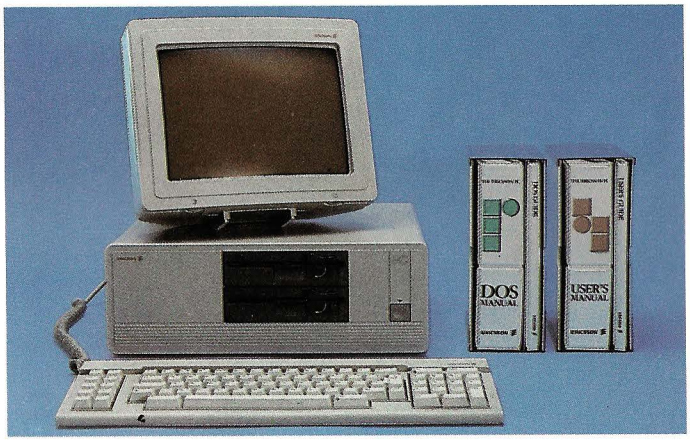
GESTION DE FICHEROS

En el entorno de los programas para el tratamiento de textos, la denominación fichero afecta al almacenamiento de los documentos creados. EASY WRITER admitirá como nombre de identificación de un fichero cualquier combinación de hasta ocho caracteres.

Como ya se apuntó, para la gestión de los ficheros el programa dispone de un menú, denominado «Easy Writer File System», en el que se ofrecen distintas opciones al usuario. Cada una de ellas está asociada a una letra distinta, de forma que el usuario se limitará a pulsar la letra correspondiente al comando que desea ejecutar. La lista de opciones está formada por trece letras, la última de las cuales, X, sirve para retornar el sistema operativo D.O.S. y, por lo tanto, finalizar la sesión de trabajo de EASY WRITER. En el próximo capítulo detallaremos con precisión los doce restantes comandos. Además de los comandos ya mencionados, el menú «File System»,



La primera pantalla informativa del EASY WRITER se obtiene sin más que teclear "EW" y pulsar la tecla ENTER, estando el ordenador bajo el control del sistema operativo. En ella se recuerda la necesidad de preparar el disquete de datos.



Las características mínimas exigidas al ordenador para que sea posible ejecutar la aplicación EASY WRITER, se concretan en 64 Kbytes de memoria RAM y una unidad para discos flexibles.

En el teclado de un IBM-PC o compatible existen, además de las teclas alfanuméricas convencionales, otras teclas especiales y de función que son utilizadas por el procesador de textos EASY WRITER.

permite seleccionar cuál de las unidades de almacenamiento será la utilizada.

MENU DE AYUDA (HELP)

Para llamar al manú de ayuda y ordenar que aparezca en la pantalla del ordenador, el usuario se limitará a pulsar la tecla funcional F1 ; inmediatamente visualizará las distintas funciones de ayuda disponibles:

- F1 (HELP ON/OFF)

Como ya apuntábamos en el párrafo anterior, la tecla F1 sirve para activar el menú de ayuda. Una vez situados en él, la misma tecla F1 sirve para abandonar el sistema de ayuda.

- F2 (PRINT FILE(S))

Sirve para imprimir el documento que se encuentra activo en la memoria principal. Su ejecución se desencadena sin más que pulsar la tecla F2 después de haber situado el editor en la pantalla.

- F3 (INSERT LINE)

Se puede utilizar para insertar una o varias líneas en blanco dentro del documento. Para ello, basta con situar el cursor en el lugar del texto donde se desea realizar la inserción y pulsar la tecla F3 tantas veces como líneas se deseen insertar.

- F4 (ADDN COMMANDS)

Al pulsar la tecla F4 aparecerá en la pantalla un nuevo menú de comandos adicionales, cuyo estudio se acometerá en el próximo capítulo.

Capacidad de los disquetes

Dos de las características más importantes de las unidades de disco flexible están relacionadas directamente con la capacidad de su soporte de almacenamiento: los discos flexibles o disquetes.

1. CARAS DE GRABACION

Algunos disquetes están preparados para ser grabados por una sola cara, en cambio otros pueden almacenar datos en ambas caras del disco. Evidentemente, la unidad de disco encargada de leer y escribir en el disquete debe estar preparada para operar sólo en una o en ambas caras.

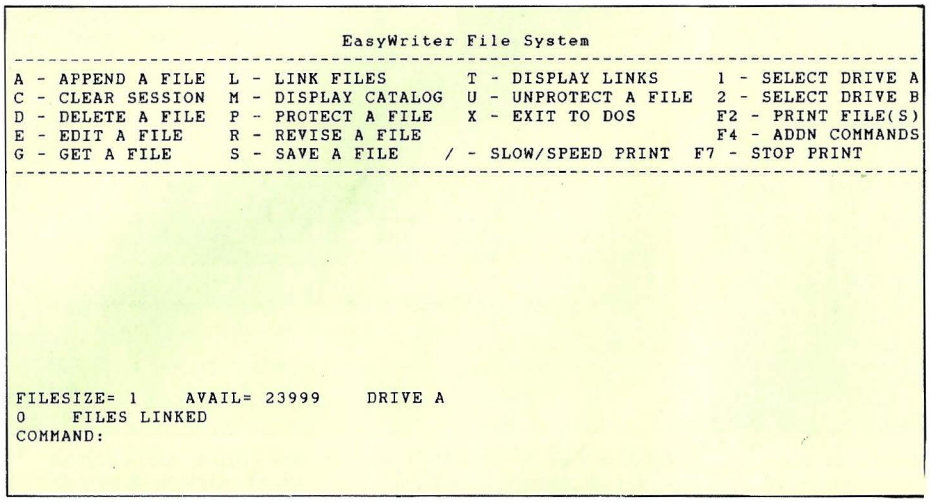
El disco magnético gira accionado por un motor dotado de un control de velocidad de gran precisión. La puesta en marcha del motor se origina con la introducción del disquete en la unidad. Para que se produzca la lectura o la escritura, es necesario que las cabezas se posicionen sobre el sector en el que se quiere operar. Por supuesto si el disquete admite información en las dos caras, el sistema de cabezas debe estar preparado para ello.

2. DENSIDAD DE GRABACION

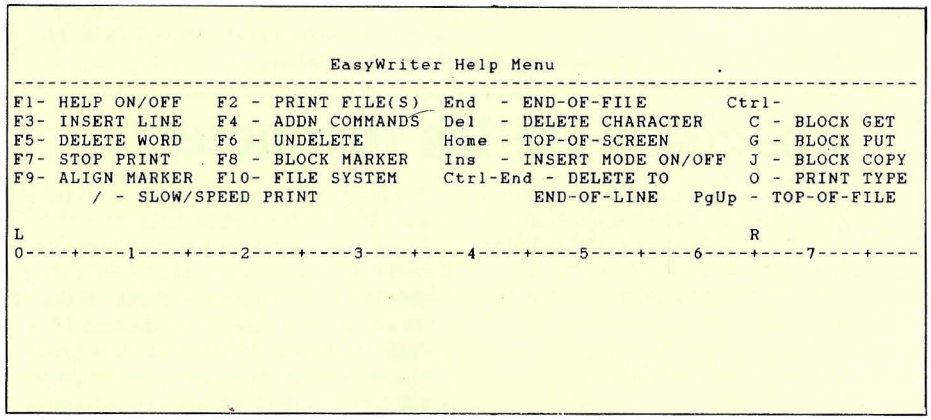
Otro dato importante para medir la capacidad de almacenamiento de un disquete es la densidad de grabación. De forma sumarial, podemos definir a la densidad de grabación como la medida de la proximidad física entre dos unidades elementales de información. En general podemos hablar de disquetes de simple y de doble densidad.

De estos dos valores —al margen de otras características— depende en buena medida la capacidad de almacenamiento de una unidad de disco flexible.

A título de ejemplo, podemos establecer una analogía entre disquete y papel. Así, un disquete de doble densidad corresponderá a un papel en el que las líneas escritas se encuentran muy próximas, mientras que un disquete de doble cara se asemejará a un papel escrito por ambas caras.



Aspecto del menú "Easy Writer File System". Este ofrece al usuario la posibilidad de activar diversos comandos relativos a la gestión de ficheros.



La aplicación incorpora un menú de ayuda (HELP), invocable mediante la tecla de función F1.

• F5 (DELETE WORD)

Si se desea borrar una palabra completa dentro del texto editado, hay que situar el cursor en el primer carácter de la palabra a borrar (utilizamos el término palabra para referirnos a cualquier serie de caracteres asociados de forma consecutiva, independientemente de que tengan o no significado) y, a continuación, se debe pulsar la tecla F5, con lo que desaparecerá la palabra del texto.

• F6 (UNDELETE)

Se puede utilizar para buscar y, en su caso, recuperar las palabras que previamente hayan sido borradas mediante la opción DELETE, para conseguir este efecto basta con pulsar la tecla F6.

• F7 (STOP PRINT)

Si el usuario acciona la tecla F7 mientras está en funcionamiento la impresora, se detendrá la impresión del documento.

• F8 (BLOCK MARKER)

Puede utilizarse para identificar o «marcar» un bloque dentro de un texto. Una vez marcado el bloque, podrán ejecutarse a continuación opciones que afecten a todo el bloque de forma unitaria.

• F9 (ALIGN MARKER)

Permite eliminar el alineamiento o justificación de una parte de un fichero. Para ello habrá que colocar una marca, mediante la tecla F9, al principio y final de bloque protegido.

• F10 (FILE SYSTEM)

Pulsando la tecla F10 se abandona el menú de ayuda y pasa a ocupar la pantalla el menú para la gestión de ficheros.

• End (END-OF-FILE)

Si se pulsa la tecla End cuando se está editando un texto, el cursor se situará automáticamente sobre el último carácter del documento y, en consecuencia, aparecerán en la pantalla las últimas líneas del mismo.

• Del (DELETE CHARACTER)

Sirve para eliminar el carácter sobre el que está situado el cursor.

• Home (TOP-OF-SCREEN)

Al pulsar la tecla Home, el cursor se desplazará inmediatamente a la primera línea y primera columna de la pantalla.

• Ins (INSERT MODE)

La tecla Ins puede utilizarse para activar o desactivar el modo de inserción. Cuando dicho modo está activado, al teclear un nuevo carácter este se situará en la posición del cursor, desplazando el resto de la línea hacia la derecha; en cambio, cuando está desactivado, al teclear un carácter, este sustituirá al que aparezca sobre el cursor, sin desplazarse el resto de la línea.

• Ctrl End (DELETE TO-END-LINE)

Pulsando automáticamente las teclas Ctrl y END se borrará la información situada desde la posición del cursor hasta el final de la línea

• Ctrl C, G, J

Pulsando simultáneamente la tecla Ctrl y C, G o J se producirá respectivamente los siguientes resultados:

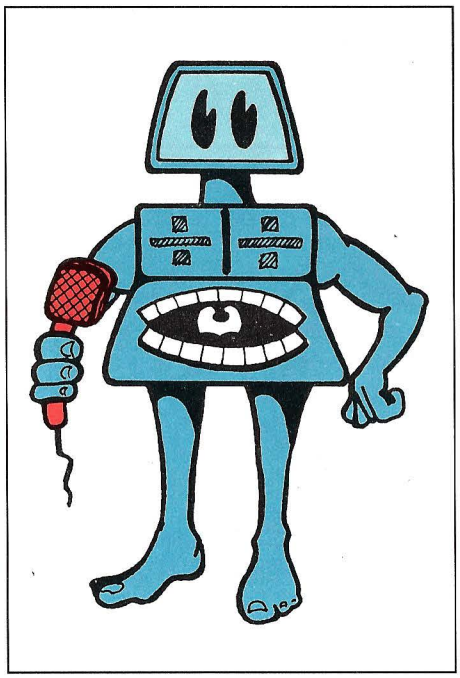
C—Obtener un bloque.
G—Colocar un bloque.
J—Copiar un bloque.

Para ejecutar estas tres opciones, es preciso que el bloque de texto esté previamente delimitado por medio del comando «Block marker», activado al accionar la tecla F8.

Introducción al sonido

Generación de sonidos con el ordenador

La revolución microinformática es un hecho incontestable. En los últimos años estamos asistiendo a una asombrosa proliferación de microordenadores. Una avalancha de equipos cuyo objetivo no es tan sólo el de automatizar aplicaciones que podríamos llamar "profesionales" o "serias". Hoy en día, el ordenador también se está convirtiendo en un electrodoméstico habitual en muchos hogares.



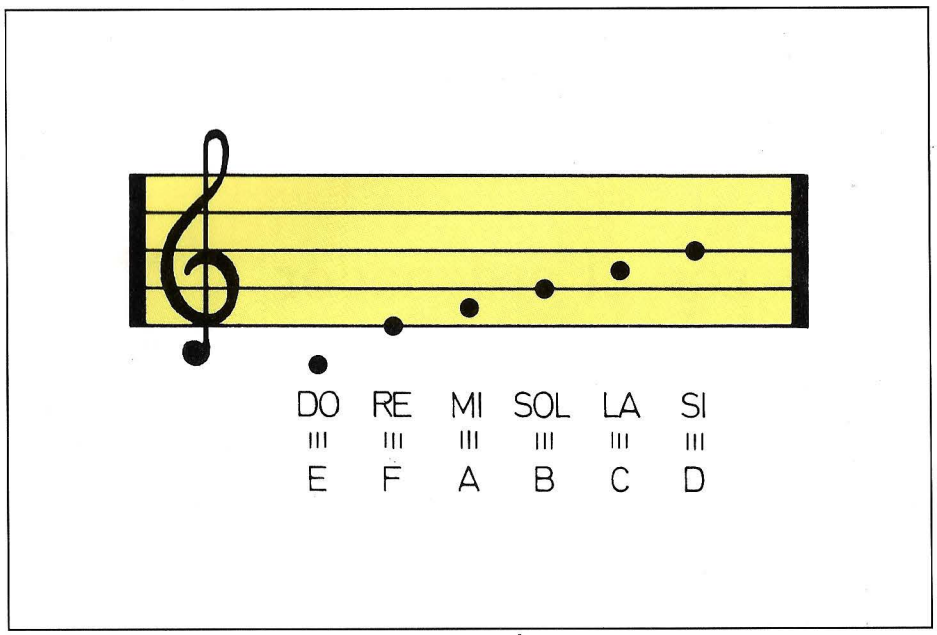
La capacidad para la generación de sonidos es una de las vertientes más desarrolladas en los modernos microordenadores. Este cúmulo de facultades se pone en práctica a través de un grupo de comandos y funciones BASIC.

El éxito de los ordenadores domésticos se debe a varios factores. En primer lugar, al convencimiento de que el futuro está en los ordenadores y es necesario familiarizarse con ellos para no quedar descolgado. En segundo lugar están las aplicaciones informáticas que podríamos llamar "menores": llevar pequeñas contabilidades, procesar textos, aplicaciones educativas y muchas otras. Por último, y con una importancia capital se encuentran los juegos. En gran medida, estos pequeños ordenadores han sustituido a las consolas de videojuegos y a otros divertimentos como instrumento de ocio y diversión para mayores y pequeños.

LA NECESIDAD DEL SONIDO

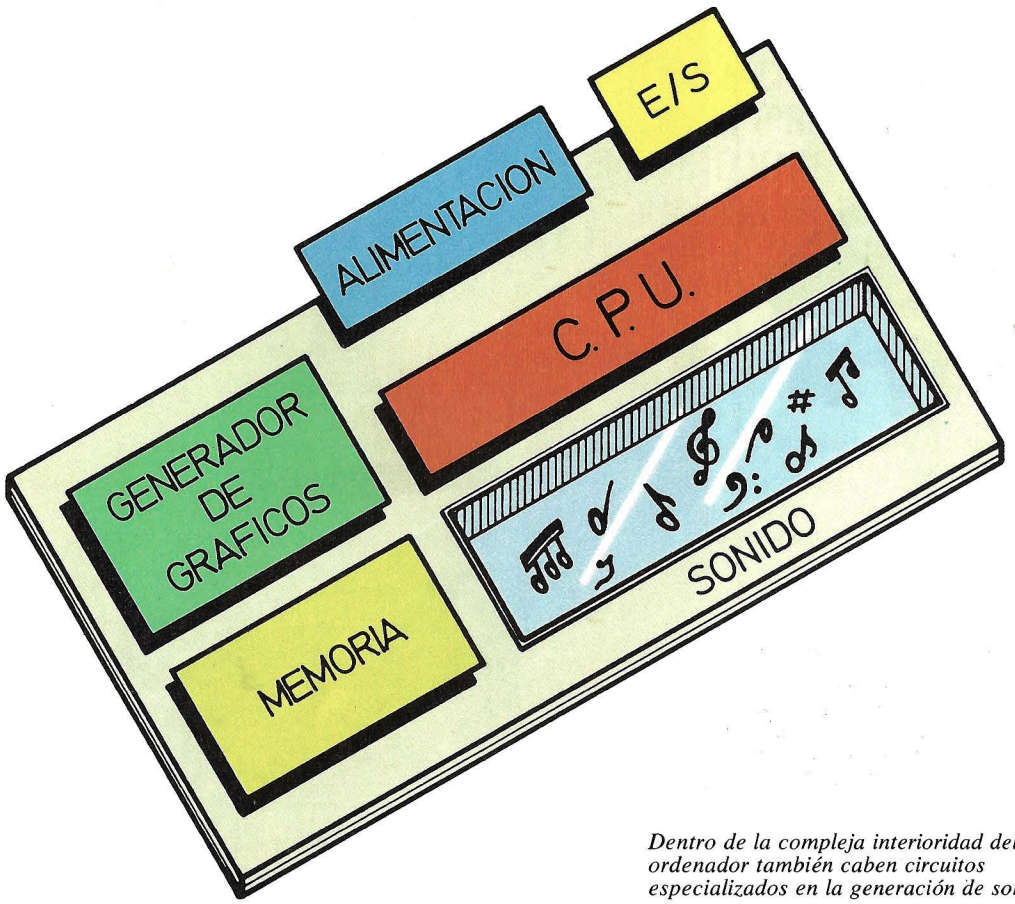
El sonido es una prestación fundamental en cualquier ordenador que pretenda de-

Recibiendo la programación adecuada, el ordenador personal también es capaz de mostrarse como un auténtico virtuoso de la interpretación musical.

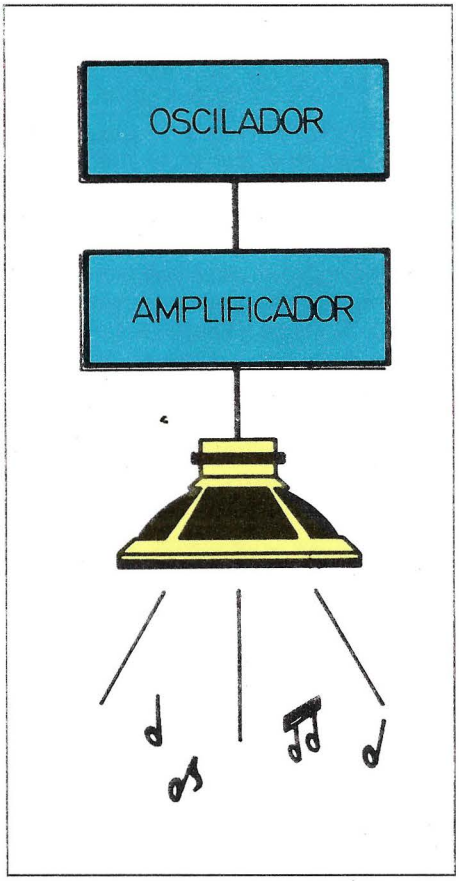


Escala musical en la que se refleja la correspondencia entre la notación habitual y la anglosajona.

dicarse a las aplicaciones de tipo lúdico o educativo. Los ordenadores más modernos ofrecen posibilidades verdaderamente asombrosas en este campo. Facultades que marcan una considerable distancia respecto a los primeros ordenadores destinados al hogar. Tal era el caso del ZX-81, el cual no estaba capacitado para emitir sonido. En la actualidad, un ordenador que no sea capaz de ofrecer al menos tres canales de sonido con un buen número de octavas por cada canal, y que no cuente con un repertorio de instrucciones BASIC especializadas en el manejo de esta posibilidad, puede considerarse como un equipo prácticamente obsoleto. Desde luego, hoy en día casi nadie puede imaginar un programa de juego en el que el sonido no esté cuidado al máximo. Este es uno de los factores a considerar a la hora de medir la calidad de un programa. La facultad de generar sonido está incluso saltando de los ordenadores creados para aplicaciones domésticas al ámbito de los



Dentro de la compleja interioridad del ordenador también caben circuitos especializados en la generación de sonidos.



La generación de sonidos por medios electrónicos exige la intervención de tres etapas básicas: el oscilador o fuente sonora, el amplificador y el altavoz o elemento reproductor del sonido.

ordenadores más serios, destinados a aplicaciones de gestión. Algunos de estos últimos empiezan a incluir entre sus prestaciones la posibilidad de emitir sofisticados sonidos que darán un mayor atractivo a los programas que para ellos se desarrollen.

DEFINIENDO TERMINOS

Al igual que sucede con otras características inherentes a los equipos informáticos, también en lo relativo al sonido existe una gran confusión, sobre todo por lo que a terminología se refiere. En los próximos párrafos se describirán algunos de los conceptos claves cuyo conocimiento resulta imprescindible para el programador.

¿Qué es el sonido? La respuesta no es difícil: se trata de perturbaciones que se propagan a través del aire y que nuestro oído es capaz de detectar. Estas perturbaciones se pueden crear de muy diversas formas: por medios mecánicos, por nuestra voz... y también por medios electrónicos. Para lograr tal objetivo, los especialistas emplearon unos circuitos electrónicos, llamados osciladores, capaces de generar ondas de una determinada frecuencia; ondas que se harán audibles al reproducirlas a través de un altavoz.

Cuando estas ondas se encuentran dentro de la gama de frecuencias audible por el ser humano, se convierten en lo que llamamos sonido.

La frecuencia, como se puede apreciar, es uno de los parámetros más importantes de una onda. Se trata de una medida de la velocidad a la que se producen las variaciones de esa perturbación. Sin embargo, estas ondas no sólo se caracterizan por su frecuencia, sino también por otros factores como son la forma de la onda, su amplitud y la potencia de la misma.

¿Qué significa que un ordenador posea tres canales de sonido y uno adicional de ruido? Definamos en primer lugar qué se entiende por canal.

Un canal de sonido se refiere a una fuente sonora; de tal forma que un ordenador con tres canales de sonido poseerá tres fuentes emisoras de sonido. Ello no quiere decir que posea tres altavoces, sino que el sonido se genera mediante tres osciladores, cuyas salidas, luego, son mezcladas y enviadas, por lo general, a un solo altavoz. El canal de ruido se concreta en otro oscilador que genera una serie de ondas, aunque esta vez de una forma un tanto peculiar. Este último resulta adecuado para crear sonidos efectistas: lo que podríamos denominar efectos especiales.

Dentro del dispositivo que produce el so-

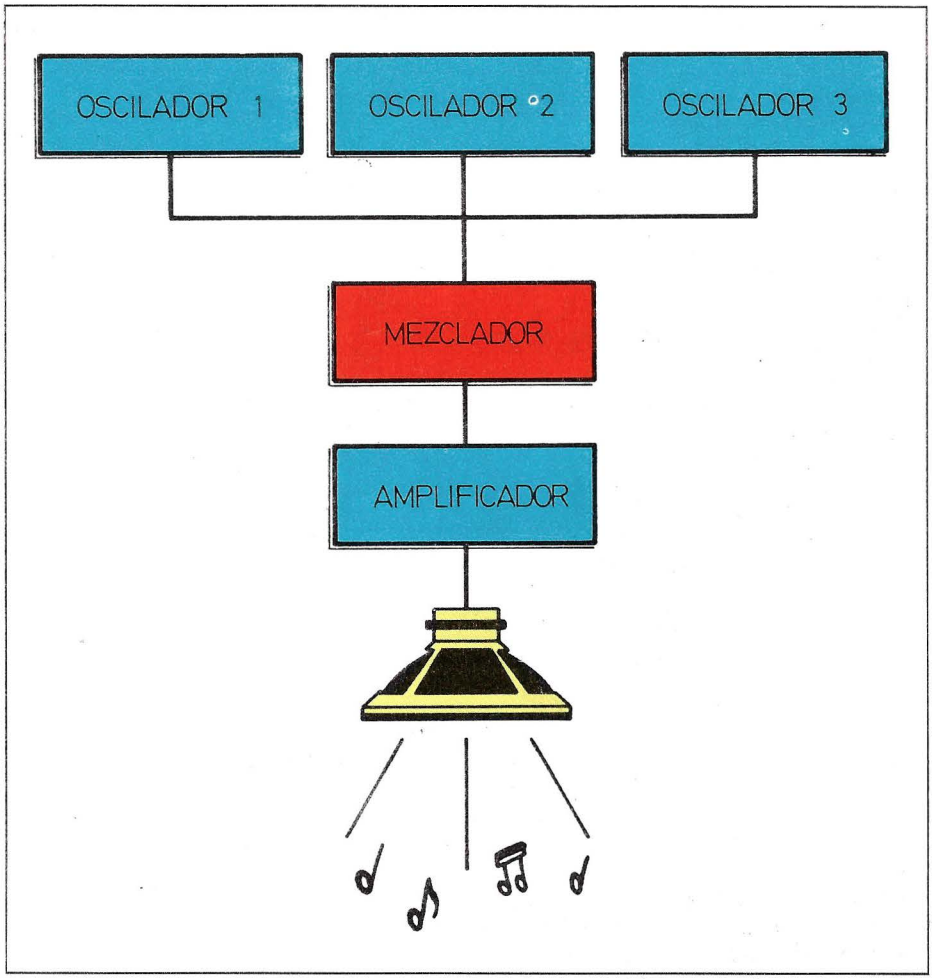
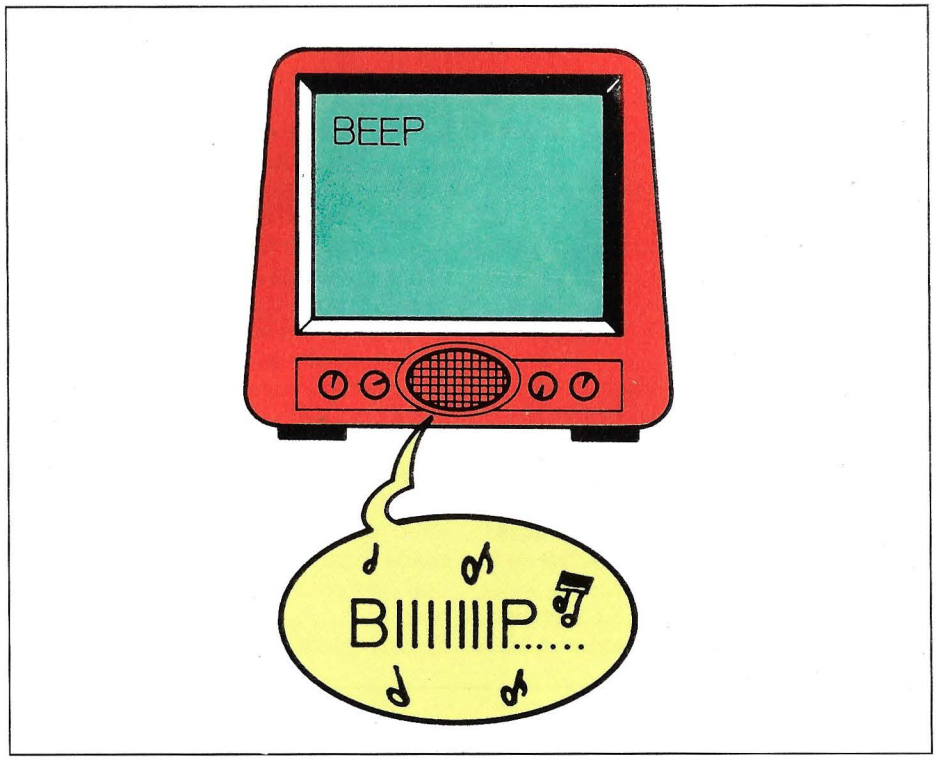
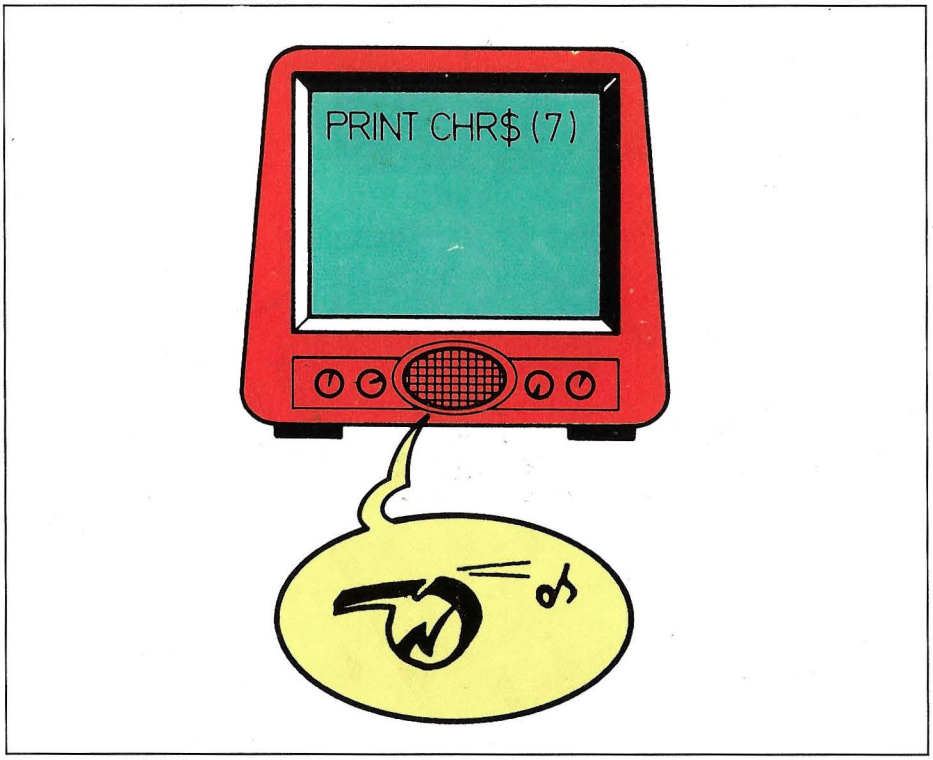


Diagrama de bloques de los circuitos habitualmente encargados de la generación de sonidos en el ordenador.



Algunos equipos ofrecen al usuario la posibilidad de ejecutar comandos BASIC que proporcionan sonidos fijos.



En ciertos microordenadores la impresión del carácter ASCII 7 no produce alteración en la pantalla, sino tan sólo la emisión de un pitido.

nido es preciso distinguir dos partes fundamentales: los osciladores, que producen el sonido, y los amplificadores y altavoces que lo acondicionan para que seamos capaces de oírlo.

Hay ordenadores que en su interior incluyen ambas cosas; pero también los hay que prescinden de la segunda zona. En este último caso, el sonido se obtiene aprovechando los amplificadores y altavoces del televisor o monitor conectado al equipo. Al efecto, la señal que el ordenador envía al televisor no es sólo la de vídeo, sino también la de sonido.

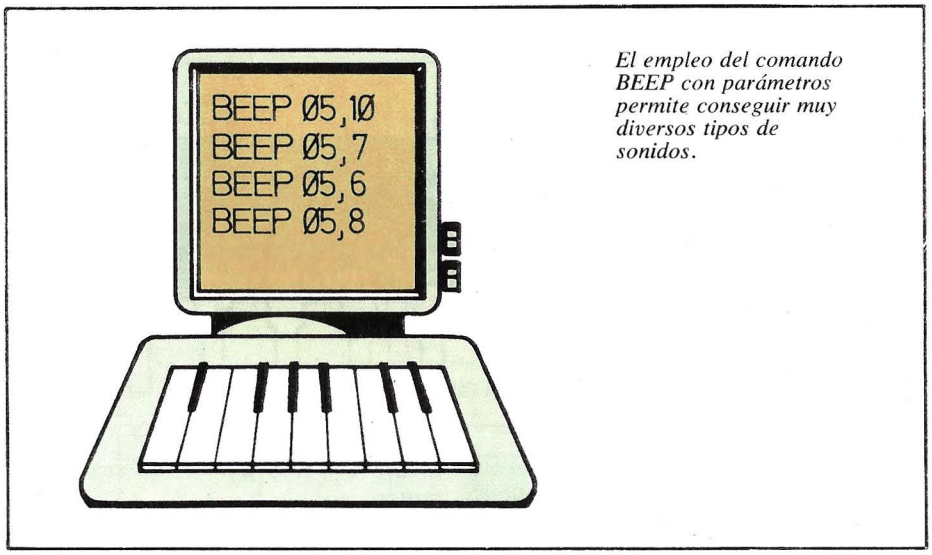
Respecto a la reproducción final del sonido cabe puntualizar algunos detalles de interés.

Para empezar, los altavoces que suelen incorporar los ordenadores son generalmente pequeños y emiten un sonido de poca importancia. Además, este sonido no suele ser regulable, esto es: no se puede ajustar el volumen del mismo una vez que éste ha sido programado. En los ordenadores que aprovechan el amplificador y el altavoz del televisor o monitor externo, la cosa cambia; éstos poseen, por lo general, un mando para controlar el volumen, mando que el usuario puede regular a voluntad.

La diferencia fundamental, de todas formas, reside en la calidad. En el caso de que el sonido se obtenga a través del receptor de TV, tiene lugar toda una serie de procesos de codificación y decodificación de la información sonora que hacen que el sonido resultante pierda calidad. También existen ordenadores que disponen de salidas para equipos de alta fidelidad (a veces en estereofonía) de los cuales cabe esperar un sonido excelente.

Son muchos los términos del lenguaje musical que se emplean en el campo de los ordenadores. Por ejemplo, una nota es un sonido de una determinada frecuencia. A su vez, una octava es un conjunto de frecuencias. Cuando se dice que un ordenador tiene más octavas que otro, se está indicando que el rango de frecuencias en el que puede trabajar es más amplio. Las notas musicales que caben dentro de una octava son esencialmente siete; aunque, de hecho, estas notas admiten variantes de tal forma que dentro de una octava hay en realidad más de siete sonidos. Por lo general, para emitir una determinada nota hay que indicar a la máquina de qué nota se trata y en qué octava se encuentra. El nombre de la nota será el mismo en cual-

quiera de las octavas, depende tan sólo de su posición relativa dentro de la misma.
Hay que tener en cuenta otro detalle de importancia a la hora de leer el manual que acompaña a cada ordenador. Habitualmente, se utiliza una escala musical en la que cada nota recibe un nombre: DO, RE, MI, FA, SOL, LA y SI. No obstante, la notación que suele encontrarse en los manuales (normalmente en inglés) no es ésta. Se trata de la misma escala, pero con la diferencia de que los nombres de las notas aparecen como: C, D, E, F, G, A y B. Esta es la notación habitual anglosajona.



El empleo del comando BEEP con parámetros permite conseguir muy diversos tipos de sonidos.

ESBOZANDO SONIDOS

Las instrucciones para el manejo del sonido, al igual que ocurre con las destinadas al tratamiento de gráficos, no están en absoluto estandarizadas. Este hecho hace que los diferentes ordenadores presenten instrucciones muy diversas para el manejo del sonido. Incluso ciertos ordenadores carecen de instrucciones BASIC de esta categoría, lo que obliga a recurrir al acceso directo a memoria.
Dentro del microordenador, suele existir algún circuito integrado al que se le encomienda la misión de apoyar las tareas de generación sonora. A este "chip" tan sólo hay que completarlo con algún dispositivo que amplifique la pequeña señal que proporciona, antes de mandar la señal resultante al altavoz.
Antes se indicó que tal amplificación no suele realizarse en el propio ordenador, sino que la señal se envía al televisor para que éste se encargue de amplificarla.
Para comunicar al circuito integrado "experto en sonido" la información que éste precisa para realizar su tarea, se utilizan unas determinadas posiciones de memoria. Posiciones a las que tiene acceso el generador de sonido y de donde toma los datos que debe depositar en sus registros.
Empezaremos hablando del caso en el que es necesario acceder directamente a la memoria principal del ordenador para programar los circuitos que éste posee para producir sonidos. Tal operación es realizable en cualquier aparato. No obstante, siempre que sea posible se huirá de ello; por la sencilla razón de que el procedimiento resulta un tanto complicado. En principio, suele ser necesario acceder a una posición de memoria para fijar el nivel de volumen deseado. Acto seguido, es preciso acceder a otras posiciones para fijar la nota, la duración, la envolvente, etc. El problema reside en que para ello es necesario recordar las posiciones de memoria asociadas a cada uno de estos registros, y los valores adecuados que han de introducirse en los mismos. Además, en muchos casos estos valores están relacionados con la frecuencia de emisión por medio de ecuaciones matemáticas más o menos complicadas.
Una alternativa sencilla para obtener sonidos consiste en acudir a una serie de palabras BASIC, que al ser interpretadas por el ordenador dan lugar a un sonido determinado y fijo, no admitiendo por lo tanto ningún parámetro. Tal es el caso, por ejemplo, de la palabra BEEP. Esta da nombre a un comando cuya ejecución hará que el altavoz emita un determinado sonido durante un cierto intervalo de tiempo. También existen otras palabras como ZAP, EXPLODE, etc. Dentro de esta misma categoría de herramientas para generar sonidos elementales cabe considerar al carácter ASCII 7; un carácter de control que produce la emisión de un pitido audible al introducirlo en el ordenador a través de una instrucción PRINT:

PRINT CHR$(7)

AVANZANDO EN EL SONIDO

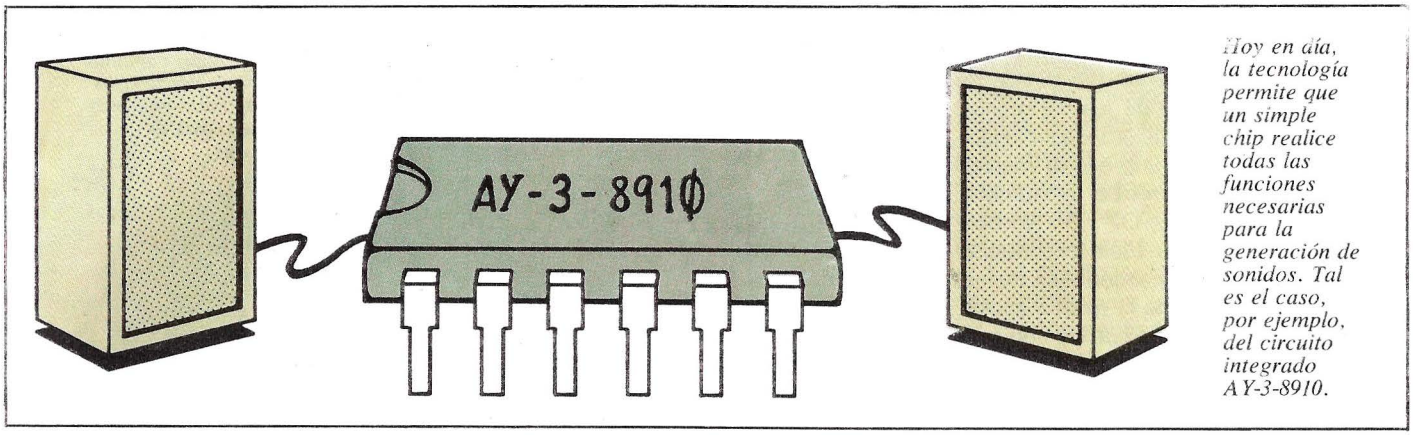
El lenguaje BASIC da entrada a otro tipo de instrucciones más elementales para la programación de sonidos. Como ya suele ser habitual —aunque lamentablemente—, las coincidencias en el repertorio de instrucciones de los distintos intérpretes

BEEP

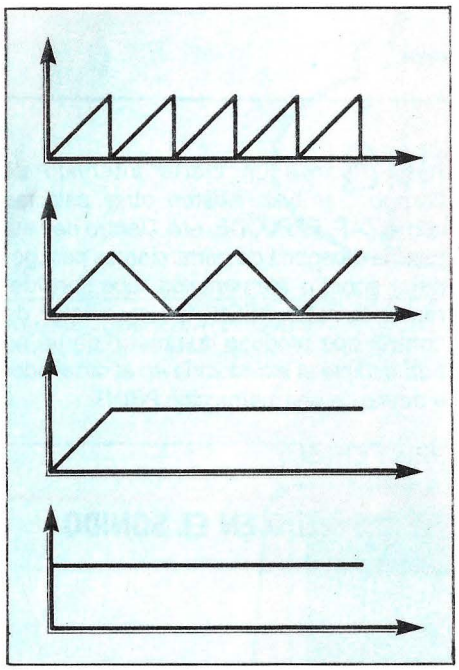
Genera un sonido con una duración y una frecuencia indicadas en su argumento.

Formato: BEEP <duración>,<frecuencia>

Ejemplos:
```
10 BEEP 0.5,7
50 BEEP 1,5
```



Hoy en día, la tecnología permite que un simple chip realice todas las funciones necesarias para la generación de sonidos. Tal es el caso, por ejemplo, del circuito integrado AY-3-8910.



Distintas formas o patrones de la envolvente de volumen que puede utilizarse para la síntesis de sonidos especiales.

BASIC en este punto son casi inexistentes.

Vamos a empezar hablando de la instrucción BEEP, pero esta vez acompañada por dos parámetros. Uno de ellos servirá para indicar la frecuencia de la nota deseada y el otro señalará la duración que se desea que tenga el sonido. En este caso no será necesario seleccionar la octava deseada, sino que bastará con dar números cada vez mayores con lo que se irá pasando de una octava a la siguiente.

Uno de los problemas inherentes a este tipo de instrucciones es que al emplear números se pierde de vista la nota y octava con la que se está trabajando. No obstante, el principal problema reside en que cuando se ejecuta esta instrucción el programa se detiene hasta que se concluye su tratamiento. Así, pues, el sonido no es independiente del proceso.

Este sistema es el que utiliza un ordenador tan popular como el ZX-Spectrum. A continuación se incluyen, a título de ejemplo, algunos programas destinados a este ordenador.

El primero de ellos permite recorrer una octava completa:

```
10 FOR I=0 TO 11
20 BEEP 0.5,I
30 NEXT I
```

El mismo programa puede ejecutarse para una octava más alta, lo que permitirá observar la diferencia entre las notas reproducidas en una u otra octava:

```
10 FOR I=0 TO 11
20 BEEP 0.5,I+12
30 NEXT I
```

Análogamente, la octava seleccionada puede ser una inferior:

```
10 FOR I=0 TO 11
20 BEEP 0.5,I-12
30 NEXT I
```

Cuando se trata de definir una composición relativamente larga, es conveniente, en lugar de escribir tantas instrucciones BEEP como notas se deseen interpretar, encerrar esta instrucción dentro de un bucle. La lectura de los datos se realizará mediante instrucciones READ/DATA; en estas últimas estarán escritos los valores que conducirán al resultado apetecido. Veamos un ejemplo:

```
100 FOR I=1 TO 100
110 READ A,C
```

REGISTROS DEL CHIP DE SONIDO AY-3-8910

Registro	Función		
0	Frecuencia canal A	Control preciso	0-255
1	Frecuencia canal A	Control de escala	0-18
2	Frecuencia canal B	Control preciso	0-255
3	Frecuencia canal B	Control de escala	0-18
4	Frecuencia canal C	Control preciso	0-255
5	Frecuencia canal C	Control de escala	0-18
6	Frecuencia de ruido		0-31
7	Programación de tono de ruido para cada canal		0-63
8	Volumen del canal A		0-16
9	Volumen del canal B		0-16
10	Volumen del canal C		0-16
11	Frecuencia de la envolvente	Control preciso	0-255
12	Frecuencia de la envolvente	Control de escala	0-255
13	Envolvente de ataque		0-255

TABLA DE CONVERSION

ORDENADOR	SONIDOS FIJOS		SONIDOS VARIABLES	ACCESO A LOS REGISTROS
	COMANDOS DIRECTOS	CHR$ (7)	BEEP n1, n2	SOUND <r>, <c>
APPLE II (APPLESOFT)	—	CHR$ (7) o CTRL+G	—	—
APRICOT (M-BASIC)	—	—	—	—
ATARI	—	—	SOUND n1, n2, n3, n4 (1)	—
CBM 64	—	—	—	—
DRAGON	—	—	SOUND n2, n1	—
EQUIPOS MSX	BEEP	CHR$ (7)	—	SOUND <r>, <c>
HP-150	—	CHR$ (7)	—	—
IBM PC	—	CHR$ (7)	SOUND n2, n1	—
MPF	—	—	—	—
NCR DM-V (MS-BASIC)	—	CHR$ (7)	—	—
NEW BRAIN	—	—	—	—
ORIC	SHOOT, ZAP, PING, EXPLODE	CHR$ (7) o CTRL+G	MUSIC n1, n2, n3, n4 (2)	—
SHARP MZ-700 (MZ-BASIC)	—	—	—	—
SINCLAIR QL	—	—	BEEP (3)	—
SPECTRAVIDEO	BEEP	CHR$ (7)	—	SOUND <r>, <c>
ZX-SPECTRUM	—	—	BEEP n1, n2	—

n1: duración. n2: nota. <r>: registro. <c>: contenido.
(1) SOUND n1, n2, n3, n4. n1: canal. n2: nota. n3: distorsión. n4: volumen.
(2) MUSIC n1, n2, n3, n4. n1: canal. n2: octava. n3: nota. n4: volumen.
(3) BEEP [n1, n2 [, n3, n4, n5 [, n6 [, n7 [, n8]]]]]. n1: duración. n2: nota. n3: nota secundaria; entre ella y n2 oscilará el sonido. n4: intervalo entre oscilaciones de notas. n5: define el tamaño de cada oscilación. n6: repetición. n7: rizado. n8: aleatorio.

```
120 BEEP A/2.50,C
130 NEXT I
140 DATA 0.25,7,0.25,9,0.5,11,0.5,14,0.75,14,0.25,
    16,0.5,14,0.5,11,0.75,7
```

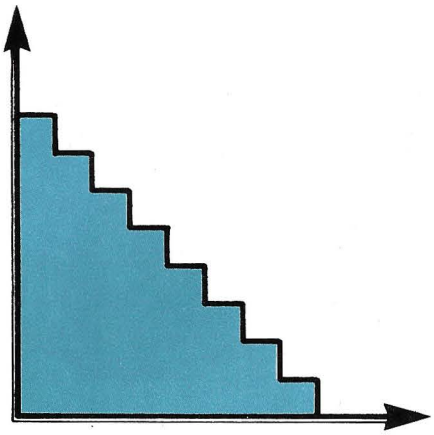
Al terminar de leer este bloque de instrucciones el ordenador se detendrá presentando un mensaje de error, debido a que no hay suficientes datos para solventar todas las pasadas que hay que dar al bucle. Esta breve rutina se ha confeccionado de forma que sea utilizable para reproducir composiciones de cualquier longitud, sin más que añadir sucesivas líneas DATA con los datos oportunos. Cuando la composición esté terminada, será suficiente con contar los datos que la integran y ajustar el valor final del bucle en función de dicho número.

SOUND

Permite acceder a los registros de sonido para su programación.

Formato: SOUND <registro>,<contenido>

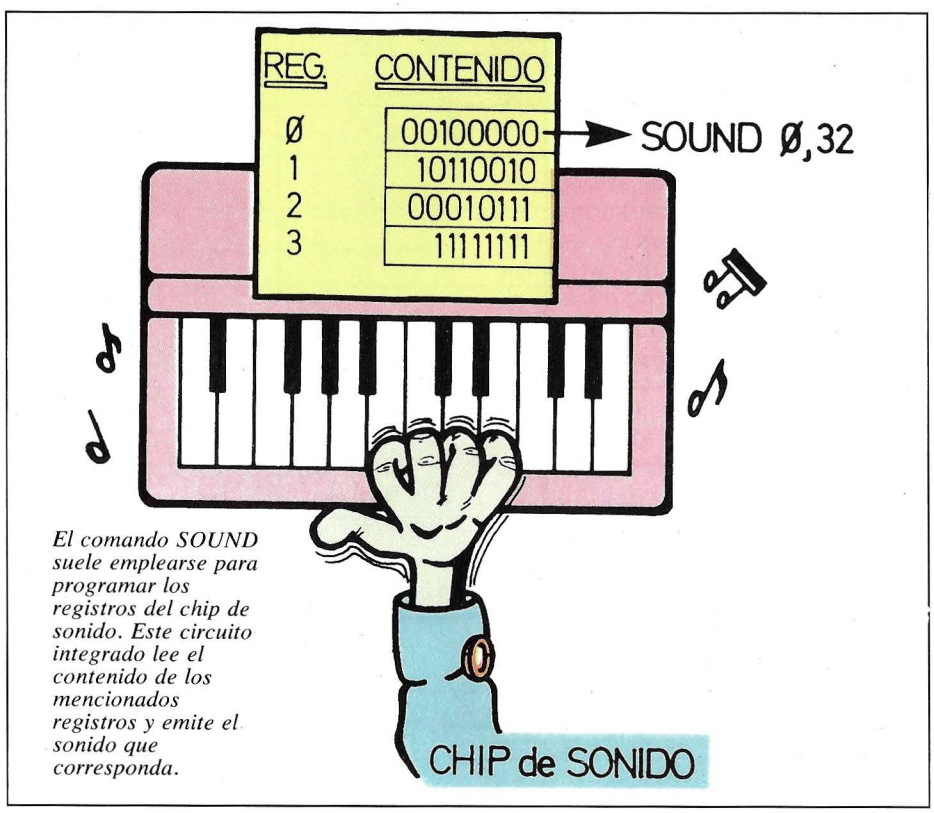
Ejemplos: 20 SOUND 5,3
70 SOUND 11,128



En un ordenador, el nivel de volumen del sonido es regulable en sucesivos "escalones" o saltos de forma digital.

ACCESO A LOS REGISTROS DE SONIDO

Al principio del capítulo se mencionó que ciertos ordenadores exigían el acceso directo a los registros de sonido para programar la producción del mismo. Por otra parte también existen ordenadores con instrucciones BASIC que facilitan el trabajo directo con los registros de sonido, sin necesidad de recurrir a los laboriosos POKES. Veamos un ejemplo práctico, utilizando el chip de sonido AY-3-8910; uno de los más empleados en los modernos ordenadores.



El comando SOUND suele emplearse para programar los registros del chip de sonido. Este circuito integrado lee el contenido de los mencionados registros y emite el sonido que corresponda.

Si el ordenador posee más de un canal de sonido, las opciones se multiplican. Existen instrucciones previas que seleccionan uno y otro canal; e incluso instrucciones que admiten más parámetros, uno para cada canal. Veamos un ejemplo de esto último:

BEEP (2,4), (3,1), (2,4)

en donde cada uno de los paréntesis se refiere a un canal.

El referido chip dispone de 14 registros accesibles por el programador. Cada uno de estos registros contiene información específica que es utilizada por el circuito integrado para saber el sonido que ha de generar. La tabla adjunta incluye una lista de los referidos registros, señalando las funciones asociadas a cada uno de ellos.

Frecuencia

Los registros del 0 al 5 se utilizan por pares para determinar la frecuencia del sonido a emitir. Se rigen por la siguiente fórmula:

X=(1.78977*10 EXP 6)/16*F
RB=INT (X/256)
RA=X−RB*256

en donde RA puede coincidir con uno de los registros 0, 2 ó 4, y RB con los registros 1, 3 ó 5.

Ruido

El registro seis selecciona la frecuencia central del ruido, por medio de un valor que ha de ser buscado en la correspondiente tabla; en ella, cada frecuencia tiene asignado un valor específico.

Selección del tipo de canal

El registro al efecto se emplea para indicar al generador el uso que se va a dar a cada uno de los canales. Cada canal puede ser utilizado para emitir sonido o bien ruido. Para efectuar esta selección se utilizan los seis bits menos significativos del registro. Según esté a uno o a cero cada uno de estos bits, indicará si el canal correspondiente está seleccionado o no. La siguiente tabla revela la utilidad de los seis bits del registro. La tabla comienza por el bit menos significativo y termina con el sexto bit (el último que se utiliza).

0 Canal A de sonido
1 Canal B de sonido
2 Canal C de sonido
3 Canal A de ruido
4 Canal B de ruido
5 Canal C de ruido

Evidentemente, el siguiente paso consiste en transformar la palabra binaria obtenida a notación decimal, e incluirla en la instrucción oportuna.

Volumen

Los registros del 8 al 10 preseleccionan el volumen con el que ha de emitir cada uno de los canales de sonido. Sus valores van del 0 al 15. Cuando su valor es 16, el control pasa al generador de envolventes.

Envolvente

La envolvente define la forma en la que cambia el nivel de volumen del sonido producido. El chip que nos ocupa tiene ocho patrones diferentes, cuyo aspecto puede observarse en la figura que acompaña al texto.

Los registros 1 y 12 fijan el período de la envolvente, mientras que el registro 13 sirve para seleccionar el patrón de la misma.

Forth (7)

El entorno de diálogo

La finalidad de un programa no es otra que instruir al ordenador para que éste realice un determinado tratamiento de los datos aportados. Estos datos pueden estar incluidos dentro del programa, con lo cual el proceso resulta sencillo. No obstante, la situación más normal es la de construir el programa sin contar con los datos específicos a procesar. Ello supone que será preciso introducirlos cada vez que se ejecute el programa, ya sea previamente o bien durante el proceso.
En el primero de los casos es suficiente con colocar los datos en la pila, y ordenar las operaciones oportunas para que el programa sea capaz de tratar esos valores. En el segundo caso los datos han de introducirse durante el proceso, y para ello es necesario utilizar las palabras que el FORTH brinda a tal efecto. Estas palabras constituyen, precisamente, el objeto del presente capítulo.

TRABAJANDO CON EL BUFFER DE ENTRADA

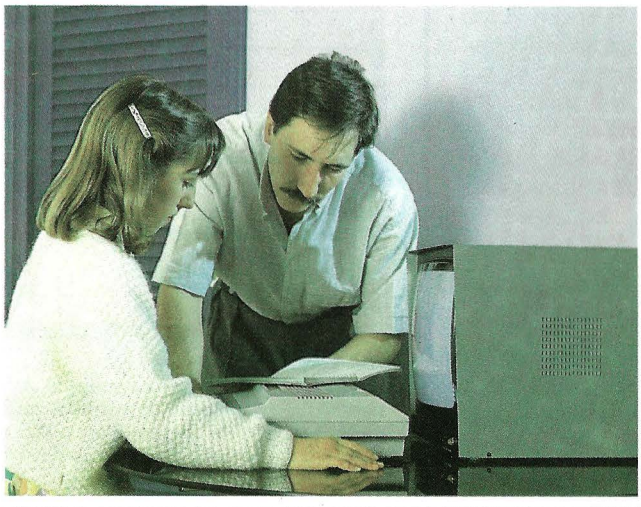
El buffer de entrada es una zona de memoria que actúa a modo de intermediario entre el usuario y el ordenador. Lo que el programador teclea al editar un programa va escribiéndose en el buffer, y pasa a la memoria del ordenador cuando se acciona la tecla de retorno de carro. En consecuencia, parece obvio que una forma de pasar datos a un proceso consiste, sencillamente, en explotar la función propia del buffer de entrada.
Existen dos instrucciones que sirven para introducir datos en el ordenador a través del buffer, éstas coinciden con las palabras QUERY y RETYPE. Ambas detienen el proceso en curso y facilitan el acceso al buffer. La diferencia entre ambas es que la primera empieza borrando el contenido previo del buffer de entrada, y a continuación permite acceder al mismo. La segunda no borra ni altera el contenido previo, por lo que será posible editar el contenido del buffer, introduciendo las modificaciones oportunas. En todo caso, ninguna de ambas palabras afecta a la pila. Desde luego, el siguiente paso es tratar el buffer de entrada de la forma más adecuada. Para ello, el FORTH dispone de cuatro palabras que será posible emplear según las necesidades específicas. Estas son:

LINE, NUMBER, FIND y WORD

El buffer de entrada actúa como intermediario entre el usuario y el ordenador.

La primera de ellas (LINE), toma sencillamente el contenido del buffer de entrada y lo interpreta, produciendo el mismo resultado que si se hubieran tecleado esas mismas palabras en modo inmediato. El uso de LINE resulta muy flexible, aunque no deja de plantear ciertos problemas, ya que la palabra o palabras introducidas en el buffer pueden originar tras su interpretación y ejecución un resultado no previsto e indeseado. Por ejemplo, el resultado puede ser la obtención de dos números, en lugar de uno solo como en principio podía estar previsto.

Veamos un ejemplo de definición en el que intervienen las palabras clave QUERY y LINE:



El diálogo con la máquina en FORTH se establece, normalmente, recurriendo al teclado como vía de entrada. En tal caso, entra en actividad el denominado "buffer de entrada": una zona de memoria que actúa a modo de intermediario entre el usuario y el ordenador.

```
: ENTRADA
CR
." INTRODUCE LO QUE DESEES"
QUERY LINE
CR
." YA SE HA EJECUTADO"
;
```

Y otra nueva definición:

```
: SACA
CR
." SE EJECUTA LA PALABRA SACA"
;
```

Al ejecutar la primera de las palabras definidas:

ENTRADA <CR>

se obtendrá la siguiente respuesta en la pantalla:

```
INTRODUCE LO QUE DESEES
```

La respuesta del usuario puede ser la siguiente:

SACA <CR>

Con lo que el ordenador responderá:

```
ENTRADA
INTRODUCE LO QUE DESEES
SE EJECUTA LA PALABRA SACA
YA SE HA EJECUTADO OK
```

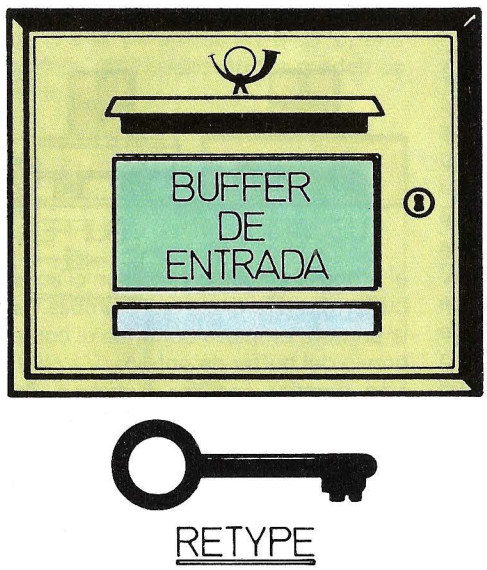
Se observará, pues, que la palabra SACA ha sido ejecutada en medio del proceso de la palabra ENTRADA.

NOTA: A partir de ahora, los listados aparecerán sin figurar en ellos los retornos de carro (<CR>) a introducir como fin de línea.

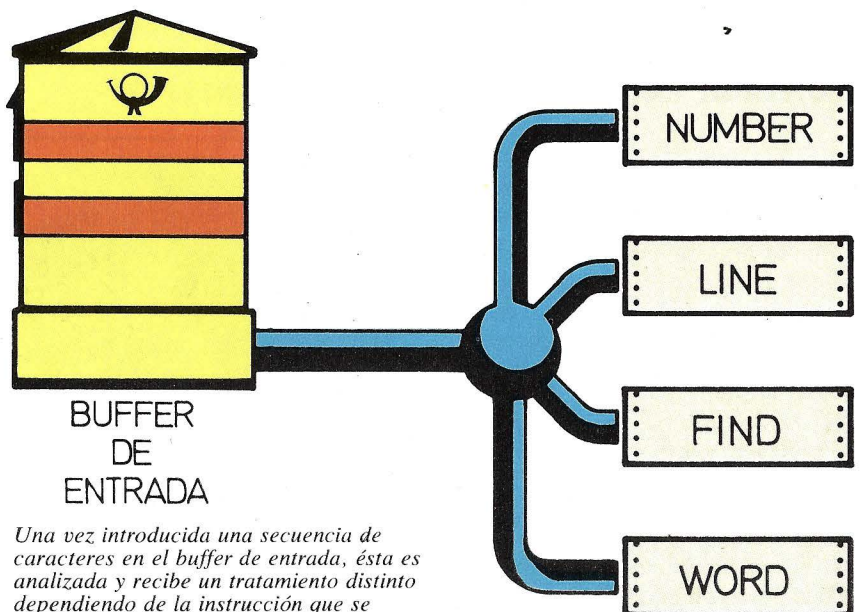
INTRODUCIENDO NUMEROS

La palabra NUMBER se utiliza para introducir números en la pila desde el buffer de entrada. NUMBER analizará el contenido del buffer de entrada; si no encuentra ningún número al principio del mismo, entenderá que se trata de un cero. El número en cuestión es tomado del buffer de entrada y depositado en la pila, colocando además un código encima del número; el código servirá para precisar si el número es entero (código 4102) o de coma flotante (código 4181).

A título de ejemplo, se define a continuación una palabra adecuada para introducir números enteros en la pila, dando además



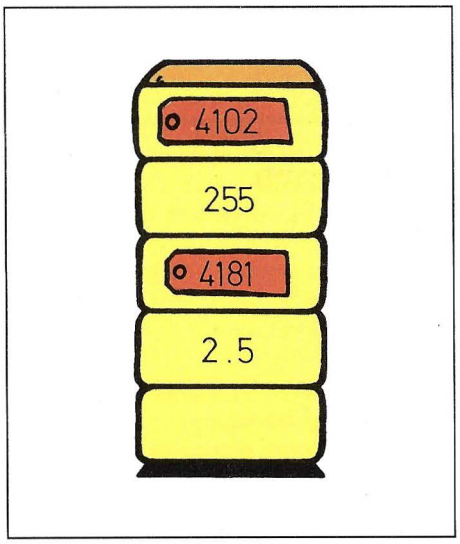
La palabra RETYPE es la llave que permite acceder al buffer de entrada para modificar su contenido.



Una vez introducida una secuencia de caracteres en el buffer de entrada, ésta es analizada y recibe un tratamiento distinto dependiendo de la instrucción que se utilice para actualizar el buffer.

la posibilidad de editar el número incorrecto que pudiera haberse introducido:

```
: ENTERO
BEGIN
RETYPE NUMBER DUP 4181=
IF
." INCORRECTO"
DROP DROP DROP 0
THEN
UNTIL
;
```



La palabra NUMBER acepta números de buffer de entrada y los deposita en la pila, situando encima de ellos una etiqueta que señala si se trata de un número de punto fijo o de punto flotante.

MAS SOBRE PALABRAS

La palabra FIND genera un resultado muy distinto respecto al obtenido con las dos anteriores. FIND toma una palabra del buffer de entrada, y la busca en el diccionario; en el caso de encontrarla, deposita en la cima de la pila la dirección de comienzo de dicha palabra. Esta dirección resulta muy útil en orden al posterior uso de la palabra EXECUTE, la cual ejecutará la palabra cuya dirección de comienzo se encuentre en la cima de la pila.
Veamos un ejemplo:

```
: DIRECCION
QUERY FIND
.
;
```

Al ejecutarlo:

```
DIRECCION <CR>
VLIST <CR>
```

la pantalla mostrará el siguiente resultado:

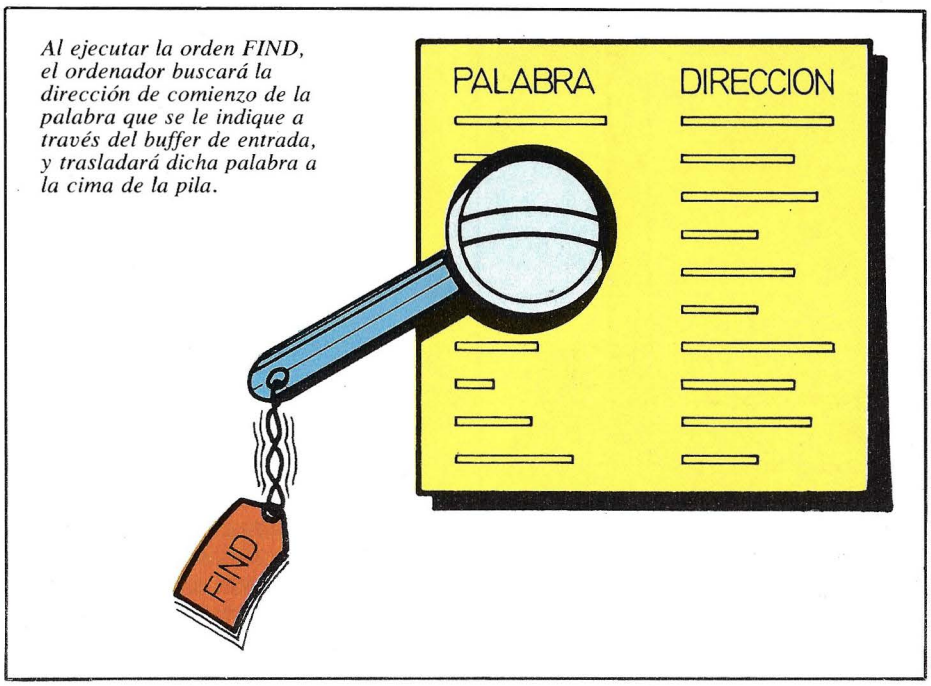
```
DIRECCION 1581 OK
```

A partir de esta dirección es posible ya ejecutar la referida palabra mediante la orden EXECUTE:

```
1581 EXECUTE <CR>
```

y de inmediato se obtendrá una lista por pantalla de todas las palabras definidas en el diccionario FORTH.

SECUENCIAS DE CARACTERES QUE NO SON PALABRAS

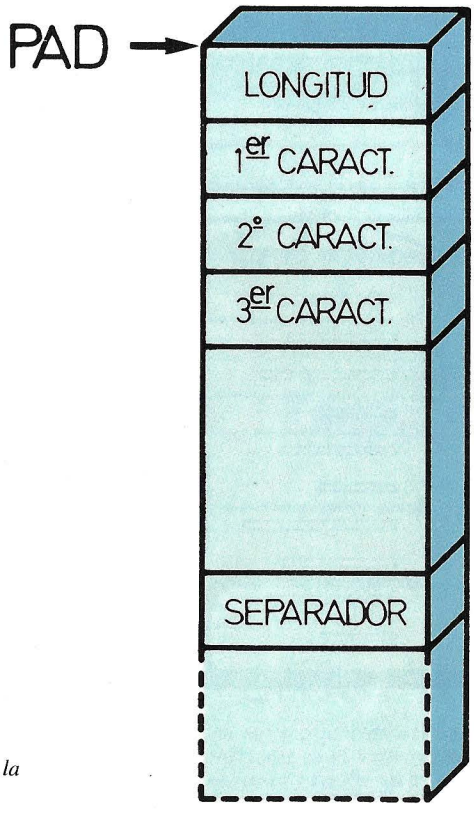


Al ejecutar la orden FIND, el ordenador buscará la dirección de comienzo de la palabra que se le indique a través del buffer de entrada, y trasladará dicha palabra a la cima de la pila.

Otra palabra FORTH de indudable interés es WORD. Su cometido se concreta en almacenar en la memoria del aparato algunas secuencias de caracteres que no constituyen palabras ni números. La zona de memoria donde se almacenan estos datos recibe el nombre de PAD. En ella, los caracteres son memorizados mediante su correspondiente código ASCII.
Para conocer la dirección de comienzo de esta zona de memoria se puede emplear la palabra PAD, la cual transferirá la dirección de comienzo a la cima de la pila.
La secuencia de acciones asociada a la ejecución de la palabra WORD es la siguiente:

1. Rellena el PAD con espacios, eliminando cualquier contenido previo.
2. Toma el código ASCII del carácter a emplear como delimitador de entradas, extrayéndolo de la cima de la pila (por lo general, suele coincidir con el espacio cuyo código es 32). El delimitador sirve al

TABLA DE ORDENES FORTH		
PALABRA	DESCRIPCION	TIPO
QUERY	Permite introducir caracteres en el buffer de entrada.	Palabra de E/S.
RETYPE	Permite editar el contenido del buffer de entrada.	Palabra de E/S.
LINE	Trabaja con el buffer de entrada, aceptándolo como una entrada normal.	Palabra de E/S.
NUMBER	Toma un número del buffer de entrada.	Palabra de E/S.
FIND	Busca la dirección en la que se encuentra compilada la palabra introducida en el buffer de entrada.	Palabra de E/S.
WORD	Trata el contenido del buffer de entrada como un texto, almacenándolo en el PAD.	Palabra de E/S.
EXECUTE	Ejecuta la palabra cuya dirección de memoria se encuentra en la pila.	Manejo de memoria.
PAD	Da la dirección de comienzo del PAD.	Manejo de memoria.
INKEY	Lee el teclado y toma el código ASCII del carácter introducido, depositándolo en la cima de la pila.	Palabra de E/S.



Representación gráfica de la estructura del PAD.

ordenador para indicar hasta dónde debe leer del buffer de entrada.
3. Lee del buffer de entrada: obvia los posibles delimitadores que se encuentren al principio del buffer, y lee la palabra hasta el siguiente limitador.
4. Copia en el PAD la palabra leída; coloca al principio la longitud de la misma, a continuación los códigos ASCII de cada uno de los caracteres, y al final el código ASCII de un delimitador (o un cero si no encontró ningún limitador antes del final del buffer).
5. Coloca en la cima de la pila la dirección de comienzo del PAD.

LECTURA DIRECTA DEL TECLADO

El lenguaje FORTH cuenta con una palabra capaz de leer cualquier pulsación que se efectúe sobre el teclado; se trata de la palabra INKEY. Esta identifica la tecla accionada y deposita su correspondiente código ASCII a la cima de la pila.
Un pequeño inconveniente de esta palabra es que no aguarda a que se pulse una tecla para continuar el proceso, sino que lee del teclado y si no se ha pulsado ninguna tecla continúa el proceso de inmediato. Para solventarlo, es conveniente utilizar la palabra INKEY dentro de un bucle como el siguiente:

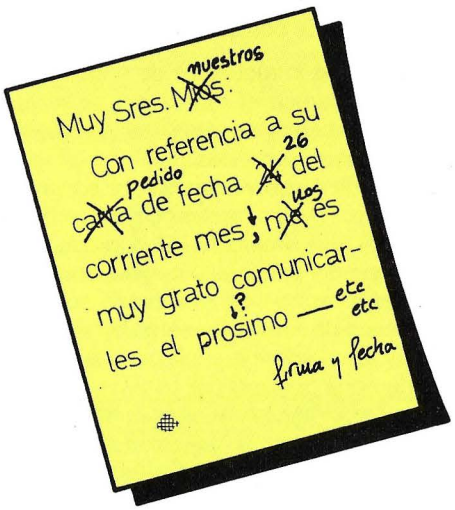
```
: BUCLE
BEGIN
INKEY
UNTIL
;
```

Cuando se ejecute la palabra BUCLE, el ordenador aguardará hasta que se pulse una tecla para continuar el proceso.
Al concluir la ejecución, quedará en la cima de la pila el código ASCII del carácter pulsado, listo para su posterior tratamiento.

OASIS (y 4)

El vocabulario de comandos

Para desarrollar su trabajo sobre el ordenador, cualquier sistema operativo se apoya en la comunicación entre los periféricos y la CPU, comunicación que el sistema gestiona respondiendo a los dictados del usuario. Estos dictados se confeccionan explotando los comandos disponibles, que a través de sus diversas opciones, permiten definir con exactitud la tarea a realizar.



El comando TEXTEDIT aporta el método básico para la creación y corrección de ficheros de texto.

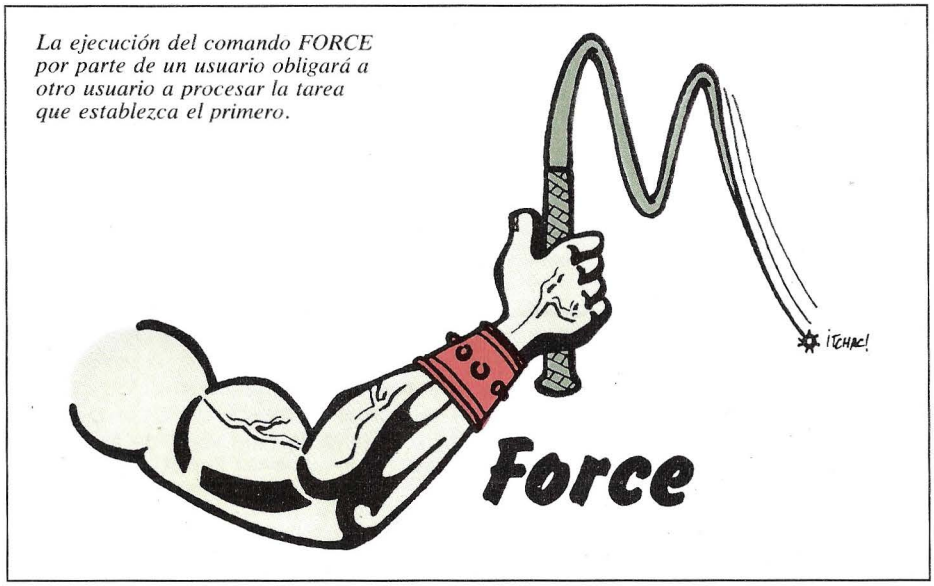
El sistema operativo OASIS, como ya se ha indicado en capítulos anteriores, está esencialmente orientado a usuarios con escasa formación informática; sus comandos se designan empleando, siempre que es posible, palabras del vocabulario inglés convencional. Ello convierte al OASIS en un sistema operativo más "cómodo" para el usuario que otros de similar categoría.

Los comandos del OASIS pueden agruparse en distintas categorías, teniendo en cuenta su función:

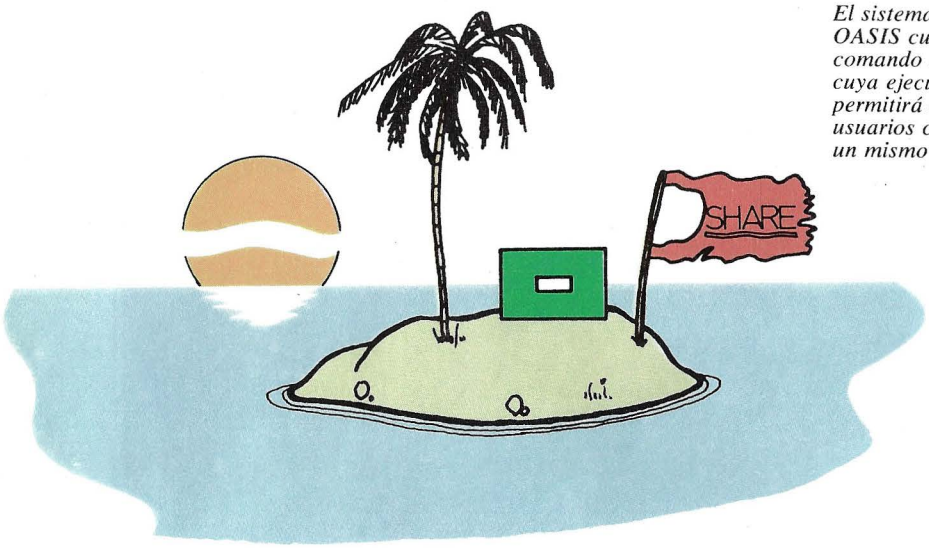
— Comandos relacionados con la gestión y ejecución de programas.
— Comandos destinados al control de los parámetros del sistema.
— Comandos destinados al control de las comunicaciones.
— Comandos relacionados con la gestión de ficheros.
— Comandos relacionados con la gestión y el control de periféricos de entrada/salida.

Dentro de cada una de estas categorías caben diversos comandos. Dado su elevado número, no vamos a entrar en detalles respecto a todos ellos, sino que tan sólo se dará una idea de aquellos que resultan más representativos.

El sistema operativo OASIS contempla la necesidad de que los diversos usuarios conectados al ordenador compartan con eficacia los recursos del sistema; por ejemplo, la impresora.



La ejecución del comando FORCE por parte de un usuario obligará a otro usuario a procesar la tarea que establezca el primero.



El sistema operativo OASIS cuenta con el comando SHARE, cuya ejecución permitirá que varios usuarios compartan un mismo fichero.

COMANDOS PARA LA GESTION Y EJECUCION DE PROGRAMAS

Dentro de los comandos relacionados con la gestión de programas, tal vez el más representativo sea el editor del sistema. A través del mismo tiene lugar la creación y el mantenimiento de ficheros que serán empleados por programas tales como el procesador de lenguaje EXEC, el MACRO-ensamblador o el intérprete/compilador BASIC. Además, también puede ser utilizado para generar y mantener ficheros destinados a operar con los programas del usuario, tales como ficheros de datos.

El editor es invocado por medio del comando TEXTEDIT, el cual abre al usuario el acceso a un repertorio de comandos de edición. Estos, en su formato y nomenclatura, son congruentes con los comandos generales del sistema. El referido editor es del tipo "de líneas"; ello significa que actúa sobre líneas completas a la hora de ejecutar sus funciones, y no sobre conjunto de líneas como haría un editor de pantalla completa.

El formato del comando que da entrada al editor es el siguiente:

TEXTEDIT (Nombre del fichero)

En estas condiciones, el OASIS inicia la búsqueda del fichero especificado. Si en el nombre del fichero no se incluye una referencia al disco en el que se encuentra, el editor realizará una secuencia de búsqueda a través de las diferentes unidades de disco asociadas al ordenador, de acuerdo con lo indicado en otros capítulos de esta obra. Una vez localizado el fichero, éste es cargado en memoria. Por medio de los mensajes oportunos, el sistema comunicará si el fichero ha sido cargado correctamente o si, por el contrario, el tamaño del fichero es superior al de la partición de memoria a disposición del usuario.

El "prompt" o indicativo de que el editor está en activo es, en el sistema operativo OASIS y al igual que en el sistema MS/DOS, un asterisco (*). El asterisco, sin embargo, puede desaparecer, toda vez que el editor del sistema operativo OASIS puede operar en dos modos distintos:

- *Modo comando* en el que el prompt se encuentra presente y en el que se introducen los diferentes comandos de actuación sobre las líneas del fichero, como por ejemplo el de borrado.

- *Modo de introducción de texto,* en el cual no aparece el asterisco y se realiza todo el proceso de modificación interna de las diferentes líneas.

El final de la operación con el editor se ordena por medio de dos comandos: FILE o QUIT. El primero de ellos produce la actualización del fichero original, toda vez que éste no ha sido modificado sobre el periférico de almacenamiento masivo sino únicamente sobre la copia almacenada en la memoria principal del sistema. En todo caso, la versión antigua del fichero no se pierde; el sistema operativo OASIS, al igual que hacen otros sistemas operativos como el MS/DOS, almacena esta versión con el mismo nombre y con el tipo de fichero BACKUP, eliminando durante el proceso cualquier otro fichero con esta denominación. El segundo comando de salida, QUIT, se utilizará cuando el usuario decida abandonar una sesión de trabajo sin almacenar en memoria los datos o modificaciones introducidas; en tal caso, el control se devuelve al entorno en el que se invocó el comando TEXTEDIT.

Un comando que se encuentra íntimamente relacionado con la ejecución de programas es el denominado FORCE. Sólo puede utilizarse cuando el OASIS opera en modo multiusuario, y su misión es la de "forzar" a otro usuario a ejecutar

una determinada tarea. El formato de esta orden es el que se indica a continuación:

FORCE (Id. participación) (Comando)

El identificador de participación señala cuál es el número de la participación del usuario o el nombre de la "cuenta" que va a ser forzada a realizar una tarea, mientras que el argumento comando define precisamente la tarea a ejecutar por el otro usuario. FORCE es un comando sumamente práctico, puesto que brinda la posibilidad de acceder a otros usuarios desde el terminal propio, permitiendo, en el caso de que no sea posible efectuar una operación desde el propio terminal, que ésta se realice sobre un terminal ajeno. Aunque, desde luego, su uso es peligroso si no se avisa previamente al otro usuario, ya que puede originar pérdidas de información por implicar una reconfiguración de la memoria.

COMANDOS PARA LA GESTION DE LOS PARAMETROS DEL SISTEMA

Dentro de este grupo caben distintos comandos cuya influencia en la gestión del sistema es variable. Veamos algunos de los más importantes.

El comando SHARE tiene la función de señalar al sistema operativo que un determinado fichero va a ser compartido por varios usuarios. Su formato es el siguiente:

SHARE (Nombre del fichero) (Cuentas)

El nombre del fichero corresponde al que va a ser compartido entre varias "cuentas" o usuarios definidos en el sistema. El parámetro cuentas, si es indicado, presenta una lista de las diversas cuentas que tienen acceso al referido fichero. De omitir tal opción, permitirá que las cuentas que tienen acceso al fichero sean modificadas.

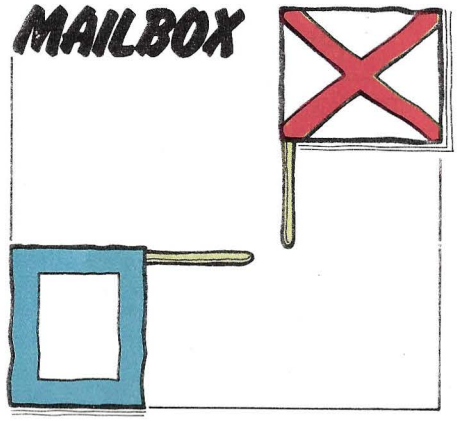
El comando ACCOUNT tiene por misión controlar el fichero SYSTEM ACCOUNT, el cual lleva el control de la "contabilidad interna del sistema". El referido fichero memoriza el nombre de las cuentas, sus sinónimos, las palabras clave de acceso y los diferentes niveles de privilegio. El formato del comando ACCOUNT es el que sigue:

ACCOUNT (Opciones)

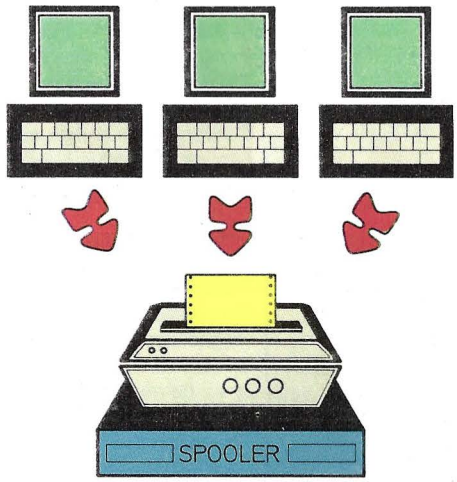
Las distintas opciones que pueden especificarse permiten eliminar usuarios o acceder al fichero histórico en el cual se encuentra almacenada la contabilidad del sistema.

COMANDOS DESTINADOS AL CONTROL DE LAS COMUNICACIONES

Dentro de este grupo existen varios comandos entre los que cabe citar, por ejemplo, a MAILBOX. Este permite el envío de mensajes entre los diferentes usuarios; desde luego, ello revela que tan sólo se encuentra disponible en la versión multiusuario del OASIS. Una vez invocado el comando y enviados los mensajes, el sistema operativo mantiene un control de los mismos de tal forma que éstos no se repitan. El sistema operativo garantiza también que cada usuario sólo reciba aquellos mensajes específicamente destinados al mismo.



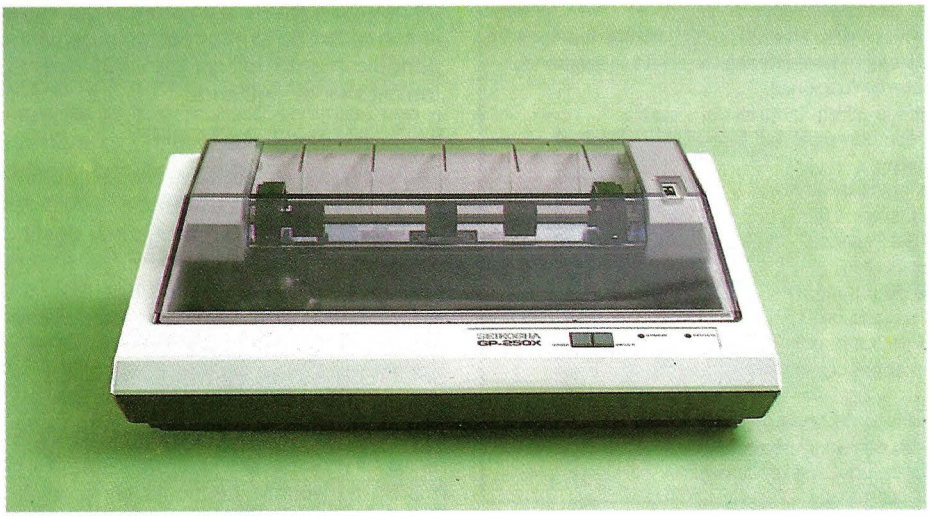
En el entorno multiusuario propio del OASIS, los diversos usuarios pueden intercambiar mensajes entre sí recurriendo al comando MAILBOX.



Para lograr que varios usuarios puedan compartir una misma impresora es preciso hacer uso del "spool" de impresión; el control de ésta herramienta corre a cargo del comando SPOOLER.

COMANDOS RELACIONADOS CON LA GESTION DE FICHEROS

El sistema operativo OASIS ofrece una amplia variedad de comandos destinados a la gestión de ficheros; comandos que



La vía de diálogo de cada usuario la aporta el terminal: teclado más pantalla. A través de este periférico de entrada/salida, el operador explota las aplicaciones y accede a los recursos del sistema bajo el control del OASIS.

PRINCIPALES COMANDOS DEL OASIS

COMANDO	FUNCION
ACCOUNT	Mantenimiento de los nombres y atributos de las "cuentas" de usuario.
ASSIGN	Asignación de canales de E/S.
ATTACH	Conexión lógica de un dispositivo para el futuro acceso al mismo.
BASIC	Creación o actualización de programas fuente en BASIC (opcional).
BISYNC	Emulación de las funciones de un terminal IBM 2780/3780.
CHANGE	Cambio del nivel de privilegio de un programa.
COPYFILE	Copia de archivos en disco.
CREATE	Creación de nuevos ficheros con formato directo o indexado.
DEBUG	Puesta a punto dinámica de programas en lenguaje máquina (opcional).
DUMPDISK	Visualización del contenido físico de un archivo en hexadecimal y ASCII (opcional)
EDIT	Creación y actualización de programas fuente (opcional).
ERASE	Borrado de archivos.
EXEC	Ejecución de programas confeccionados en lenguaje EXEC.
FILELIST	Listado de los nombres y atributos de los archivos de un directorio.
FILT8080	Traslado de nemotécnicos de ensamblador INTEL a nemotécnicos Zilog, compatibles con el Macroensamblador del OASIS (opcional)
| FORCE	Imperativo para que otra partición de usuario ejecute un comando.
GETFILE	Transferencia de un archivo secuencial desde un disco no compatible con OASIS (opcional).
INTELHEX	Conversión de código objeto INTEL a código objeto OASIS (opcional).
KILL	Elimina un archivo del directorio sin reasignar su espacio (opcional).
LINK	Traslado con edición de un fichero de programa objeto a un archivo imagen (opcional).
LIST	Listado del contenido de un archivo en la pantalla o impresora.
| LOAD	Carga de un comando no residente para su posterior uso.
MACRO	Traslado de un programa fuente en ensamblador a programa objeto (opcional).
| MAILBOX	Recuperación de un mensaje enviado por otro usuario.
MEMTEST	Diagnóstico de memoria (opcional).
MOUNT	Autorización para cambio de disco.
| MSG	Envío de mensajes a otros usuarios.
OWNERCHG	Transferencia de la propiedad de archivos a otros usuarios.
PATCH	Corrección de anomalías en un programa de tipo comando.
PEEK	Acceso a la salida por terminal de otros usuarios.
RECEIVE	Recogida y grabación de un archivo transferido desde otro sistema (opcional).
RELOCATE	Creación de un archivo objeto reubicable a partir de dos archivos objeto (opcional).
RENAME	Cambio del nombre, tipo o entrada de protección de un archivo.
RUN	Ejecución de programas BASIC compilados (opcional).
SECTOR	Visualización de los números de sector utilizados por un archivo (opcional).
SEEK	Comprobación por la lectura de sectores aleatorios de un disco (opcional).
SEND	Enviar un fichero hacia otro sistema (opcional).
SET	Autorización/Inhibición de diversos conmutadores del sistema.
SHARE	Autorización/Inhibición del acceso compartido a un fichero por parte de otras "cuentas" de usuario.
SHOW	Visualización del estado de varios conmutadores y parámetros del sistema.
SORT	Clasificación de un archivo en disco en la secuencia especificada (opcional).
SPOOLER	Cambio o examen del estado del "spooler" de impresora.
| START	Inicialización de una partida de usuario.
STATE	Comprobación de la existencia de un archivo.
| STOP	Anulación de una partición de usuario.
SYSGEN	Grabación en disco del estado de los parámetros del OASIS.
TERMINAL	Imperativo para que el sistema actúe como terminal de otro sistema externo (opcional)
TEXTEDIT	Creación o actualización de archivos de texto.
| UNLOAD	Eliminación de la memoria de un comando no residente.
VERIFY	Comprobación del buen estado de un disco (opcional).

La tabla enumera una amplia selección de comandos propios del OASIS. La relación no es exhaustiva, sino que corresponde a una síntesis en la que figuran los comandos más característicos de éste sistema operativo.

permiten cambiar el nombre a un fichero, eliminarlo de la memoria, crearlo o copiar un fichero en otros. No vamos a profundizar en los habituales, sino que tan sólo citaremos a uno de ellos, peculiar y propio del OASIS. Se trata del comando MOUNT, cuya función es la de informar al sistema que un disco va a ser cambiado. Su formato es:

MOUNT (Ident.) |*

La zona "ident" corresponde a la etiqueta del volumen o soporte que va a ser cambiado, mientras que el asteristo (*) indica que todos los discos asociados pueden ser objeto de cambio. De producirse un cambio de disco durante un proceso de actualización, sin que el usuario haya invocado a este comando, se obtendrá un mensaje de error.

COMANDOS PARA EL CONTROL DE PERIFERICOS DE ENTRADA/ SALIDA

Básicamente se orientan en dos direcciones: la que contempla la gestión de los diferentes discos asociados al sistema, y la relativa a la gestión de la impresora.
Dentro del primer grupo se encuentra el comando BACKUP, cuya misión es la de copiar el contenido completo de un disco a otro o bien a una cinta magnética. Tal operación conviene realizarla a intervalos de tiempo regulares, en orden a garantizar la seguridad de la información obteniendo periódicamente copias de seguridad. Su formato es el siguiente:

BACKUP (d1 d2) (NOVERIFY)

d1 y d2 son, respectivamente, las etiquetas de los directorios (discos o volúmenes) correspondientes al origen y destino de la información a copiar. El parámetro NOVERIFY, en el caso de ser incluido, señala que la copia no debe ser comprobada.
SPOOLER es un comando destinado al control del trabajo con impresora; este puede ser utilizado siempre y cuando el hardware lo soporte. Por medio del mismo es posible gestionar las colas de impresión que se generan en el sistema al enviar los diferentes usuarios sus listados hacia la impresora.

Easy Writer (2)

Comandos para la gestión de ficheros

En el capítulo precedente se detallaron las principales características del procesador de textos EASY WRITER, al tiempo que se estudió el sistema de ayuda que ofrece al usuario para facilitar su manejo. Se citó también al menú "Easy Writer File System", utilizado para gestionar los documentos almacenados en una unidad de memoria externa. En el presente capítulo profundizaremos en los comandos para la gestión de ficheros.

EASY WRITER FILE SYSTEM

```
                        EasyWriter File System
-------------------------------------------------------------------------------
A - APPEND A FILE   L - LINK FILES        T - DISPLAY LINKS      1 - SELECT DRIVE A
C - CLEAR SESSION   M - DISPLAY CATALOG   U - UNPROTECT A FILE   2 - SELECT DRIVE B
D - DELETE A FILE   P - PROTECT A FILE    X - EXIT TO DOS        F2 - PRINT FILE(S)
E - EDIT A FILE     R - REVISE A FILE                            F4 - ADDN COMMANDS
G - GET A FILE      S - SAVE A FILE     / - SLOW/SPEED PRINT   F7 - STOP PRINT
-------------------------------------------------------------------------------
   PRUEBA     1408  1- 1-80

FILESIZE= 1200     AVAIL= 22800     DRIVE B
FILE IN MEMORY = PRUEBA          0     FILES LINKED
COMMAND:
```

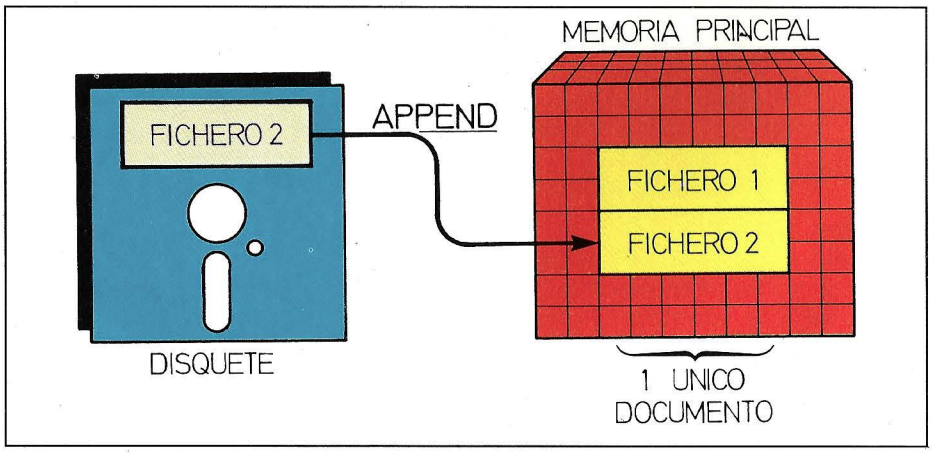
En el menú "Easy writer file system" se pueden distinguir cuatro zonas distintas; éstas son, de arriba hacia abajo: (1) zona de cabecera, (2) zona central o de catálogo, (3) zona de parámetros y (4) zona de comandos.

Cuando el usuario comienza la sesión de trabajo, inmediatamente después del menú de presentación visualizará el menú para la gestión de ficheros. En este último menú cabe distinguir cuatro zonas distintas:

1. ZONA DE CABECERA
Localizada en la parte superior de la pantalla y en la que aparece un literal con el nombre del menú, acompañado por una lista de comandos disponibles que se estudiarán más adelante.

2. ZONA CENTRAL
También denominada zona de catálogo, en la que se reflejan los documentos contenidos en la unidad de memoria auxiliar. La información relativa a cada documento incluye: el nombre del fichero en que está almacenado el documento, el espacio ocupado por el mismo (medido en caracteres) y la fecha de almacenamiento. Dado que en la zona de catálogo tan solo caben 27 nombres de documentos, en el caso de que existan más, EASY WRITER presentará el siguiente mensaje al usuario: "PRESS ENTER TO CONTINUE; OR 'ESC' FOR PROMPT"; de forma que si se pulsa la tecla [ENTER] seguirán apareciendo nuevos ficheros, mientras que si se pulsa la tecla [ESC] no se continuará.

3. ZONA DE PARAMETROS
La tercera zona del menú está reservada para que el usuario pueda conocer en todo momento la situación en que se encuentra. Los parámetros incluidos son los siguientes:



El comando APPEND permite unir dos ficheros en un único documento. Para ello, uno de los documentos debe residir previamente en la memoria principal; al ejecutar el referido comando, un segundo fichero se situará a continuación del primero.

● FILESIZE
Señala el número de caracteres utilizados por el fichero cargado en la memoria principal.

● AVAIL
Representa el número de caracteres que aún pueden añadirse en la memoria principal. Cuando el tamaño de la memoria principal es de 64 Kbytes, el contenido máximo será de 14.000 caracteres; cuando la memoria principal sea mayor, el máximo crecerá hasta 24.000 caracteres.

● DRIVE
Indica la unidad de disco que contiene la memoria auxiliar o de masa en activo

● FILE IN MEMORY
Sirve para que el usuario conozca el nombre del fichero contenido en la memoria principal.

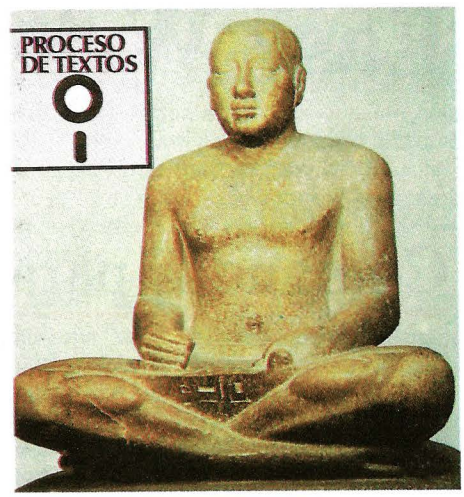
● FILES LINKED
El último de los parámetros visibles sirve para especificar los ficheros ligados, esto es, relacionados entre sí.

4 ZONA DE COMANDOS
Esta última zona del menú para la gestión de ficheros, está reservada para que el usuario seleccione el comando que desea ejecutar. En el caso de que para ejecutar dicho comando el EASY WRITER necesite algún dato relacionado con las operaciones ordenadas, utilizará también esta zona para solicitarlo al usuario.

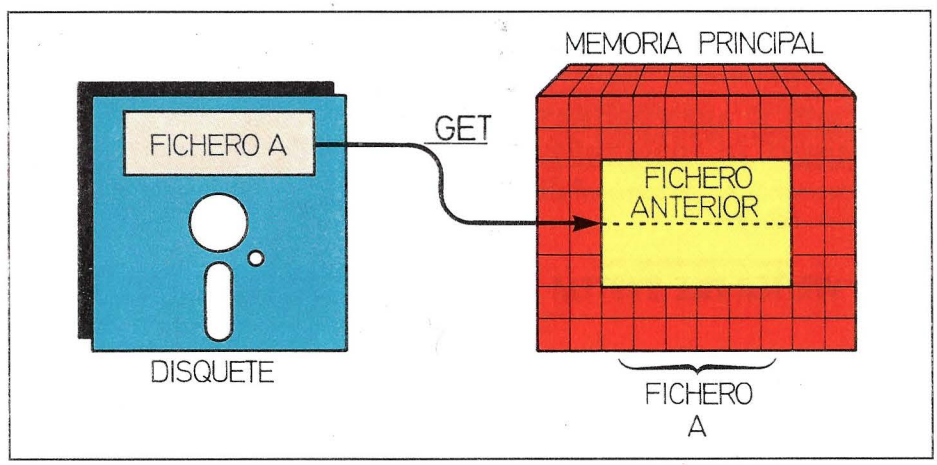
COMANDOS PARA LA GESTION DE FICHEROS

Dentro del menú "Easy Writer File System" aparece un total de trece comandos:

(A) APPEND A FILE
Su misión es situar un fichero a continuación de otro: la primera línea del segundo fichero se situará inmediatamente después de la última línea del primero.
Todas las operaciones se realizan en la memoria principal aunque a continuación puede almacenarse el documento producido por concatenación de otros. Para ejecutar este comando debe encontrarse un documento cargado en la memoria; el usuario se limitará a pulsar la tecla [A], inmediatamente en la zona de comandos. Acto seguido, EASY WRITER escribirá el nombre del comando, es decir APPEND, y solicitará el nombre del fichero que se quiere concatenar con el residente en la memoria principal. Realizadas estas operaciones, el programa "limpiará" la zona de comandos para que el usuario solicite otro, teniendo en cuenta que en la memoria principal han quedado ya agrupados los dos documentos solicitados.



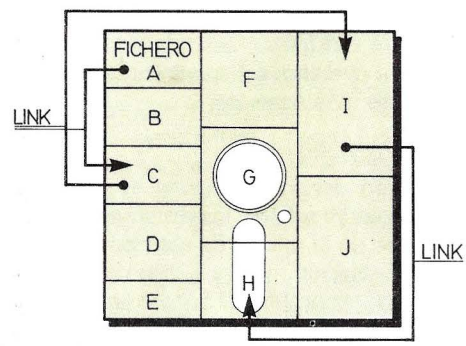
(C) CLEAR SESION
El comando asociado a la letra [C] puede utilizarse para "borrar" una sesión de trabajo; su ejecución es equivalente a la frase "empecemos de nuevo". Cuando el usuario pulsa la tecla [C] el programa EASY WRITER, antes de ejecutar el comando, solicita su confirmación mediante el literal: "ARE YOU SURE?", al que se puede contestar N (NO) para suspender la ejecución o Y (YES) para seguir adelante y, en consecuencia, borrar toda la información residente en la memoria principal.



El comando GET tiene como misión trasladar ficheros de la memoria auxiliar a la memoria principal; desde luego, el fichero que anteriormente estuviera cargado en la memoria principal se perderá, ocupando su espacio el nuevo fichero cargado.

(D) DELETE A FILE
Como su propio nombre indica, el comando DELETE sirve para eliminar un fichero de la memoria auxiliar. Para ejecutarlo, el usuario se limitará a pulsar la tecla [D] y contestar a la pregunta: "FILENAME: ..." con el nombre del fichero que desea borrar. A continuación, el programa solicitará la oportuna confirmación para borrar definitivamente el fichero.
En el caso de que el nombre del fichero no coincida con ninguno de los existentes en el disco, aparecerá un mensaje notificándoselo al usuario; "FILE nombre NOT THERE, PRESS ENTER TO RETURN".

(E) EDIT A FILE
El comando EDIT puede utilizarse para

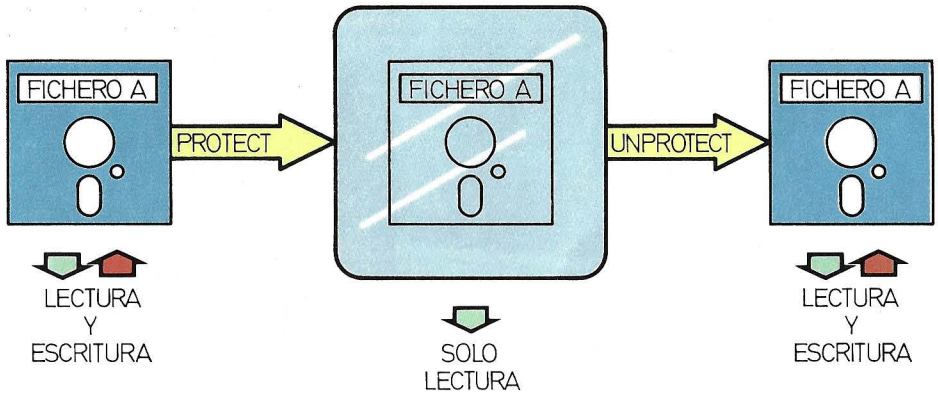


Mediante el comando LINK se pueden encardenar varios ficheros dentro de la memoria auxiliar. A partir de ese momento y para cierto tipo de operaciones, se puede considerar que los ficheros "encadenados" forman un único documento.

dos finalidades distintas; crear un nuevo documento o modificar el que se encuentra en la memoria principal. Cuando el usuario pulsa la tecla [E], el programa mostrará el siguiente mensaje: "PLEASE STAND BY". Si previamente no existía un documento en la memoria principal, aparecerá una pantalla en blanco donde el usuario podrá escribir el texto que desee; en caso contrario, es decir si ya existía un documento cargado, el cursor aparecerá en la primera línea del documento en cuestión.

(G) GET A FILE

Al pulsar la tecla [G], el programa entiende que debe localizar un fichero en la



Mediante el comando PROTECT se puede conseguir que un fichero quede protegido en una "urna de cristal": será posible leer su contenido, pero no escribir en el mismo. Para desproteger al fichero basta con ejecutar el comando UNPROTECT.

TASWORD TWO, un procesador de textos para el SPECTRUM de 48 K

Aunque un mano-procesador, como el ZX-SPECTRUM, no está capacitado para operar con procesadores de textos convencionales (por ejemplo, el EASY WRITER), sí existen programas sustitutivos de relativa calidad a los que puede tener acceso. Uno de ellos es el que nos ocupa: su nombre es TASWORD (de la compañía TASMAN SOFTWARE).

Dado que el SPECTRUM sólo visualiza 32 caracteres por cada línea de pantalla y el TASWORD utiliza líneas más largas, se ofrecen dos soluciones al usuario: comprimir el tamaño de las letras para visualizar una línea completa, o hacer "scrolling" horizontal conservando el tamaño de letra original.

Para ayuda del usuario, el programa dispone de dos pantallas del HELP en las que se detallan las posibilidades ofrecidas por el TASWORD; entre ellas cabe destacar las siguientes:

- Centrado y gestión de líneas

Mediante las teclas [<=], [<>] y [>=] se puede desplazar a la izquierda, centrar, o desplazar a la derecha una línea, respectivamente.

- Inserciones

Utilizando la tecla [AND] se pueden insertar caracteres en una línea, o líneas en un texto.

- *Scrolling*

Para efectuar desplazamientos hacia abajo o hacia arriba se dispone de las teclas [TO] y [THEN].

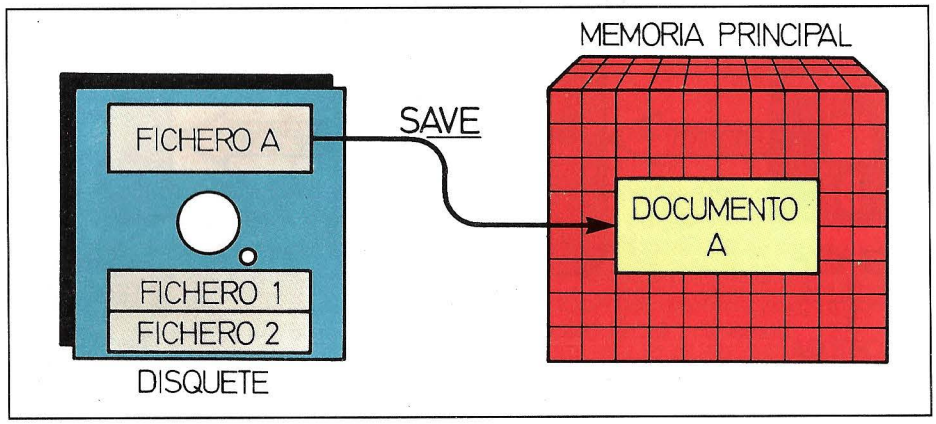
- Gestión de almacenamiento

Mediante la tecla [STOP] se puede transferir un documento del casete a la memoria, de la memoria al casete, o de la memoria hacia la impresora.

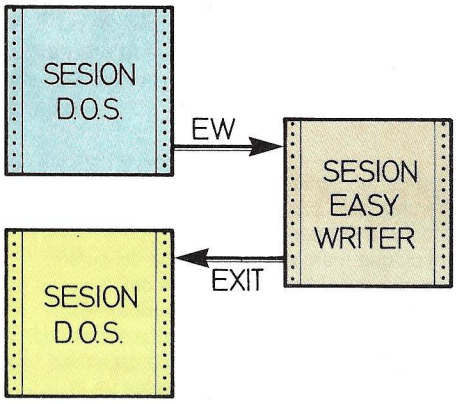
- Comandos de bloque

En este punto se cuenta con cuatro funciones, asociadas a las teclas [B], [V], [N] y [M]; las dos primeras sirven para marcar el comienzo y final de un bloque, la tercera para copiar el bloque marcado a la posición del cursor, y la última para mover el bloque marcado a la posición del cursor.

En resumen, las posibilidades del TASWORD son relativamente potentes. Aquí tan sólo hemos apuntado algunas de ellas. En todo caso, la reducida capacidad del ordenador en el que se ejecutará le impide llegar a prestar una colaboración tan importante como la que brinda un procesador de textos convencional. No obstante, sus servicios pueden ser suficientes para pequeños trabajos de proceso de textos.



El comando SAVE es complementario al comando GET: sirve para grabar un documento en la memoria auxiliar o de masa, tomándolo de la memoria principal.



Una vez que el ordenador está conectado y bajo el control del sistema operativo, basta con teclear las letras EW y pulsar RETURN para pasar al dominio de la aplicación EASY WRITER; a partir de esta situación es suficiente con ejecutar el comando EXIT para retornar al control del sistema operativo DOS.

memoria auxiliar y cargarlo en la memoria principal. Para ello solicitará al usuario un nombre de fichero; en el caso de que, en efecto, exista un documento con dicho nombre, éste se cargará en la memoria principal y el usuario visualizará el siguiente mensaje: "FILE IN MEMORY = nombre". Si el fichero no resultara localizable, EASY WRITER lo comunicará con un mensaje idéntico al ya descrito anteriormente: "FILE nombre NOT THERE, PRESS ENTER TO RETURN".

(L) LINK FILES

El objeto del comando LINK es encadenar de forma lógica varios ficheros. Es importante no confundirlo con APPEND; este último permitía concatenar documentos con fin de obtener un nuevo documento en la memoria principal; en cambio, el comando LINK produce un encadenamiento lógico sin que se unifiquen los documentos.

Tras accionar la tecla [L], el programa presenta el siguiente mensaje: "ENTER FILE NAMES SEPARATED BY COMMAS", y aguarda la introducción de una lista de nombres de ficheros. El número máximo de ficheros encadenables depende del número de caracteres que tengan como nombre; así, se podrían encadenar 124 ficheros con denominaciones de un único carácter, o tan sólo 27 si su denominación incluye ocho caracteres en el nombre.

(M) DISPLAY CATALOG

Se invoca pulsando la letra [M], y su misión consiste en mostrar por la pantalla los nombres de todos los ficheros almacenados en la memoria auxiliar. En la zona de catálogo del menú para la gestión de ficheros sólo caben 27 denominaciones; por lo tanto, cuando en el soporte de almacenamiento existan más documentos, EASY WRITER mostrará el siguiente mensaje: "PRESS ENTER TO CONTINUE, OR 'ESC' FOR PROMPT", y esperará a que el usuario indique si desea seguir visualizando o prefiere interrumpir el catálogo.

(P) PROTECT A FILE

Mediante la opción [P] del menú para la gestión de ficheros el usuario puede proteger un documento, de forma que a partir de ese momento no podrá ser borrado ni modificado. Después de ordenar la ejecución del comando PROTECT, el programa solicitará el nombre del fichero a proteger, siendo imprescindible que dicho fichero se encuentre en la memoria auxiliar y no en la principal.

(R) REVISE A FILE

EASY WRITER permite obtener versiones revisadas de un documento. Antes de pulsar la tecla [R], asociada a este comando, el usuario debe realizar un GET sobre el fichero que desee revisar y, a continuación, mediante el comando EDIT, puede incluir las modificaciones que considere oportunas. Una vez realizadas las anteriores operaciones, sin más que ejecutar el comando REVISE y tras confirmar al programa que en efecto se desea ejecutar ese comando, en la unidad auxiliar de almacenamiento se grabará un fichero con el mismo nombre del documento original pero con la versión revisada.

(S) SAVE A FILE

La utilidad del comando SAVE consiste en facilitar la transferencia o almacenamiento de un documento desde la memoria principal a la memoria auxiliar. Para ello, basta con usar la tecla [S], e indicar el nombre con el que se desea almacenar el documento.

Una vez realizada esta operación, se visualizará el mensaje: "SAVE FILE nombre", y en la zona de catálogo aparecerá el nombre del fichero recién creado.

(T) DISPLAY LINKS

Si el usuario desea conocer el nombre de los ficheros encadenados por efecto del comando LINK, debe ejecutar la opción DISPLAY LINKS, para lo cual es suficiente con pulsar la tecla [T].

(U) UNPROTECT A FILE

El comando UNPROTECT es complementario respecto al comando PROTECT. En efecto, cuando se pulsa la tecla [U] y a continuación se especifica el nombre de un fichero protegido, EASY WRITER presentará en la pantalla el siguiente mensaje: "FILE nombre UNPROTECTED, PRESS ENTER TO RETURN"; desde ese preciso instante el documento quedará desprotegido y, en consecuencia, podrá ser borrado o revisado.

(X) EXIT TO DOS

En un punto anterior se indicó que el comando CLEAR permitía abandonar el entorno de trabajo y comenzar de nuevo. El cometido del EXIT es similar, aunque a mayor nivel: una vez que el usuario pulsa la tecla [X] y confirma que no se trata de un error, se abandonará la sesión de trabajo con EASY WRITER pasando directamente al control del sistema operativo.

Archivos en BASIC (2)

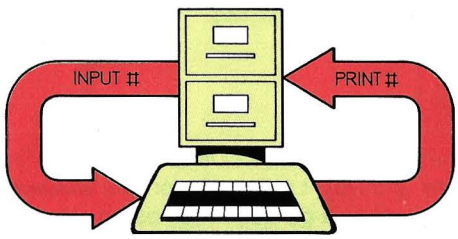
Uso eficiente de los archivos secuenciales

En un capítulo precedente se introdujo el concepto de archivo y se señalaron las principales características de esta estructura de almacenamiento. En aquella ocasión se estudiaron los comandos básicos destinados al tratamiento de archivos secuenciales. El objeto de este capítulo es profundizar en el tema, tratando de obtener un mayor partido del uso de este tipo de archivos. Para ello, se presentarán las restantes herramientas al efecto que aporta el lenguaje BASIC.

VARIABLES DE CONTROL

Antes de seguir adelante es conveniente repasar la filosofía del tratamiento y manipulación de archivos. Desde el punto de vista del programador, existen tres etapas básicas en el trabajo con archivos. Estas se identifican como: apertura, consulta y cierre. Las operaciones de apertura y cierre se efectúan por medio de los comandos OPEN y CLOSE, respectivamente.
En cuanto a la consulta, ésta puede conducir a la adición o recuperación de datos. La dirección del flujo de datos, ya sea entrada (desde el archivo) o salida (hacia el mismo), ha de ser indicada en la propia instrucción de apertura. Dicho flujo se controla empleando los comandos PRINT# (salida) e INPUT# (entrada).
Un ejemplo práctico del uso de un archivo secuencial consiste en el almacenamiento de una matriz.

```
...
100 OPEN "O",#1,"MATRIZ"
110 FOR I=1 TO 20
120 PRINT# 1,A(I)
130 NEXT I
140 CLOSE#1
...
```



El diálogo entre la unidad central del ordenador y los archivos de información se materializa mediante los comandos PRINT# e INPUT#, según se ordene una operación de escritura o lectura en el archivo.

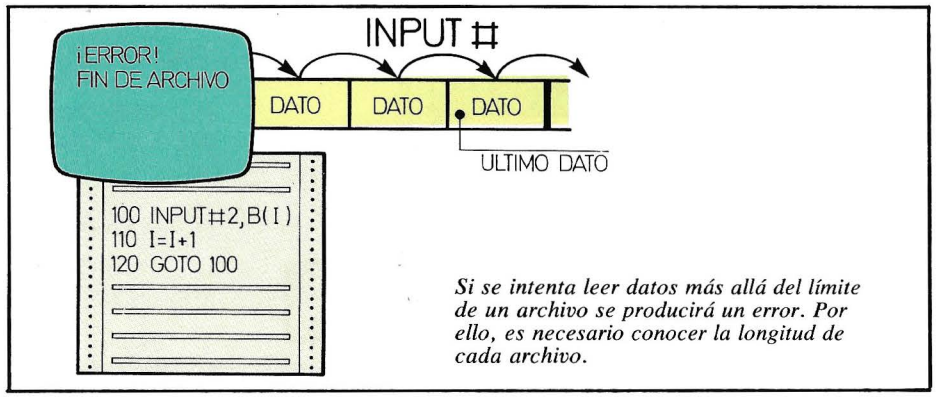
Para recuperar los datos es suficiente con formular la rutina inversa. Esto es: sustituir el comando PRINT# por INPUT# y cambiar el modo de apertura para que se autorice la entrada de datos. La rutina de lectura del archivo recién creado será, pues, la siguiente:

```
...
200 OPEN "O",#1,"MATRIZ"
210 FOR I=1 TO 20
220 PRINT# 1,B(I)
230 NEXT I
240 CLOSE#1
...
```

El proceso no reviste mayor complicación ya que, en este caso, el tamaño de la matriz es conocido. El problema aparece cuando se lee un archivo del que se desconoce la longitud. Si el archivo guarda un total de 10 datos, se producirá un error al intentar recuperar el undécimo. Ese error hará que se detenga la ejecución, con los inconvenientes que ello conlleva.
Así pues, lo más acertado es tener una indicación de la longitud del archivo que se está tratando. Esta indicación llega de la mano de un nueva función BASIC. Se trata de LOF (Length Of File = longitud del archivo) que proporciona el número de bytes de que consta un determinado archivo. LOF admite un dato de entrada en su argumento que indica el archivo del que se desea averiguar la longitud; éste ha de corresponder con el número de identificación especificado en la apertura del fichero. Véase un ejemplo.

```
...
170 OPEN "I",#1,"ARCHIV"
180 L=LOF(1)
...
```

En el ejemplo se almacena dicha longitud en la variable L. El número identificativo del archivo (en este caso el 1) es el que figura en el argumento de LOF. Tras la ejecución de estas líneas, la variable L pasa a memorizar el número de bytes de que consta el archivo "ARCHIV".
Sin embargo, esta medida no es siempre la más apropiada. También se puede cal-



Si se intenta leer datos más allá del límite de un archivo se producirá un error. Por ello, es necesario conocer la longitud de cada archivo.

LOF

Proporciona el número de bytes de que consta un determinado archivo.

Formato: LOF (<número de archivo>)
Ejemplos: 10 LET LONG=LOF(2)
50 PRINT LOF(1)/8

EOF

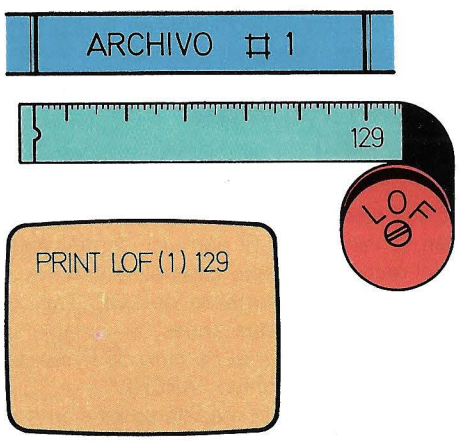
Adopta el valor lógico CIERTO cuando se alcanza el final de un archivo.

Formato: EOF (<número de archivo>)
Ejemplos: 20 IF EOF(1) THEN END

cular el número de datos contenidos en un archivo a partir del número de bytes del mismo. Pero sólo se podrá hacer de forma sencilla si todos los datos almacenados son del mismo tipo.

Existe otra función que resulta más útil para delimitar la longitud de un archivo; una función lógica que toma al valor CIERTO cuando se alcanza el fin del archivo. La palabra del BASIC asociada a la referida función es EOF (End Of File = fin de archivo). La formulación del EOF es análoga a la de LOF. Veamos un ejemplo en el que se emplea esta última función:

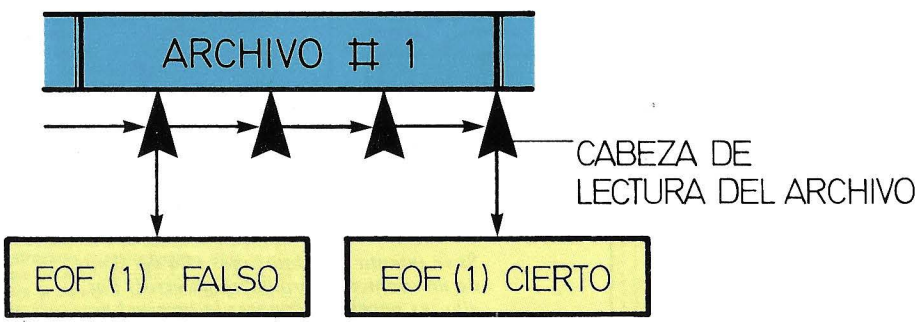
```
200 OPEN "I",#1,"DATOS"
210 I=1
220 IF EOF(1) THEN GOTO 260
230 INPUT# 1,B(I)
240 I=I+1
250 GOTO 220
260 CLOSE#1
...
■
```



La función LOF proporciona la longitud, medida en bytes, de un determinado archivo.

En la rutina se emplea un bucle, controlado por la variable I, para leer los datos del archivo. La variable de control se inicializa al valor 1 en la línea 210. Tras recuperar un dato (línea 230) se incrementa dicho contador. Esto hará que los datos leídos se almacenen en posiciones sucesivas de la matriz B.

La línea 250 cierra el bucle, permitiendo la lectura del siguiente dato; sin embargo, la clave de la rutina se encuentra en la línea 220. El comando IF contenido en la misma proporciona el punto de salida del bucle. Esta se producirá cuando EOF(1) sea cierto; esto es: cuando se haya alcanzado el fin del archivo. De esta forma se evita la lectura del siguiente dato (dato, por otra parte, inexistente).



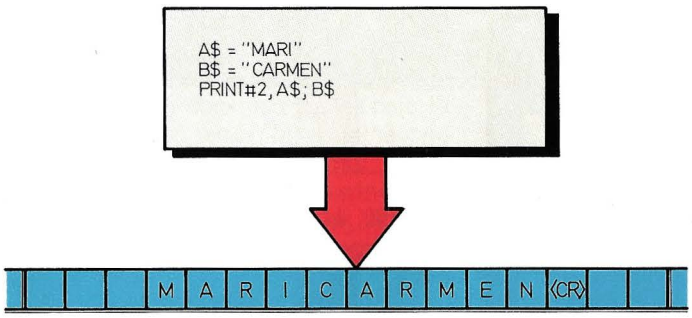
Para localizar el límite de un archivo se emplea la función EOF. Dicha función lógica adopta el valor CIERTO cuando se alcanza dicho límite.

DELIMITADORES DE DATOS

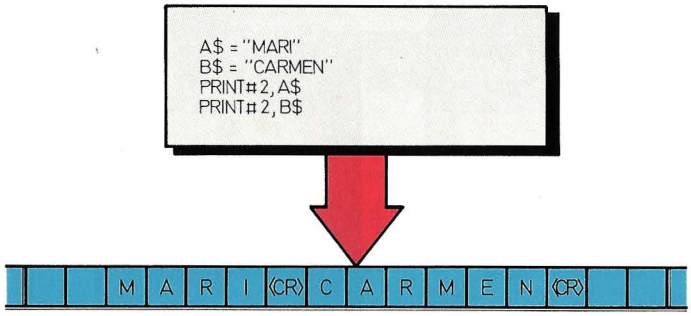
En el apartado precedente se ha descrito el método a seguir para averiguar la longitud de un archivo. Como ya se comentó en el primero de los capítulos dedicados al tema de archivos, el propio ordenador se encarga de grabar una marca de fin de archivo tras el último dato. Sin embargo, todavía no se ha mencionado cómo se separan los datos entre sí. Ello va a depender de la forma en la que se hayan almacenado.

Según lo explicado hasta ahora, la grabación de datos se efectúa por medio del comando PRINT#. Dicho comando actúa de forma análoga al encargado de la presentación de datos en pantalla (PRINT,sin "#").PRINT# seguido por un solo dato escribe dicho dato en el archivo añadiendo un carácter al final; carácter que equivale al de "retorno de carro" que se introduce al escribir en pantalla. Si se utiliza más de un dato en su argumento, éstos deben separarse por medio del signo punto y coma. Al igual que ocurre en la presentación en pantalla con PRINT, los datos se almacenan de forma contigua; sin ningún carácter entre ellos. Esto hace que no sea posible identificar los límites de cada dato. A continuación se muestra un ejemplo de lo indicado.

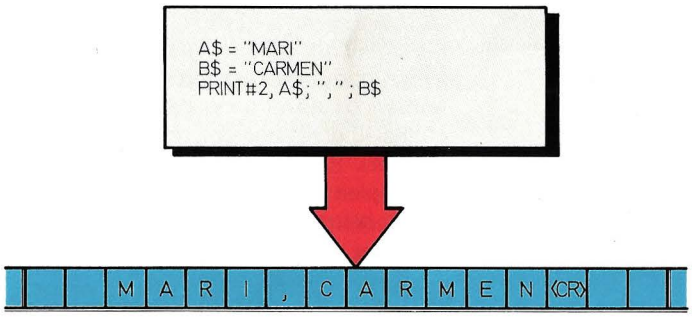
```
100 A$="MARI"
110 B$="CARMEN"
120 PRINT#2,A$;B$
■
```



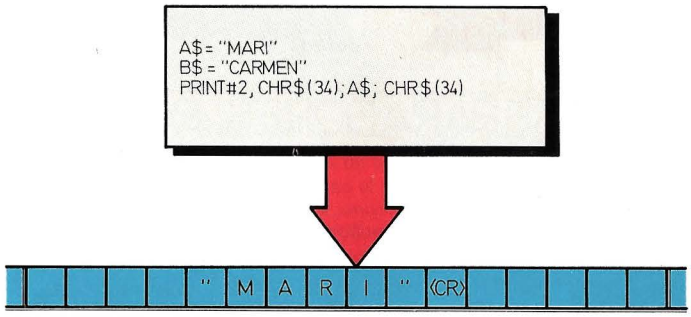
Si en el argumento de PRINT# se separan los datos con punto y coma, éstos se escribirán de forma contigua al ejecutar la referida instrucción.



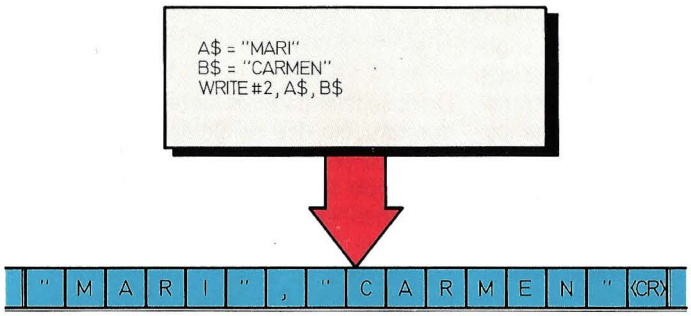
El comando PRINT# introduce al final de cada lista de datos un carácter de retorno de carro. Tal carácter permite distinguir unos datos de otros.



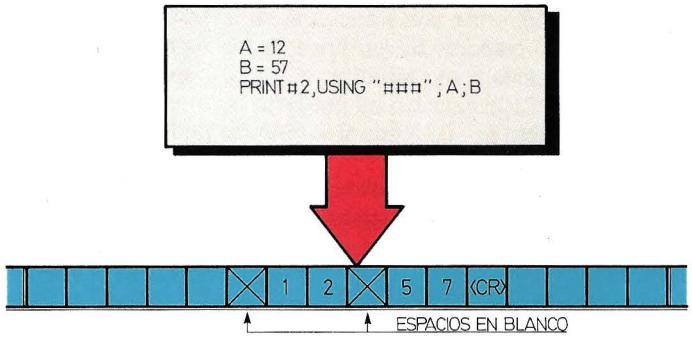
Una forma de separar datos en un archivo consiste en introducir el signo "coma" entre cada dos de ellos.



En el caso de las cadenas de caracteres, éstas se pueden delimitar introduciendo el signo de comillas antes y después de cada dato.



El empleo del comando WRITE evita la necesidad de introducir separadores de datos. El propio comando se encarga de colocar los separadores adecuados.



Otra forma de separar correctamente los datos consiste en especificar su formato mediante la opción USING del comando PRINT#.

Esta secuencia de instrucciones almacenaría en el archivo número 2 lo siguiente:

MARICARMEN

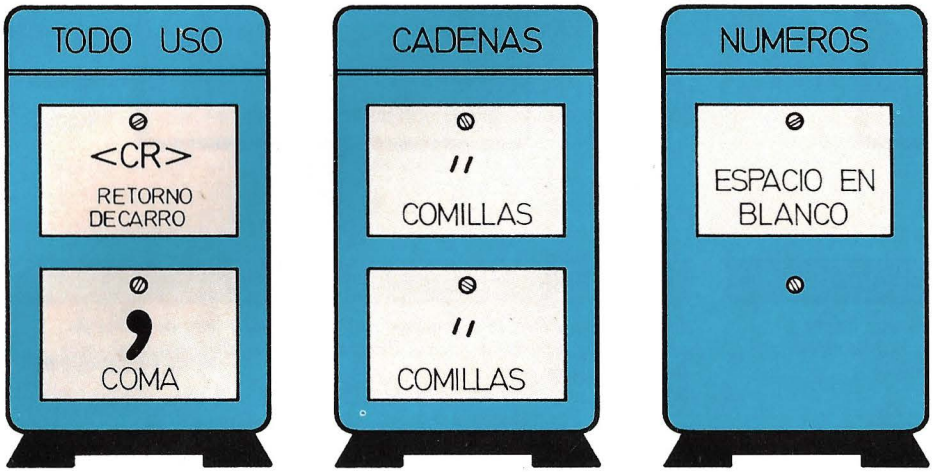
Si, posteriormente, se extraen los datos de ese mismo archivo, ambos datos se procesarían como si de uno solo se tratase. Para distinguirlos es necesario introducir un separador entre ellos.
Empleando un comando PRINT# por cada dato se incluye automáticamente el carácter de retorno de carro como separador. Incorporando tal propiedad al último ejemplo, éste quedaría como sigue:

```
100 A$="MARI"
110 B$="CARMEN"
120 PRINT#2,A$
130 PRINT#2,B$
■
```

Cuyo resultado en el archivo mostraría el siguiente aspecto:

MARI§CARMEN

Con el símbolo § se ha querido representar el carácter de retorno de carro.
Así pues, el primer delimitador o separador de datos es el carácter de retorno de carro. Este, como se ha visto, se introduce automáticamente al final de cada lista de datos.
En todo caso, el carácter de retorno de carro no es el único separador permitido. También puede utilizarse como separador el carácter "coma". De esta forma será posible incluir varios datos en el argumento de PRINT#:



SEPARADORES PARA ARCHIVOS

A la hora de almacenar datos en un archivo entran en juego varios tipos de separadores. Entre ellos están el retorno de carro y la coma. En el caso de trabajar con datos numéricos se considera el espacio en blanco como un separador más; mientras que al operar con cadenas hay que considerar el uso de comillas.

```
100 A$="MARI"
110 B$="CARMEN"
120 PRINT# 2,A$;",";B$
```

En esta ocasión, entre uno y otro dato se almacenará el carácter coma. En el archivo ello quedaría como sigue:

MARI,CARMEN

Al ser la coma un separador válido, ambos datos estarán perfectamente delimitados. En consecuencia, la posterior ejecución de la instrucción:

```
200 INPUT# 2,T$,P$
```

asignaría a las variables T$ y P$ los valores MARI y CARMEN respectivamente. Pero esto no es todo. Cualquier coma será leída como delimitador de datos. Incluso las comas que hayan sido introducidas como parte de un dato. Así, el dato alfanumérico "PEREZ,JOSE" será leído de un archivo como dos cadenas distintas: "PEREZ" y "JOSE".
Para que sea posible incluir el carácter coma en el interior de una cadena alfanumérica se hace necesario delimitar ésta de otro modo. Puede recurrirse a un nuevo carácter separador: las comillas.
Al ejecutar un comando INPUT#, si éste encuentra como primer carácter a las comillas, tomará como límite del dato las siguientes comillas. Supóngase, por ejemplo, que en el archivo se encuentra almacenada la siguiente serie de caracteres:

"PEREZ,JOSE"

Si se ejecuta, a continuación, el comando INPUT#1,T$, la variable alfanumérica T$ adoptará el valor PEREZ,JOSE.
Resumiendo, existen tres caracteres que sirven como delimitadores de datos alfanuméricos en el seno de un archivo secuencial: el retorno de carro, la coma y las comillas. En el caso de datos numéricos se emplea un cuarto delimitador, el espacio en blanco.

EMPLEO DE LOS DELIMITADORES

De los delimitadores anteriormente citados tan sólo dos se almacenan automáticamente en el archivo: el retorno de carro, al final de una lista de datos, y el espacio en blanco, entre datos numéricos de poca longitud.
Se ha visto ya la forma de intercalar un carácter coma. Para la grabación del carácter comillas la cosa se complica un poco más. Lo mejor en este caso es utilizar el código ASCII del referido carácter. Con ello, la grabación del dato asociado al último ejemplo se efectuaría con la siguiente línea de programa:

```
350 PRINT#1,CHR$(34);"PEREZ,JOSE";
    CHR$(34) .....
```

En efecto, se ha utilizado la función CHR$ para obtener el carácter comillas, carácter cuyo código ASCII suele coincidir con el número 34.
Esta es una forma de intercalar delimitadores entre los datos a almacenar, aunque, no cabe duda que el método resulta ligeramente engorroso.
En algunos dialectos BASIC se dispone de un comando que inserta automática-

WRITE#

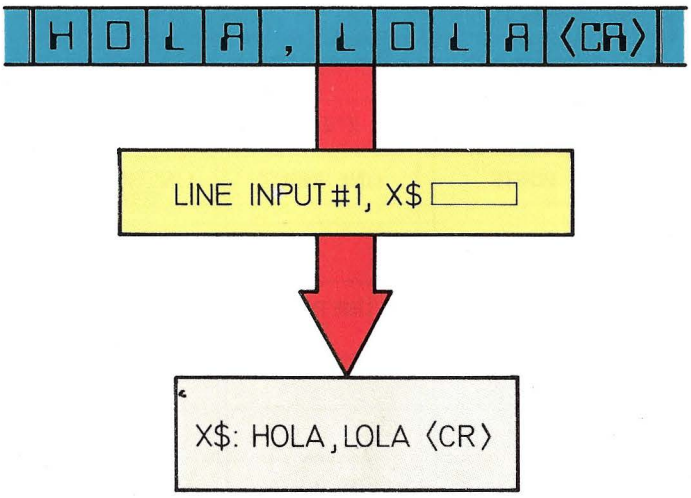
Escribe datos en un archivo introduciendo los separadores adecuados entre ellos.

Formato: WRITE #<número>,<dato>[;<dato> ...]
Ejemplos: 20 WRITE #1,"PERICO"
70 WRITE #2, A+B

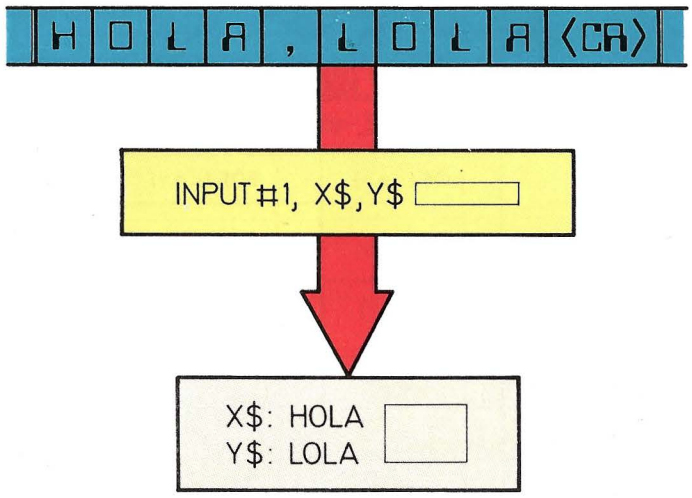
WRITE

Escribe datos en pantalla separándolos por comas; las cadenas se representan entre comillas.

Formato: WRITE <dato1> [,<dato2>...]
Ejemplos: 20 WRITE B,A$
70 WRITE "A=",352



El comando INPUT# toma como punto final de un dato a cualquier carácter separador válido



El comando LINE INPUT# sólo admite un separador de datos: el carácter de retorno de carro. Por lo demás, incluye dicho carácter en la cadena leída.

mente los delimitadores adecuados. El mencionado comando se asocia a la palabra WRITE#, cuya formulación es similar a la del comando PRINT#.

(Número de línea) WRITE# <número>,
<dato1>[,<dato2>.....]

El comando WRITE# inserta comas entre los datos que se van escribiendo en el archivo. Los datos de tipo alfanumérico se graban encerrados entre comillas. Además, WRITE# proporciona un carácter de retorno de carro al final de cada lista de datos. En definitiva, este comando deja los datos perfectamente delimitados en el archivo.
Al igual que PRINT#,WRITE# tiene un comando homólogo para la representación en pantalla. Se trata de WRITE (sin el signo # al final). WRITE actúa en pantalla de la misma forma que WRITE# en un archivo. Véase un ejemplo:

```
PRINT "DOS Y DOS SON ", 2+2
DOS Y DOS SON 4
WRITE "DOS Y DOS SON ", 2+2
"DOS Y DOS SON ",4
■
```

En él se pone de manifiesto la clara diferencia que existe entre PRINT y WRITE.

Para la escritura en un archivo secuencial también se puede hacer uso de PRINT# con la opción USING. Esta opción permite definir el formato con el que van a ser grabados los datos. Su funcionamiento es totalmente análogo al de PRINT USING, al que ya se ha dedicado un capítulo en esta obra.
Por lo que respecta a la lectura de los datos contenidos en el archivo, es preciso indicar el modo en el que ésta se efectúa. El comando utilizado hasta ahora se formula a partir de la palabra clave INPUT#. Ya se ha indicado que este comando distingue varios delimitadores de datos:

— Datos numéricos:
  Espacio en blanco
  Coma
  Retorno de carro
— Datos alfanuméricos:
  Coma
  Retorno de carro
  Comillas

En el caso de las comillas, se toma como un solo dato todo lo encerrado entre dos de esos caracteres.
Estos caracteres son precisamente los que emplea el comando INPUT# para identificar el final de un dato. De esta forma, el binomio WRITE#/INPUT# permite un fácil y eficiente manejo de los archivos secuenciales.
El BASIC suele ofrecer aún otro comando dedicado a la recuperación de datos de un archivo secuencial; comando que admite como único delimitador el carácter de retorno de carro. Para su puesta en práctica se emplean las palabras LINE INPUT#, de acuerdo al formato que se expone a continuación.

(Número de línea) LINE INPUT#
<número>,<variable cadena>

La ejecución de este comando asigna un valor a la variable de cadena especificada; valor que corresponderá a la serie de caracteres que se lean del archivo. En dicha serie de caracteres se toma el retorno de carro como único delimitador. Es más, el propio carácter de retorno de carro se considerará como parte del dato.
El comando LINE INPUT# deriva del comando análogo reservado para la lectura de datos introducidos mediante el teclado: LINE INPUT (como siempre, sin #).

LINE INPUT#

Lee datos de un archivo tomando como único delimitador el carácter de retorno de carro.

Formato: LINE INPUT#<número>,<var1>[,<var2>...]
Ejemplos: 10 LINE INPUT#1,A$
50 LINE INPUT#2, R$,P$,S$

TABLA DE CONVERSION (1)						
ORDENADOR	DELIMITACION DEL ARCHIVO		ESCRITURA		LECTURA	
	LOF (<n>)	EOF (<n>)	WRITE #	WRITE	LINE INPUT	LINE INPUT #
APPLE II (APPLESOFT)	—	—	—	—	—	—
APRICOT (M-BASIC	LOF (<n>)	EOF (<n>)	WRITE #	WRITE	LINE INPUT	LINE INPUT #
ATARI	—	—	—	—	—	—
CBM 64	—	—	—	—	—	—
DRAGON	—	EOF (<n>)	—	—	LINE INPUT	—
EQUIPOS MSX	LOF (<n>)	EOF (<n>)	—	—	LINE INPUT	LINE INPUT #
HP-150	LOF (<n>)	EOF (<n>)	WRITE #	WRITE	LINE INPUT	LINE INPUT #
IBM PC	LOF (<n>)	EOF (<n>)	WRITE #	WRITE	LINE INPUT	LINE INPUT #
MPF	—	—	—	—	—	—
NCR DM-V (MS-BASIC)	LOF (<n>)	EOF (<n>)	WRITE #	WRITE	LINE INPUT	LINE INPUT #
NEW BRAIN	—	—	—	—	LINPUT	LINPUT #
ORIC	—	—	—	—	—	—
SHARP MZ-700 (MZ-BASIC)	—	—	—	—	—	—
SINCLAIR QL	—	EOF (<n>)	—	—	—	—
SPECTRAVIDEO	LOF (<n>)	EOF (<n>)	—	—	LINE INPUT	LINE INPUT #
ZX-SPECTRUM	—	—	—	—	—	—

Este último realiza la misma acción que el comentado LINE INPUT#. La única diferencia entre ambos reside en el dispositivo del que se toman los datos. Así pues, LINE INPUT es equivalente al comando INPUT, pero con la salvedad de que sólo admite variables de cadena y que toma el retorno de carro como un carácter más de la cadena (el último de la misma).

ARCHIVOS Y DISPOSITIVOS ASOCIADOS

Todo lo concerniente a los archivos secuenciales se asocia, por regla general, al dispositivo de almacenamiento más habitual: el casete. Sin embargo, se pueden crear archivos secuenciales en otros periféricos.

Los dos soportes más empleados para el almacenamiento de datos son las cintas de casete y los discos magnéticos. Al respecto, cabe añadir que el uso de archivos secuenciales almacenados en disco es idéntico al de los archivos en casete.

Aparte de los archivos de almacenamiento de datos existe otro tipo de archivos. Se trata de los archivos abiertos en otros dispositivos que no son propiamente de almacenamiento. En realidad no se trata de archivos en el sentido estricto de la palabra; sino que más bien habría que definirlos como canales de comunicación con periféricos. Para la manipulación de estos canales se emplean los mismos

LINE INPUT

Recoge una cadena del teclado incluyendo en ella el carácter de retorno de carro.

Formato: LINE INPUT <var1>[,<var2>...]
Ejemplos: 20 LINE INPUT CA$

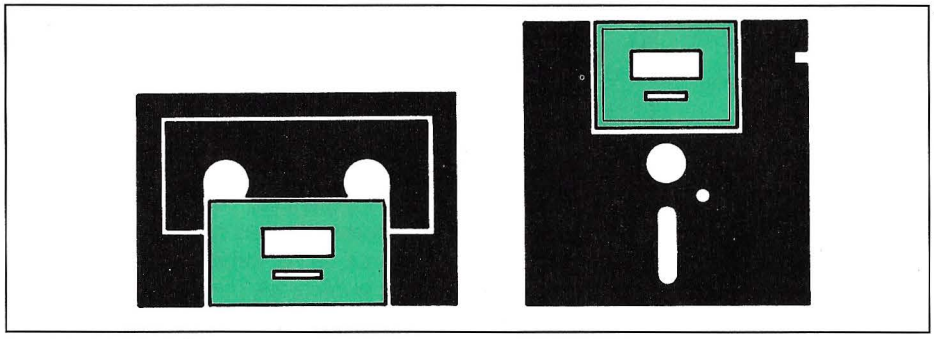
TABLA DE CONVERSION (2)

ORDENADOR	CARACTERES IDENTIFICADORES DE PERIFERICOS					
	CASETE	DISCO	IMPRESORA	TECLADO	PANTALLA	COMUNICACIONES
APPLE II (APPLESOFT)	—	—	—	—	—	—
APRICOT (M-BASIC)	—	—	—	—	—	—
ATARI	"C:	"D:	"P:	"K:	"S:	"R:
CBM 64 (1)	1	8	4	—	∅	—
DRAGON (2)	−1	—	−2	—	—	—
EQUIPOS MSX	"CAS:	—	"LPT:	—	"CRT: "GRP: (3)	—
HP-150	—	—	—	—	—	—
IBM PC	"CAS1:	"A: "B:	"LPT: "LPT2: "LPT3:	"KYBD:	"SCRN:	"COM1: 'COM2:
MPF	—	—	—	—	—	—
NCR DM-V (MS-BASIC)	—	—	—	—	—	—
NEW BRAIN (4)	1 2	—	8	5	∅	9
ORIC	—	—	—	—	—	—
SHARP MZ-700 (MZ-BASIC)	—	—	—	—	—	—
SINCLAIR QL	—	MDV (5)	—	CON	SCR	SER
SPECTRAVIDEO	"CAS:	1 2	"LPT:	"KYBD:	"SCRN:	—
ZX-SPECTRUM	—	—	—	—	—	—

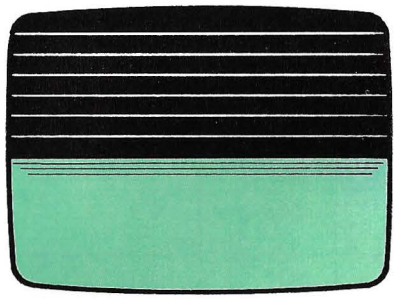
(1) El número indicado se ha de adjuntar en el segundo parámetro del comando OPEN. (2) El identificador se sitúa en la posición del número de archivo, dentro del comando OPEN. (3) CTR abre el canal en la pantalla de texto, GRP lo hace en la pantalla de gráficos. (4) El número indicado se ha de colocar como segundo parámetro en el comando OPEN. (5) En este caso el dispositivo no es el disco sino el Microdrive de Sinclair.

comandos que para el manejo de archivos secuenciales. Es por ello por lo que, a menudo, se identifican con los archivos reales.

Se pueden abrir canales para la comunicación con los siguientes periféricos:



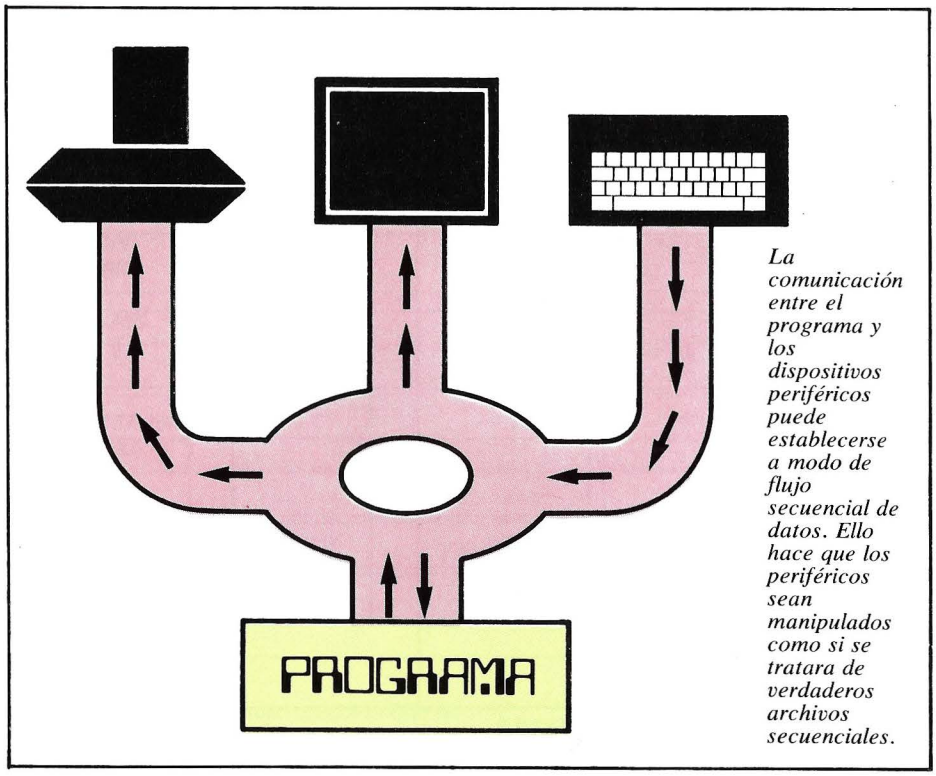
Los archivos se crean habitualmente en dispositivos de almacenamiento. Los más utilizados en el terreno de los microordenadores son la cinta en casete y el disco flexible.



```
OPEN "0",#1,"SCRN:"
```

Los comandos para tratamiento de archivos secuenciales permiten también la comunicación con otros periféricos; por ejemplo, con la pantalla u órgano de visualización.

El modem es un dispositivo que permite la comunicación entre ordenadores distantes, haciendo uso de la red telefónica. Las redes locales son agrupaciones de ordenadores conectados entre sí. Por regla general, éstos comparten ciertos periféricos como puede ser la impresora. Como su nombre indica, una red local une ordenadores (y periféricos) que se encentran próximos (a menudo en la misma habitación o habitaciones contiguas).
La indicación del dispositivo adecuado se suele incluir en el comando de apertura del canal OPEN, y más concretamente en el nombre del archivo. Así, para la apertura de un canal asociado al archivo DATOS, se pueden especificar los siguientes nombres en el argumento de OPEN.



La comunicación entre el programa y los dispositivos periféricos puede establecerse a modo de flujo secuencial de datos. Ello hace que los periféricos sean manipulados como si se tratara de verdaderos archivos secuenciales.

—Magnetófono de casetes
—Unidad de disco
—Pantalla
—Impresora
—Modem o red local

En la lista se han incluido las unidades de almacenamiento externo (magnetófono y unidad de disco), ya que la apertura de un archivo en ellas lleva implícita la comunicación a través de un canal. En el último lugar de la lista se han mencionado dos modalidades de comunicación genérica.

"CAS:DATOS"
"D:DATOS"
"SCRN:DATOS:
"KYBD:DATOS"
"LPT:DATOS"

Los caracteres que preceden a los dos puntos especifican el periférico. En el ejemplo, corresponden a: casete, unidad de disco, pantalla, teclado e impresora. Estas formulaciones no están estandarizadas. Aquí se ha tomado una muy semejante a la adoptada por el BASIC de Microsoft.

La identificación de periféricos se realiza de formas muy heterogéneas, dependiendo de cada aparato en particular.
En algunos casos, cada dispositivo tiene asignado un número y se consideran permanentemente abiertos. Así, bastaría con teclear PRINT#3,A para mandar el dato A a la impresora (suponiendo que el 3 fuera su canal asociado).
Abriendo los canales adecuados se puede establecer la comunicación con cada periférico. A continuación se describe el posible empleo que se puede hacer de cada uno.

- Teclado:

El teclado es un dispositivo de entrada de datos. Ello significa que sólo se pueden recibir datos y nunca mandarlos hacia el teclado. En la mayor parte de los casos no es necesario abrir un canal para la comunicación con este periférico. De utilizar un canal, el resultado de ejecutar los comandos INPUT# y LINE INPUT# será idéntico al de INPUT y LINE INPUT.

- Pantalla:

El caso de la pantalla es similar al del teclado, con la única diferencia de que ésta sólo admite comandos de salida de datos. La apertura de un canal para la pantalla suele ponerse en práctica en los ordenadores que disponen de varios modos gráficos. En tal caso se utiliza el canal para escribir caracteres cuuando se está en un modo gráfico que no admite texto directamente.

- Impresora:

Derterminados ordenadores no poseen instrucciones BASIC apropiadas para el manejo de la impresora. Cuando sucede esto, el usuario se ve obligado a crear un canal y trabajar con la impresora como si se tratara de un archivo de datos. De nuevo, los únicos comandos utilizables son los de salida de datos: PRINT# y WRITE#.
Normalmente, en la apertura de canales para estos periféricos no es necesario adjuntar un nombre de archivo. Para crear un canal para la comunicación con la impresora basta con ejecutar la orden:

```
OPEN "O",#1,"LPT:"
```

En efecto, no es necesario identificar un archivo específico ya que, en realidad, tal archivo es puramente ficticio.

Forth (8)

Aritmética avanzada

El sistema de numeración casi generalizado en el ámbito del ser humano es el decimal o de base diez. Ello parece derivar del hecho que la forma más sencilla de contar no es otra que utilizar los dedos: y como quiera que son diez los dedos que se tienen entre las dos manos, el sistema de numeración más obvio y natural es el de base diez.

Desde luego pueden emplearse otros sistemas de numeración de distinta base. Tal y como se ha detallado a lo largo de la obra, en el terreno de la informática se utilizan los sistemas de numeración binario, octal y hexadecimal, cuyas bases son, respectivamente: 2, 8 y 16.

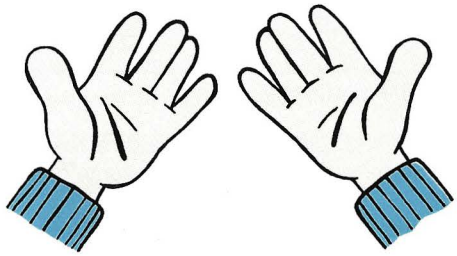
Un artículo que puede ser de gran ayuda para comprender las bases de numeración es el ábaco.

Trabajar en un determinado sistema de numeración es muy semejante a operar con un ábaco cuyo número de bolas en cada varilla sea igual a la base menos una.

Hemos señalado que los sistemas de numeración más utilizados son aquellos cuyas bases coinciden con 2, 8, 10 y 16. Las razones que motivan su empleo son diversas. En primera instancia, la base decimal es la más obvia en el ámbito humano. A su vez, la binaria es la que emplean internamente los ordenadores. Y por último, el uso de la octal y la hexadecimal se fundamenta en que son muy adecuadas para realizar conversiones entre las bases decimal y binaria; de ahí su frecuente presencia en el mundo de la informática.

TRABAJANDO CON OTRAS BASES EN FORTH

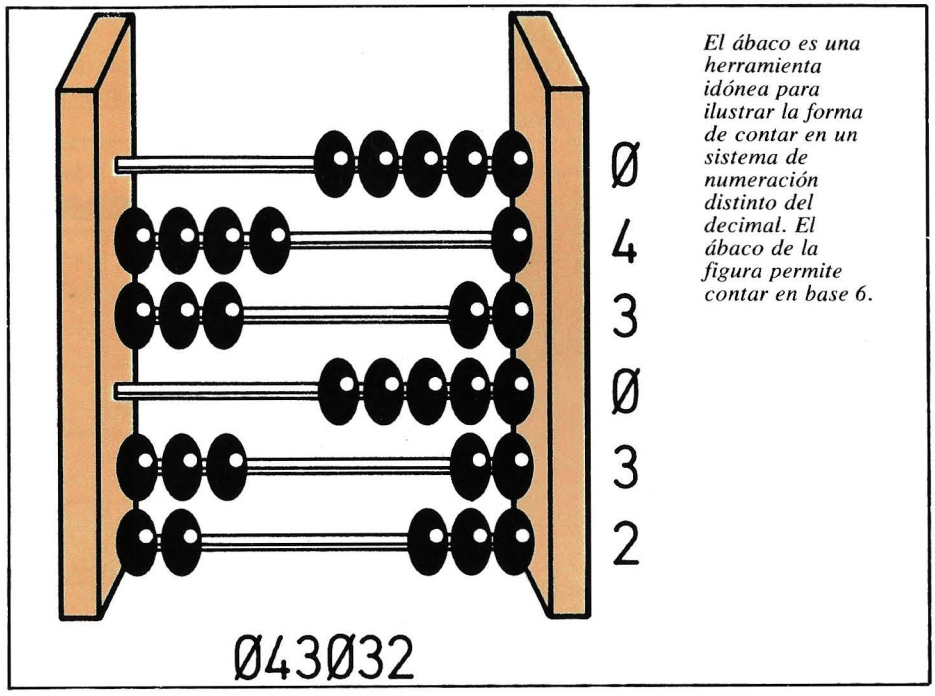
Se ha dicho ya que los ordenadores trabajan internamente en el sistema binario; no obstante están preparados para aceptar números en otras bases de numeración, bases que suelen coincidir generalmente con la decimal, octal y hexadecimal. El lenguaje FORTH contempla tal diversidad de sistemas de numeración, hasta el punto de que permite trabajar en cualquier base de numeración que desee el usuario; basta tan sólo con comunicarlo al ordenador de la forma adecuada.



El sistema de numeración más tradicional y difundido es el decimal o de base diez. Probablemente, su uso generalizado tiene mucho que ver con el hecho de que el hombre posea diez dedos entre ambas manos.

Normalmente, el FORTH trabaja recibiendo y presentando datos en formato decimal; ésta es la base que se asume al conectar el aparato bajo el control del FORTH.

Para comprobar con qué base se está trabajando en cada momento, es necesario acceder a una variable del sistema que, precisamente, recibe el nombre de BASE. Esta variable es de ocho bits, por lo que no es posible trabajar con ella mediante las palabras @ y !. Ello supone que para acceder a la misma hay que recurrir a dos palabras que también operan con direcciones de memoria —al igual que @ y !— aunque accediendo tan sólo a un byte; estas son: C @ y C!



El ábaco es una herramienta idónea para ilustrar la forma de contar en un sistema de numeración distinto del decimal. El ábaco de la figura permite contar en base 6.

La actuación de las palabras BASE C! y C @ es la que se detalla a continuación.
La palabra BASE deposita en la pila la dirección de memoria correspondiente a la variable que comparte dicho nombre; a continuación, las palabras C! y C efectuarán las operaciones que se ordenen a partir de esa dirección de memoria.
Por ejemplo, si se desea conocer la base con la que se trabaja, puede ejecutarse la siguiente orden:

BASE C@ . <CR>

La respuesta de la máquina será:

```
BASE C@. 10 OK
```

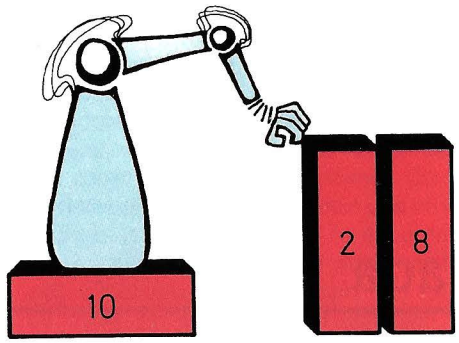
lo cual señala que se está trabajando con el sistema de numeración decimal (base 10).
Para cambiar de base, tan sólo es necesario emplear la palabra C! de la siguiente forma:

2 BASE C! <CR>

Mediante esta configuración de palabras se instruye al ordenador para que cambie el sistema de numeración en curso, de base 10, pasando a base 2. La pantalla mostrará la siguiente respuesta:

```
2 BASE C! OK
```

Acto seguido pueden ya efectuarse las operaciones que se deseen en base 2; por ejemplo, una sencilla suma de 2 más 2 unidades:



La base de numeración habitual en FORTH es la decimal; si bien, este lenguaje está capacitado para operar con otras bases de numeración.

10 10 + . <CR>

(Cabe recordar que en el sistema binario, el número decimal 2 se representa como 10).
La respuesta en pantalla coincidirá con la que sigue:

```
10 10 + . 100 OK
```

En efecto, la suma de 10 (2 en decimal) más 10 (2 en decimal) es el número binario 100 (el cuatro en el sistema de numeración decimal).
Suponga que ahora deseamos conocer cuál es la base con la que estamos trabajando. Para ello, habrá que repetir el mismo proceso realizado anteriormente:

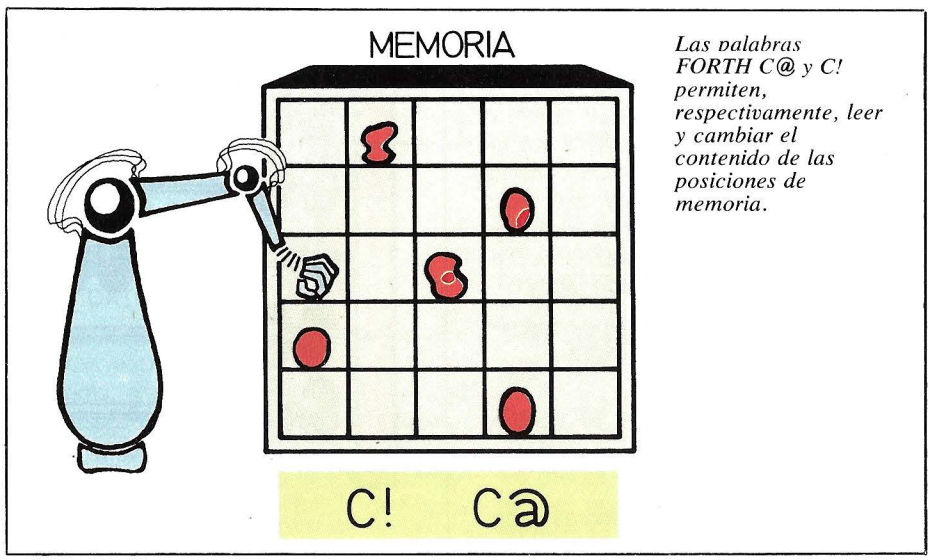
BASE C@ . <CR>

Y la respuesta —tal vez sorprendente— será la que aparece en la siguiente pantalla:

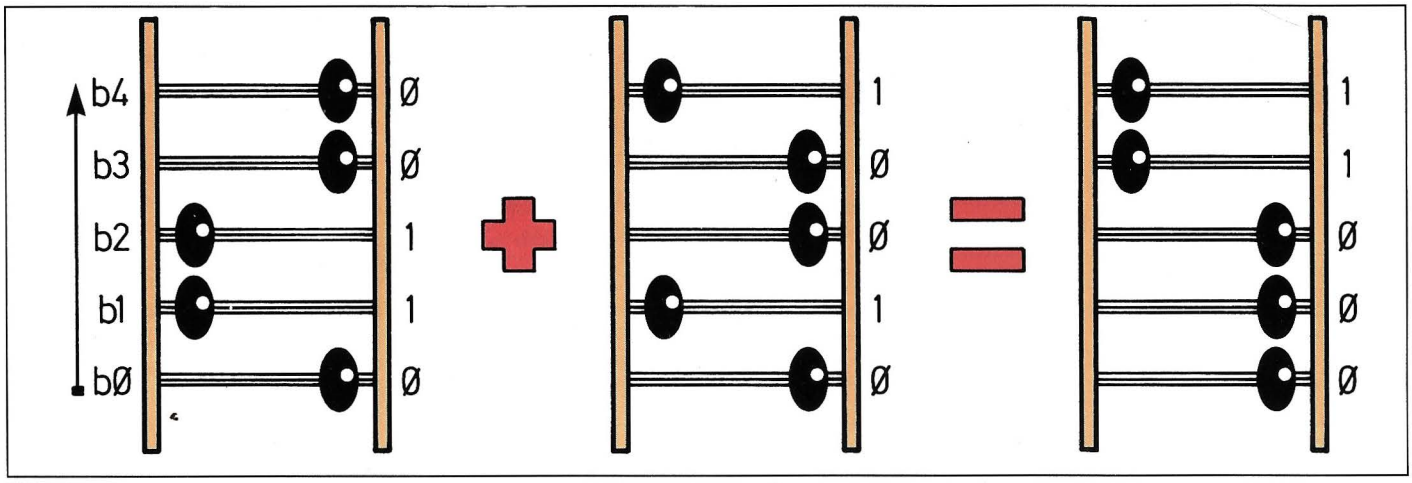
```
BASE C @ . 10 OK
```

¿Qué ha sucedido?.. Nada anormal, puesto que estamos trabajando en base dos; y no hay que olvidar que el número dos se representa en binario como 10. Evidentemente, ésta puede ser una fuente de problemas, ya que el usuario no tiene por qué saber en qué base opera en un determinado momento. Para evitar tal inconveniente, resulta muy oportuno volver al sistema decimal, mediante la palabra DECIMAL, antes de solicitar al ordenador que muestre la base en la que se trabaja.

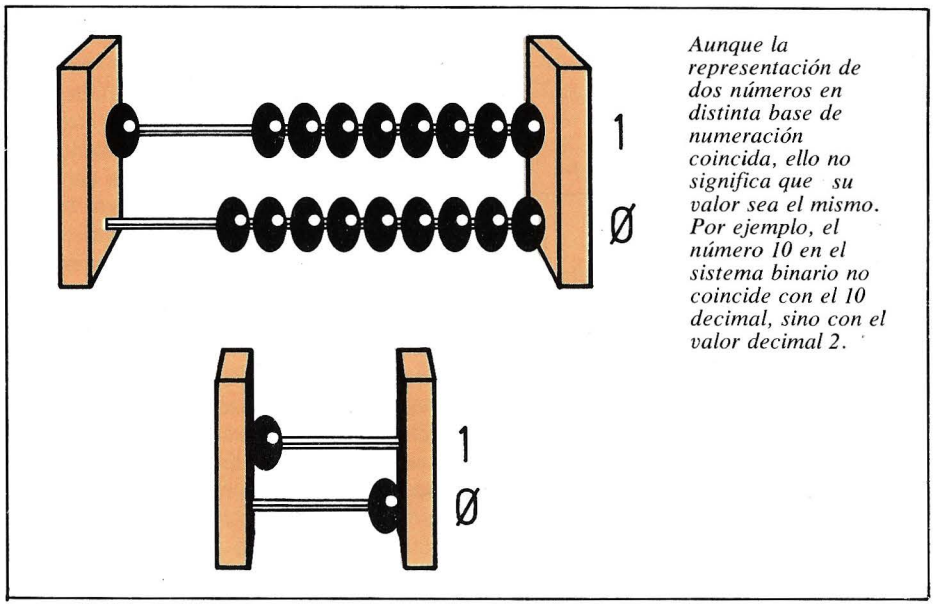
DECIMAL <CR>



Las palabras FORTH C@ y C! permiten, respectivamente, leer y cambiar el contenido de las posiciones de memoria.



Representación mediante un ábaco de la suma de dos números en expresión binaria.



Aunque la representación de dos números en distinta base de numeración coincida, ello no significa que su valor sea el mismo. Por ejemplo, el número 10 en el sistema binario no coincide con el 10 decimal, sino con el valor decimal 2.

DECIMAL OK

Con ello, se tendrá la garantía de haber vuelto al sistema de numeración decimal. También desde el propio sistema binario se podría haber ordenado el retorno a decimal mediante las siguientes palabras:

1010 BASE C! <CR>

(1010 en binario, corresponde al número 10 decimal). A lo que el ordenador respondería con el siguiente mensaje:

1010 BASE C! OK

En este punto se podría seleccionar otro sistema de numeración, el hexadecimal, de gran utilidad en determinadas ocasiones:

16 BASE C! <CR>

La imagen en pantalla mostrará el siguiente aspecto:

16 BASE C! OK

Acto seguido pueden ya efectuarse algunas acciones que revelarán la activación del sistema de numeración hexadecimal. Por ejemplo:

1A 122 + . <CR>

1A 122 + . 13C OK

Los dígitos utilizados en el sistema de

TABLA DE ORDENES FORTH

PALABRA	DESCRIPCION	TIPO
BASE	Contiene la base del sistema de numeración con el que se está trabajando.	Variable del sistema.
C	Lee el contenido de una posición de memoria.	Manejo de memoria.
C!	Cambia el contenido de una posición de memoria.	Manejo de memoria.
DECIMAL	Restaura el valor de la base del sistema de numeración utilizado a decimal.	Sistema de numeración.

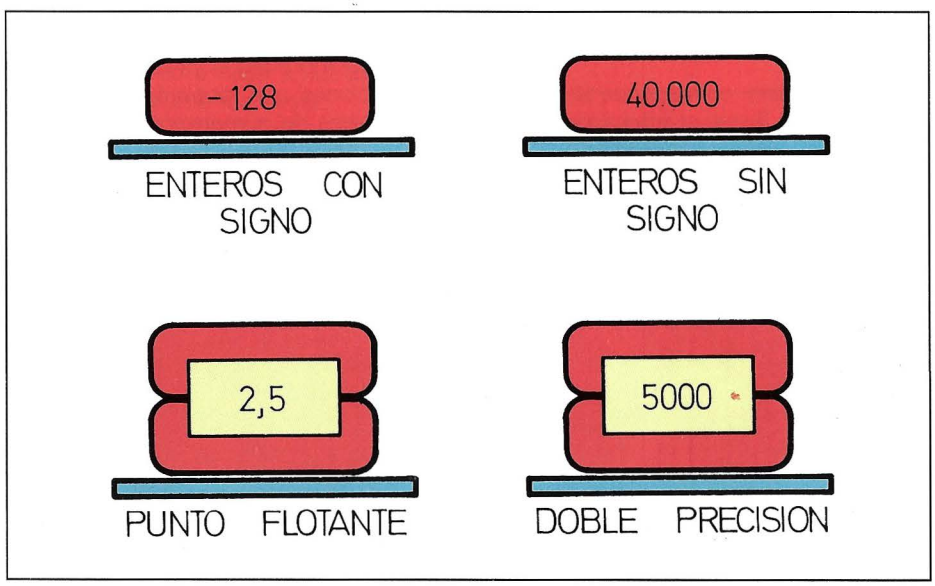
TIPOS DE NUMEROS UTILIZABLES EN FORTH

TIPO	BYTES	RANGO	NOTACION CIENTIFICA
Enteros con signo	2	−32768 a 32767	No
Enteros sin signo	2	0 a 65535	No
Punto flotante	4	−9.99999E64 a 9.99999E62	Sí
Doble precisión	4	Según su empleo	No

numeración de base 16 coinciden con las cifras decimales de 0 a 9, además de con las letras A a la F para representar, respectivamente, los valores 10, 11, 12, 13, 14 y 15.

Es obvio pues, que el lenguaje FORTH permite al usuario trabajar a voluntad con cualquier sistema de numeración.

En este caso, hay que tener en cuenta que la base seleccionada sólo afecta a lo que se muestra en la pantalla; internamente, el ordenador recoge los números en una base, los convierte al sistema binario, los opera, y cuando es necesario mostrar el resultado por pantalla, recoge los datos de memoria en binario y los visualiza previa conversión al sistema de numeración seleccionado por el usuario.



Distintos tipos de numeración utilizables en FORTH y su respectiva ocupación de elementos de la pila (cabe recordar que cada elemento de la pila ocupa dos bytes de memoria).

DISTINTOS TIPOS DE NUMEROS

Hasta ahora se han manipulado tan sólo números de un mismo tipo: números enteros con signo y de simple precisión. Sin lugar a dudas, el trabajo con ellos resulta más sencillo; si bien, también es cierto que se ven limitados por sus características. Este es el momento de presentar nuevos tipos de números que ampliarán el abanico de posibilidades del FORTH. La tabla adjunta resume las distintas categorías de números normalmente utilizables en FORTH, así como sus características más representativas.

En ella figuran cuatro tipos de números: cada tipo resulta especialmente adecuado para un distinto marco de aplicación. La primera columna de la tabla incluye el nombre que se otorga a cada categoría numérica; nombre que suele ser lo suficientemente representativo como para evidenciar sus características más relevantes.

La columna bytes señala el número de bytes de memoria necesarios para almacenar un número de dicho tipo.

Hay que recordar que cada elemento de la pila ocupa dos bytes de memoria; de ahí que un número entero con signo esté formado tan sólo por un elemento de la pila, mientras que un número de doble precisión lo está por dos elementos.

La columna de rango define el margen dentro del que pueden estar comprendidos los valores de un número, de acuerdo a su correspondiente tipo. En el caso de los números de doble precisión no se especifica su rango, ya que se pueden utilizar de diversas formas (dependiendo, por ejemplo, de si se considera el signo o no) y por lo tanto el rango puede variar.

La notación científica consiste en fraccionar el número en dos partes: un primer número en punto flotante seguido por la letra E y, tras ésta, un número entero con signo correspondiente a la potencia de 10 por la que deseamos multiplicarlo. Esta notación resultará ampliamente conocida por los habituales usuarios de calculadoras.

En capítulos sucesivos se estudiarán cada uno de estos tipos, entrando en detalles acerca de las palabras a utilizar para trabajar con ellos.

T.O.S. (1)

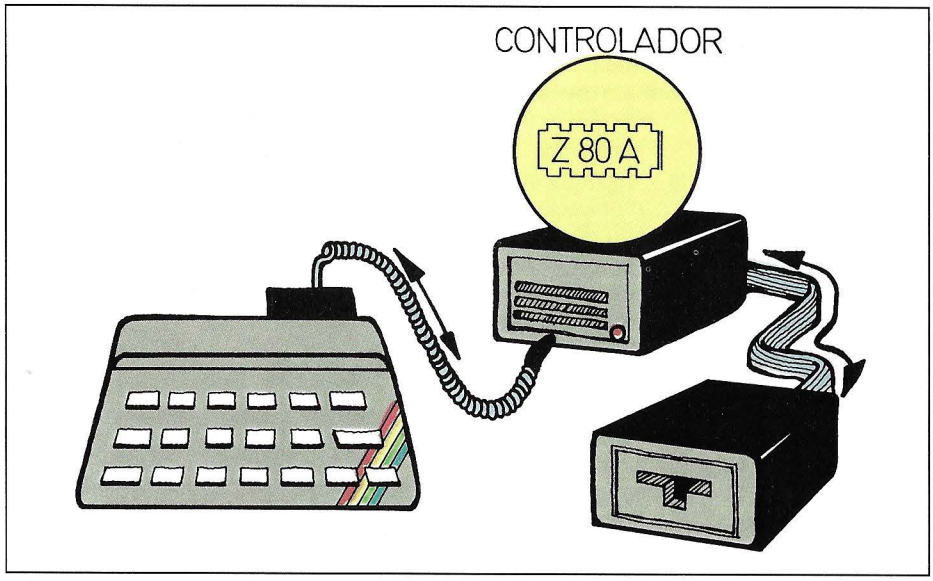
Un sistema operativo de disco para el ZX-SPECTRUM

Los ordenadores domésticos, cuyo destino más frecuente es la ejecución de programas de juego, ascienden algunos peldaños en su nivel de utilidad al contar con periféricos de almacenamiento de mayor capacidad y velocidad de acceso que el tradicional casete.

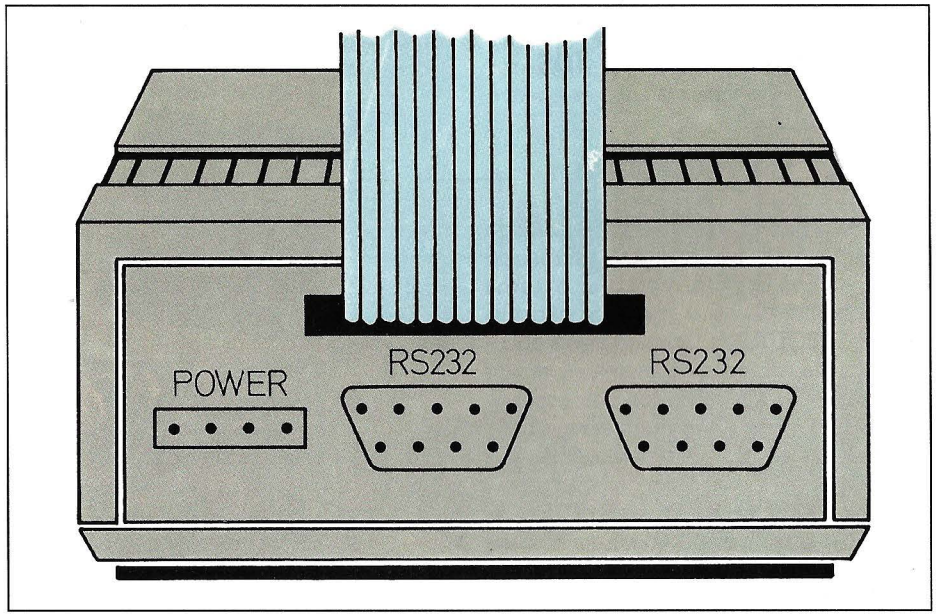
Un claro ejemplo lo constituye el ZX-SPECTRUM. Las unidades de disco diseñadas para este equipo amplían su capacidad de operación, permitiendo su entrada en el terreno de las aplicaciones de gestión; de moderada complejidad, aunque de incuestionable interés para el usuario.

En comparación con las memorias de masa tradicionalmente utilizadas con el ZX-SPECTRUM —casetes y microdrives—, la unidad de disco flexible aporta una mayor capacidad y velocidad, además de permitir el acceso directo a los programas y ficheros de datos sin tener que leer previamente toda la información que le precede en el soporte de almacenamiento.

Una de las unidades de disco más popularizadas es la fabricada por la firma americana TIMEX, y comercializada en España bajo la denominación de INVESDISK-200.



El controlador de la unidad de disco incluye un microprocesador Z80A; éste controla la lectura y escritura de información en el disquete, liberando de esta tarea el ZX-SPECTRUM.



Además de la función de gestionar las entradas y salidas de los disquetes, el controlador abre una doble vía de diálogo con el mundo exterior a través de dos conectores para comunicación serie RS/232.

DESCRIPCION DEL SISTEMA

Los productos encuadrables en la gama Sinclair muestran una acusada tendencia a separarse de los estándares del mercado, marcando un estilo de desarrollo muy propio y peculiar (véase el atípico microdrive). La unidad de disquete INVESDISK-200 sigue la misma tónica; lejos de adoptar un aspecto tradicional, aparece constituida por un mínimo de tres módulos:

—Fuente de alimentación
—Controlador de la unidad de disquete
—Unidad de disquete

La fuente de alimentación es capaz de nutrir al controlador de la unidad de disquete y a un máximo de dos unidades lectoras. Al efecto, incluye los conectores adecuados localizados en el panel posterior del módulo de alimentación.

El controlador es, con diferencia, la zona más importante del sistema de almacenamiento en disco. En él reside el sistema operativo TOS (Timex Operating System). El controlador incluye un microprocesador del tipo Z80A, además de su propia zona de memoria y dos conectores para comunicación externa en formato serie (RS-232).

El módulo de control es quien gobierna los procesos de lectura y escritura sobre el disquete, así como la comunicación con el entorno exterior, liberando de esta tarea al ZX-SPECTRUM. La conexión con el microordenador se establece a través de un cartucho de interface que conecta con el bus de expansión del SPECTRUM.

La unidad de disquete trabaja con disco flexible de 3 pulgadas, grabando las dos caras del mismo en doble intensidad. Ello equivale a una capacidad de almacenamiento total de 320 Kbytes una vez formateado el disquete (160 K por cara). Los discos ofrecen un aspecto sólido, al tiempo que garantizan la seguridad en su manipulación, al estar encerrados en estuche de plástico rígido.

EL SISTEMA OPERATIVO TOS

El sistema operativo TOS es el encargado de gestionar el trasiego de información entre el soporte de memoria externa —los disquetes— y la memoria principal del ZX-SPECTRUM. También tiene encomendada la misión de transmitir y recibir información a través de los dos accesos de comunicación en formato serie.

Uno de los puntos más destacables del TOS es que la estructura de la información contenida en los disquetes es de tipo jerárquico o en árbol invertido. Esta se basa en la configuración típica de directorios raíz, subdirectorios y ficheros.

Así pues, la información puede agruparse fácilmente en directorios que contengan ficheros con una serie de características comunes, permitiendo su rápida localización.

COMANDOS DEL TOS

Quizás la parte más floja de todo el sistema operativo resida en la sintaxis de los comandos. Para que éstos puedan acomodarse a las claves o "tokens" directamente introducibles desde el teclado del ZX-SPECTRUM, ha sido preciso sacrificar la oportuna denominación de los comandos; hasta el punto de que el significado de algunos de ellos no es del todo evidente, resistiéndose su aprendizaje bastante más de lo que cabría desear.

Como norma general, todos los coman-

Aspecto "al completo" del sistema de almacenamiento en disco INVESDISK-200. Este consta de la fuente de alimentación, la unidad lectora de discos, el controlador y el módulo de interface asociado directamente al ZX-SPECTRUM.

La unidad de disco, gestionada bajo el control del sistema operativo T.O.S., constituye una alternativa ventajosa a los tradicionales microdrives; tanto por lo que respecta a la capacidad, como a la velocidad de acceso y organización de la información almacenada.

dos del sistema operativo se componen de una palabra clave más un asterisco y, en ciertos casos, de una lista de atributos u opciones que definen totalmente el comando.
La ejecución de los comandos del TOS no interfiere en absoluto con las instrucciones propias del ZX-SPECTRUM relativas al control del casete. De ahí que sea posible utilizar el casete al mismo tiempo que la unidad de disco, incluso desde un mismo programa.
Los comandos pueden ser ejecutados activándolos desde el teclado, o bien desde un programa BASIC que los contenga al igual que cualquier otra instrucción de este lenguaje; ello otorga al sistema operativo TOS la propiedad de constituirse en una virtual extensión del BASIC.

NOMENCLATURA DE LOS FICHEROS

Como quiera que los comandos se refieren primordialmente a ficheros, antes de entrar en la descripción de éstos es conveniente precisar la nomenclatura que utilizan los ficheros TOS.
Los nombres de los ficheros se definen como cadenas de ocho caracteres, seguidos, opcionalmente, por cadenas de tres caracteres que indican la extensión o tipo de fichero; ambas cadenas deben aparecer separadas por un punto.
En la confección de los nombres de los ficheros es lícito utilizar símbolos, además de dígitos y caracteres, siempre y cuando los símbolos no sean dobles (<>, >=, etc.) o coincidan con alguno de los símbolos reservados (", '?' y '+').
Nombres correctos son:

"NOMINA.DATA"
"PAGOS.DIR" (".DIR" indica que es un directorio)
"CORREO.SPC" (".SPC" indica que es un canal de comunicaciones)

Para dar mayor agilidad a la acción de los comandos que actúan sobre ficheros, cabe la posibilidad de definir nombres genéricos de ficheros mediante la utilización de diversos símbolos especiales. Dichos símbolos son '+', que reemplaza a un nombre o a un tipo, y '?' que sustituye a un carácter. Por ejemplo:

"+.+": Hace referencia a cualquier fichero de cualquier tipo.

"+.BAS": Alude a todos los ficheros de tipo BAS.

"PON???": Se refiere a los ficheros de seis letras que empiezan por PON.

COMANDOS GENERALES

Los comandos que se describen a continuación son de uso general, resultando válidos para cualquier tipo de fichero.

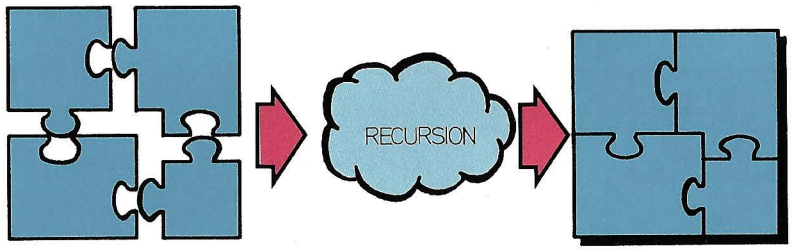
Recursividad e iteración

La informática clásica utiliza casi exclusivamente una metodología iterativa en la confección de programas (¿Qué programador no ha utilizado en múltiples ocasiones la sentencia: "¿Desea ejecutar el programa de nuevo?" ...?). No obstante, existen otros métodos, distintos de la iteración, para resolver problemas o tareas de carácter repetitivo; entre ellos cabe destacar la recursividad.
Antes de entrar en más detalles, es conveniente definir *iteración* y *recurrencia* de una forma concisa para poder apreciar claramente sus diferencias.
Un proceso iterativo es, esencialmente, un proceso de repetición que se basa en información conocida; se resuelve un problema para un conjunto determinado de condiciones y, en definitiva, es posible resolver de nuevo el mismo problema para otro conjunto de condiciones. En suma, se trata de un típico proceso en serie: primero se resuelve un caso, luego otro, etc.
Por contra, un proceso recursivo fundamenta su actividad en información desconocida en principio. Es decir, un problema complejo se reduce a otro más simple que servirá como paso para resolver el problema inicial. Y si esta simplificación todavía no llevase directamente a la resolución del problema, se volvería a fraccionar en situaciones más sencillas hasta que fuera posible resolverlo. Este método es claramente un proceso en paralelo, ya que una vez que se ha obtenido una solución las demás son inmediatas.
Un ejemplo válido lo constituye la serie de Fibonacci, definida como:

$f_n = f_{n-1} + f_{n-2}$

para $n>1$, siendo $f_0=1$ y $f_1=1$

El procedimiento iterativo para calcular el elemento sexto de la serie partiría de los elementos más bajos, calculando a continuación los elementos posteriores. En cambio, el procedimiento recursivo comenzaría el cálculo por el elemento sexto; al no poderlo calcular probaría con el quinto... y así sucesivamente hasta llegar al segundo, en cuyo caso sí podría calcularlo, obteniéndose a continuación los elementos superiores hasta el sexto.



La iteración resuelve el mismo problema con determinadas condiciones, mientras que la recursión subdivide el problema en partes más sencillas que, una vez resueltas, componen la solución al problema inicial.

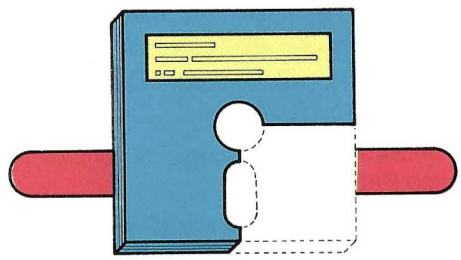
FORMAT*
Antes de utilizar cualquier disquete por vez primera es preciso formatearlo, o lo que es lo mismo, definir las pistas y los sectores que se utilizarán en el transcurso de las posteriores operaciones de lectura y escritura.
Este comando ha de ejecutarse tan sólo cuando sea estrictamente necesario, pues su ejecución supone destruir la información contenida en el disquete. En todo caso, el sistema operativo avisa de este hecho, de tal forma que es posible anular su ejecución una vez que ha sido invocado.
Su formato es:

FORMAT* "unidad" TO "nombre del disco"

LOAD*
Carga un archivo residente en disquete, llevándolo a la memoria del ZX-SPECTRUM. Es análogo al comando destinado a cargar archivos grabados en cinta, por lo que las opciones disponibles son las mismas en ambos casos.
Su formato es:

LOAD* "nombre" [opciones]

SAVE*
Graba en disquete un programa o una determinada zona de la memoria del ZX-SPECTRUM. El sistema pregunta al usuario si debe reescribirse el fichero en el caso de que el nombre indicado en el comando haga referencia a un fichero ya existente. Al igual que en el caso anterior



Los ficheros residentes en el disco pueden escamotearse a las miradas indiscretas haciéndolos invisibles para el usuario. Al hacerlos invisibles, su presencia no se verá reflejada en el directorio del disco.

es análogo al comando destinado a la grabación en casete.
Su formato es:

SAVE* "nombre" [opciones]

MERGE*
Mezcla un nuevo programa y sus variables con el programa existente en la memoria del ZX-SPECTRUM.
Su formato es:

MERGE* "nombre"

LET*
Cambia el nombre de un fichero, directorio o canal.
Su formato es:

LET* "nombre antiguo" TO "nombre nuevo"

MOVE*
Copia un fichero en otro fichero.
Su formato es:

MOVE* "nombre origen" TO "nombre destino"

ERASE*
Borra un fichero en el caso de que éste no se encuentre protegido.
Su formato es:

ERASE* "nombre" [n]

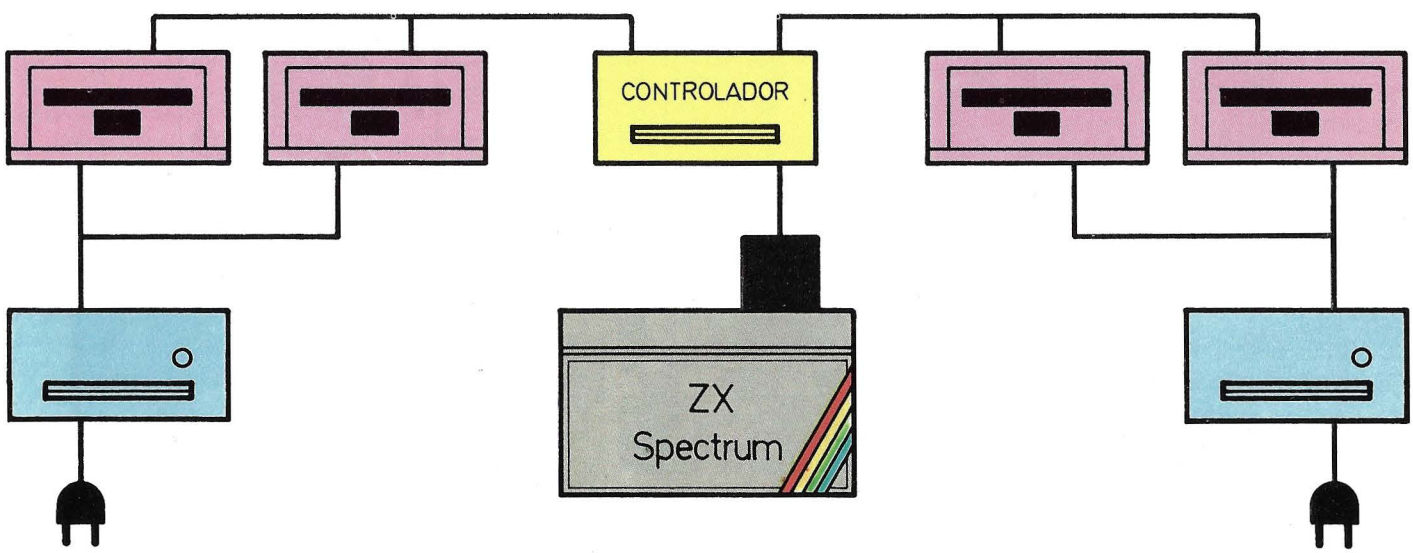
(La opción n es para que el sistema no solicite la confirmación del usuario acerca de si efectivamente desea borrar el fichero).

ATTR*
Define la protección de un fichero:
Su formato es:

ATTR* "nombre" opción.

Siendo las posibles opciones:
P – protege un fichero
U – desprotege un fichero
I – vuelve invisible un fichero
V – devuelve la visibilidad a un fichero.

CAT*
Lista el contenido de un directorio, indicando los nombres de los ficheros (columna "Name"), su tipo (columna "Typ"), la cantidad de bytes que contienen (columna "Size"), el espacio ocupado (columna "Alloc"), si están abiertos o cerrados (columna "S") y el tipo de operación (columna "P").

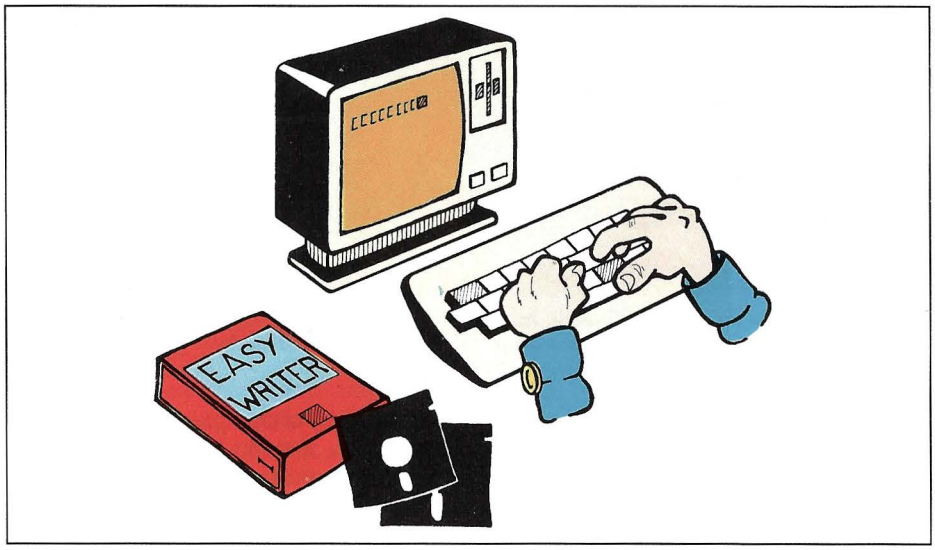


La máxima configuración que se puede alcanzar con la INVESDISK-200 llega a cuatro unidades de disquete, gobernadas por el mismo controlador y alimentadas por dos fuentes de alimentación.

Easy Writer (y 3)

Sesión práctica con el paquete Easy Writer

Para concluir el estudio del programa para el tratamiento de textos EASY WRITER, vamos a centrarnos en los comandos adicionales del menú "Easy Writer File System". Mediante ellos el usuario puede desencadenar operaciones propias del proceso de palabras. También en este capítulo se incluirá una sesión práctica con el paquete EASY WRITER, concretada en la producción de un documento en el que se resuman las características fundamentales asociadas a este programa.



COMANDOS ADICIONALES

Tanto desde el menú de ayuda como desde el menú para la gestión de ficheros, se puede acceder a un nuevo menú denominado: "ADDITIONAL COMMANDS". Para ello basta con pulsar la tecla [F4]. De inmediato aparecerá en la pantalla una lista de comandos, específicos para el proceso de textos, entre los que el usuario puede elegir a voluntad:

(A) ALIGN TEXT

Cuando se esté editando un documento, el usuario no tiene por qué prestar atención a la alineación de los márgenes del texto. Normalmente, todas las líneas comenzarán en la columna 1; sin embargo, cada una de ellas terminará en una posición completamente distinta, que dará al texto un aspecto irregular. Mediante el comando ALIGN, el programa se encargará de eliminar los espacios en blanco y "moverá" la ubicación de las líneas de forma que todas acaben aproximadamente en la misma posición; por supuesto, excepción hecha de las líneas que acaben con un punto y aparte

(C) CENTER A LINE

Antes de ejecutar este comando, el usuario debe situar el cursor sobre la línea que se desee centrar; a continuación pulsará la tecla [F4] para situarse en el menú de comandos adicionales —siempre que no estuviera ya en él—, y por último pulsará la tecla [C].

Inmediatamente, el contenido de la línea se desplazará hacia la derecha o izquierda hasta quedar perfectamente centrado.

(H) HORIZONTAL MOTION INDEX

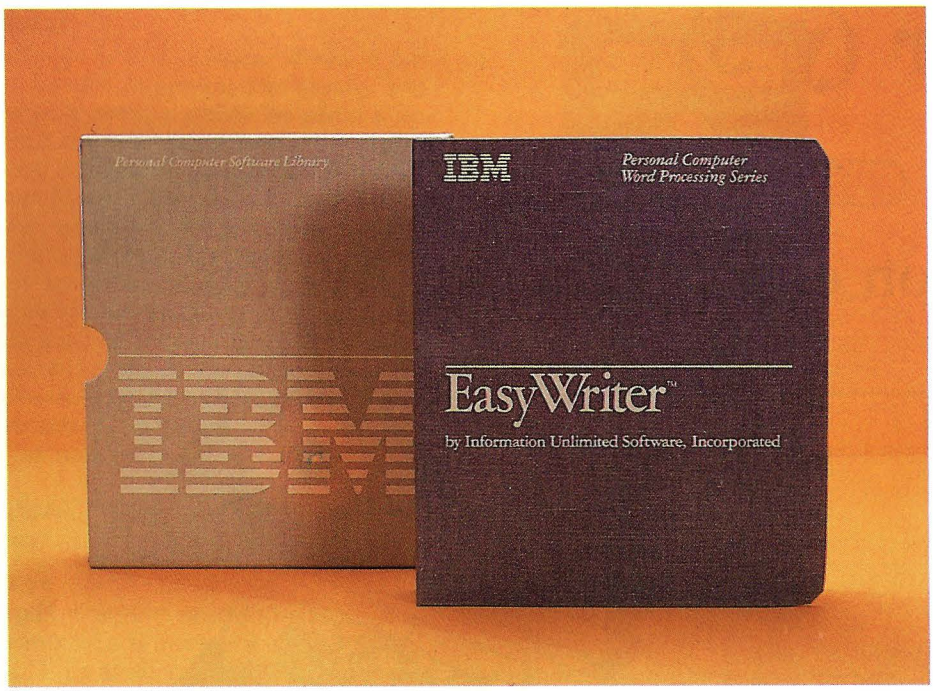
Esta opción está destinada a hacer compatibles los documentos creados con EASY WRITER con impresoras que no sean de tipo IBM.

(J) JUSTIFY

El comando JUSTIFY tiene un comportamiento similar al de un conmutador, esto es: accionando la tecla [J] el comando queda activado (ON), y volviendo a pulsar [J] el comando queda desactivado (OFF). Mediante la opción ON, EASY WRITER se encargará de justificar sobre el margen derecho; en cambio, mediante la opción OFF no se produce la justificación del texto.

```
                    A D D I T I O N A L   C O M M A N D S
-------------------------------------------------------------------------------
 A - ALIGN TEXT              M - MARGIN SETTINGS        T   - TAB SETTINGS
 C - CENTER A LINE           N - PAGE #/# OF COPIES     W   - WORD COUNT
 H - HMI SETTING             P - PRINT TO SCREEN        F4  - GO TO EDITOR
 J - JUSTIFY ON/OFF          S - SEARCH AND REPLACE     F10 - GO TO FILE SYSTEM

 COMMAND?
L                                                                        R
0----+----1----+----2----+----3----+----4----+----5----+----6----+----7----+----
```

El menú de comandos adicionales permite al usuario ejecutar algunas funciones destinadas al tratamiento de palabras. Este menú puede ser invocado partiendo de cualquier otro menú de la aplicación EASY WRITER.



EASY WRITER es un programa para el tratamiento de textos comercializado en distintas versiones. La más difundida es la orientada a su explotación en ordenadores personales del tipo IBM-PC o compatibles.

Si después de haber escrito un texto con JUSTIFY OFF, se desea justificar el mismo, basta con pulsar la tecla [J] e inmediatamente aparecerá en la pantalla la sentencia JUST–ON; a continuación, se pulsará la tecla [A] para ejecutar el comando de alineamiento con las nuevas condiciones.

(M) MARGINS

EASY WRITER permite situar el margen derecho desde la posición 26 hasta la 254. Por defecto, el margen izquierdo se situará en la posición 0 y el derecho en la 65. Mediante el comando MARGINS, el usuario puede desplazar el margen derecho o realizar una indentación o sangrado sobre el margen izquierdo.
Después de recibir la pulsación de la tecla [M], EASY WRITER emitirá el mensaje ENTER: R,I: y aguardará la introducción de uno o dos valores separados por comas. Si se teclea sólo un dato, éste pasará a ser el nuevo margen derecho; mientras que si se teclean dos, el primero será el nuevo margen derecho y el segundo la indentación.

(N) PAGE#/#OF COPIES

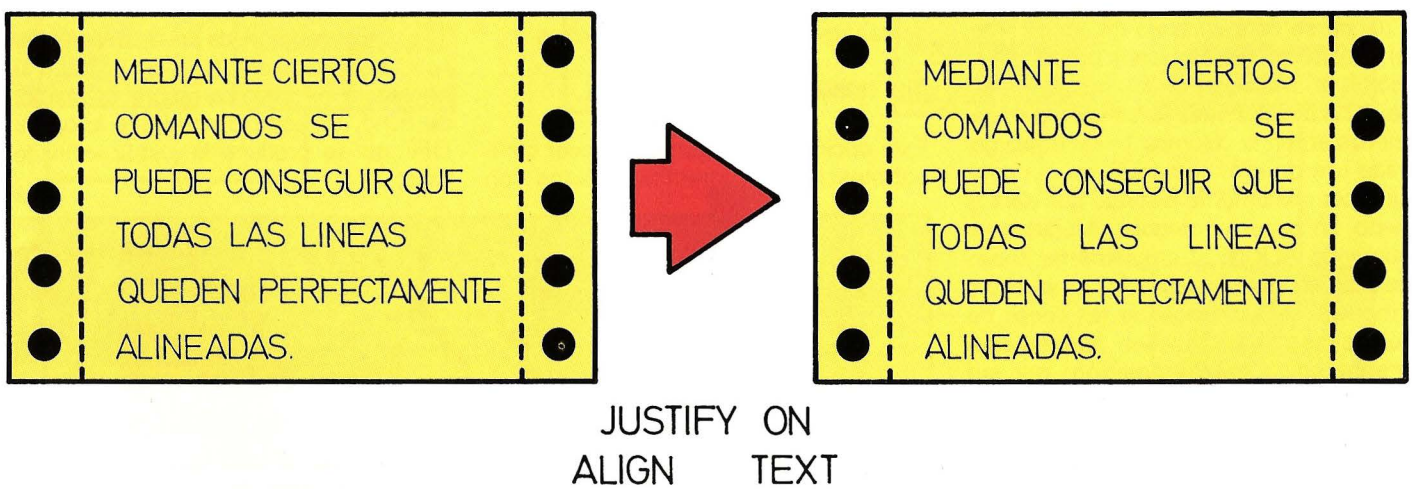
Este comando tiene encomendadas tres misiones distintas. Por un lado permite comenzar la paginación de un documento a partir de cualquier número, y por otro lado sirve para imprimir varias copias de un mismo documento. Como tercera opción, también permite imprimir sólo parte de un documento.
Para ejecutar este comando se debe pulsar la tecla [N] y contestar a las preguntas que aparecerán en la pantalla.

(P) PRINT TO SCREEN

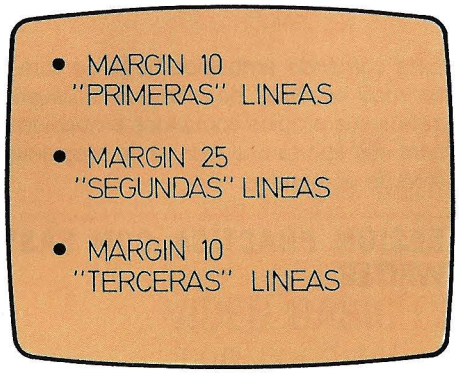
En algunos casos, antes de imprimir un documento en papel resulta interesante observar en la pantalla el aspecto final del mismo; ya sea para comprobar sus márgenes, la paginación, las indentaciones, etc. De esta forma, si el aspecto no es satisfactorio, se podrá modificar antes de su impresión definitiva. Mediante el comando PRINT TO SCREEN, que se ejecuta sin más que pulsar la tecla [P], puede visualizarse por pantalla el aspecto final de un documento.

(S) SEARCH AND REPLACE

Una de las funciones más importantes de todo programa para el tratamiento de textos consiste en la localización y sustitución de palabras. EASY WRITER brinda esta posibilidad mediante el comando SEARCH AND REPLACE. Sin más que pulsar la tecla [S], el programa solicitará: "SEARCH FOR:" y aguardará la introduc-



Para conseguir que todas las líneas terminen en la misma posición es preciso activar el comando JUSTIFY (JUSTIFY ON) y, a continuación, ejecutar el comando ALIGN TEXT.



Los comandos embebidos se identifican porque empiezan siempre por un punto y se encuentran solos en la línea que ocupan.

ción de una cadena de caracteres. Cuando ésta haya sido tecleada, de nuevo solicitará: "REPLACE WITH:" y esperará la introducción de una segunda cadena. Por último, el programa preguntará: "ALL OR SOME?", y esperará como respuesta "A" o "S". A continuación serán localizadas de forma automática todas las ocurrencias de la frase buscada (primera cadena de caracteres), y reemplazadas por la frase sustitutiva (segunda cadena de caracteres). Dicha operación se realizará en todo el texto si se teclea "A" como respuesta a la tercera pregunta, u ocurrencia por ocurrencia si se teclea "S".
Otra posibilidad que ofrece este comando consiste en localizar palabras sin sustituirlas; para ello, basta con pulsar la tecla [S] sin introducir ningún texto en respuesta a la segunda pregunta.

(T) TAB

Cuando el usuario pulsa la tecla [T], EASY WRITER le presentará el siguiente mensaje: "ENTER TAB SETTING SEPARATED BY COMMAS" y quedará esperando una lista de números separados por comas. Estos números servirán desde el momento de la ejecución para prefijar los tabuladores; es decir, al pulsar la tecla [⇄] el cursor se desplazará desde la posición actual a la siguiente columna indicada en la lista de números.

(W) WORD COUNT

El último integrante del menú de comandos adicionales sirve para contar el número de palabras de un texto. Sin más que pulsar la tecla [W], EASY WRITER mostrará en la pantalla el resultado de contar las palabras del documento que se encuentra en la memoria principal.

COMANDOS EMBEBIDOS

Además de todos los comandos descritos hasta este momento, existen otros a los que se denomina "embebidos"; precisamente porque su ejecución no se desencadena desde el menú, sino que son invocados por inclusión del texto. Evidentemente, a la hora de imprimir el documento estos comandos no se escribirán, aunque sí producirán determinados efectos. Para que EASY WRITER pueda diferenciar un comando de las restantes palabras del texto, los comandos embebidos deben seguir ciertas reglas de carácter general:
1. Deben codificarse de forma que en la línea que ocupan, sean los primeros caracteres distintos de espacios en blanco.
2. No se debe escribir ningún tipo de texto a la izquierda del comando en su misma línea.
3. Deben comenzar siempre por el carácter punto (".").
4. Cada línea sólo podrá contener un único comando embebido.
5. Como último carácter del comando debe aparecer el asociado a la pulsación de la tecla [↵] .

Sin pretender dar una lista completa de todos ellos, se relacionan a continuación algunos de los comandos embebidos del EASY WRITER.

.EJECT

Produce un salto de página, de forma que la línea situada a continuación será la primera de la siguiente página.

.EOL

Sirve para definir o redefinir la posición deseada como final de línea.

.FORMSTOP

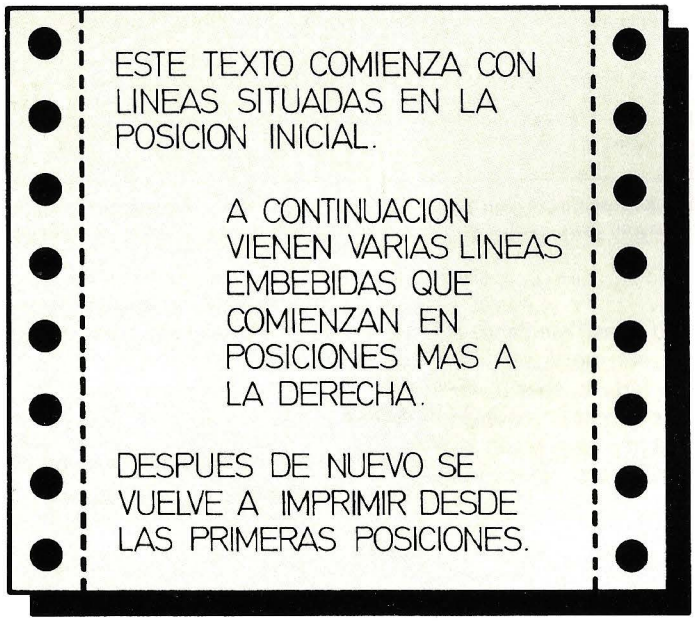
EASY WRITER asume por defecto que la impresora manejará papel continuo. En el caso de trabajar con papel hoja a hoja, debe incluirse el comando .FORMSTOP que detendrá la impresión al finalizar una página; de esta forma, el usuario podrá colocar la siguiente hoja de papel.

.LINES

Por defecto se imprimirán 66 líneas por página; si bien, mediante este comando embebido puede modificarse dicho parámetro.

.MARGIN

Permite definir el margen izquierdo. Este comando puede parecer redundante con el comando MARGINS descrito anterior-



Al imprimir el documento, los comandos embebidos no son escritos, si bien producen los efectos oportunos.

```
Resumen de las características del EASY WRITER
==============================================


EASY WRITER Help Menú:

Por medio del menú de ayuda se puede visualizar en la parte
superior de la pantalla la asignación de funciones a las teclas
"programables", así como las funciones asociadas a las restantes
teclas reservadas al control de la aplicación.


EASY WRITER File System:

El menú para la gestión de ficheros ofrece una serie de opciones
para que el usuario, sin más que ejecutar el comando oportuno,
realice las operaciones necesarias sobre la memoria auxiliar.


 EASY WRITER Comandos adicionales:

 Además de los dos menús descritos anteriormente, el programa
 ofrece la posibilidad de utilizar una serie de comandos
 adicionales de plena utilidad a la hora de procesar textos.


 EASY WRITER Comandos embebidos:

 También es posible incluir algunos comandos dentro del texto, de
 tal forma que al imprimir el documento los comandos producirán su
 efecto, aunque no se imprimirán.


 EASY WRITER Valoración final:

 Las dos características más notables de este procesador de texto
 son su gran potencia para el proceso de palabras y la considerable
 sencillez de uso.
```

La sesión práctica con el paquete EASY WRITER consistirá en ejecutar los pasos necesarios para producir el documento que se reproduce en la figura.

```
Your EasyWriter program diskette contains the following programs:

    AUTOEW ---- Change EasyWriter program diskette to automatically load
                EasyWriter. This is the default.
    CHKDSK ---- Display available space on diskette and in memory
    CONVERT --- Convert EasyWriter Version 1.00 documents to Version 1.10
    DISKCOPY -- Copy or backup diskettes
    EW -------- EasyWriter
    FORMAT ---- Format diskettes
    NOAUTOEW -- Change EasyWriter program diskette to automatically display
                the programs on the diskette when loaded
    RECONFIG -- Reconfigure EasyWriter program diskette for new EasyWriter
                defaults and/or serial printer support
    TRANSFER -- Copy and change EasyWriter file to DOS ASCII file or DOS
                ASCII file to EasyWriter file

On the DOS prompt line below, enter the name of the program you want to
run. Also, you can run a DOS program on another diskette by putting that
diskette in Drive A and entering the program's name.

A>
```

Al abandonar la sesión de trabajo, el control queda de nuevo en manos del sistema operativo y en la pantalla aparecerá una lista con los programas almacenados en el disco de la aplicación.

mente, aunque ello no es así; mediante el comando MARGINS del menú se fijan unos márgenes para toda la sesión; en cambio, mediante el comando embebido .MARGIN puede alterarse el margen izquierdo varias veces dentro de un mismo documento.

.PAGE

Este comando se puede utilizar para indicar desde dentro del propio texto el tipo de numeración deseado para las páginas del documento.

.SPACE

Se limita a escribir tantas líneas en blanco como se indique.

.TITLE

Puede utilizarse para introducir un título en la parte superior o inferior de todas las páginas del documento.

.USER

Este comando embebido tiene un carácter muy especial: permite que el usuario defina sus propios comandos embebidos, para ello aporta una técnica de programación de comandos.

SESION PRACTICA CON EASY WRITER

Dada la sencillez de manejo de este programa, hemos reservado un espacio muy pequeño para ilustrar prácticamente su funcionamiento. En este caso, nos vamos a centrar en producir un documento de una sola página en el que se resuman las características del programa EASY WRITER.

Después de conectar el ordenador y activar la unidad de almacenamiento en donde se vaya a introducir el disquete con el programa EASY WRITER, basta con teclear EW y pulsar la tecla de retroceso de carro para que dé comienzo la sesión. Inmediatamente aparecerá en la pantalla el "Easy Writer File System" y, a continuación, se darán los siguientes pasos:

1. Seleccionar el comando EDIT para "entrar" en el editor.
2. La pantalla aparecerá en blanco y en ella se tecleará el texto en el que se resumen las características del programa (ver cuadro adjunto).
3. A continuación se pulsará la tecla [F4] para que aparezca el menú de comandos adicionales.
4. Dentro del menú de comandos adicionales se seleccionará el comando PRINT TO SCREEN para apreciar el aspecto final del informe.
5. Posteriormente, se pulsará la tecla [F10] para volver al menú de gestión de ficheros.
6. Una vez situados en este menú, se pulsará la tecla [F2], con lo que inmediatamente se comenzará a imprimir el informe.
7. A continuación se accionará la tecla [S] para ejecutar el comando SAVE, y obtener así una copia del documento en la memoria auxiliar.
8. Para finalizar la sesión de trabajo se pulsará la tecla [X], con lo que el programa cederá el control al sistema operativo.

Juegos "a medida"

Campo minado

En este nuevo capítulo, dedicado a la práctica del lenguaje BASIC, se acometerá la confección, paso a paso, de un programa adecuado para ejecutar el tradicional juego del "campo minado".
Como apunta el nombre, el objetivo del juego es cruzar con éxito el peligroso trayecto, sorteando las minas ocultas que pueden hacer saltar a nuestro personaje por los aires.
Como instrumento de ayuda, el jugador dispone de un detector de minas que le informa, mediante un pitido, de la inmediata proximidad de un artefacto explosivo. Para cruzar el campo minado, el personaje puede desplazarse en cuatro sentidos: arriba, abajo, derecha e izquierda.
El proceso a seguir para el desarrollo del programa se expondrá con toda suerte de detalles en los próximos párrafos. El diagrama de flujo adjunto resume la estructura general del programa a confeccionar.

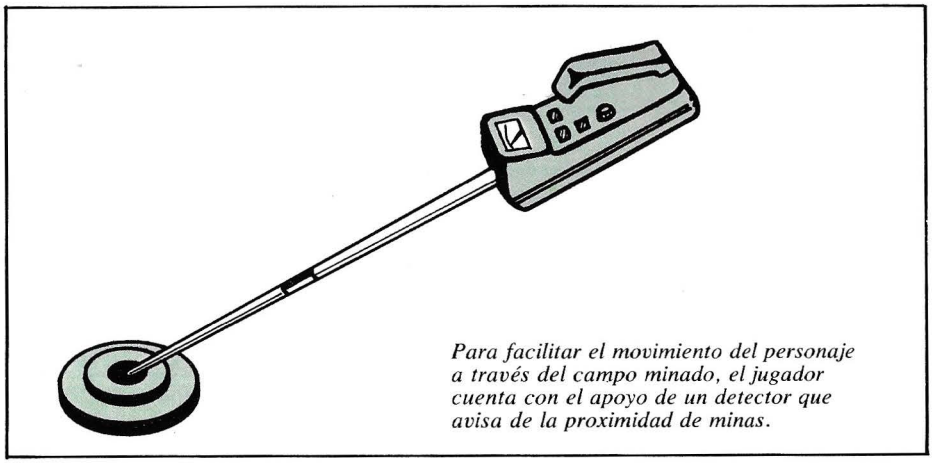
El objetivo del presente capítulo, dedicado a la práctica del BASIC, es confeccionar un programa que permita ejecutar el tradicional juego del "campo minado".

PREPARACION DE LA PANTALLA

Para empezar es conveniente preparar la pantalla de forma adecuada. Entre las operaciones que conducirán a tal preparación se encuentra el borrado de la pantalla y la selección del modo gráfico más idóneo.

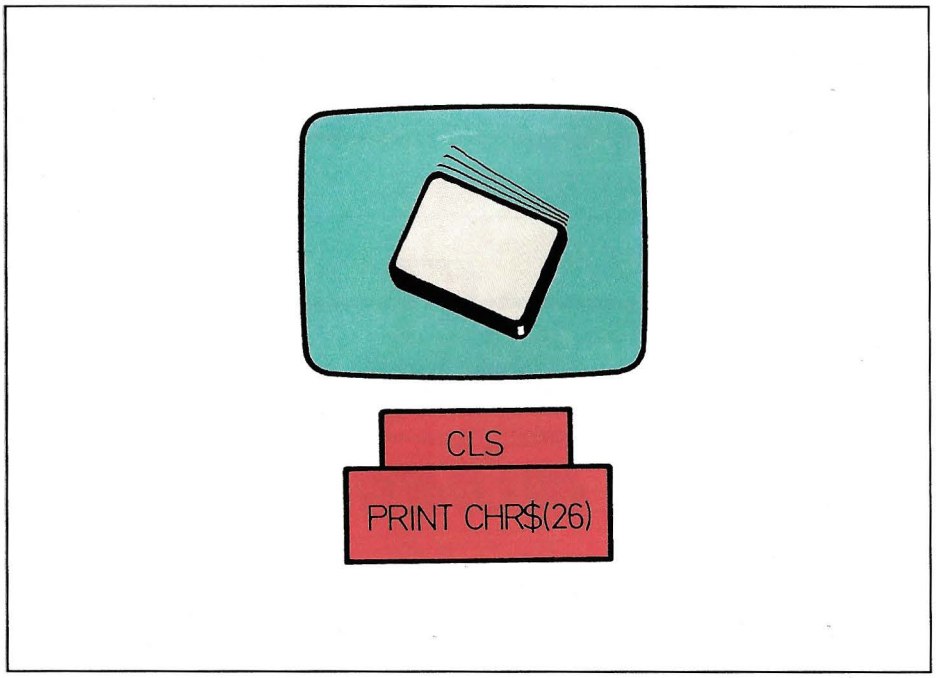
```
10 REM JUEGO DEL CAMPO MINADO
20 CLS:SCREEN 2
```

La línea 20 es la encargada de borrar la pantalla por medio del comando CLS. Evidentemente, en los ordenadores que no

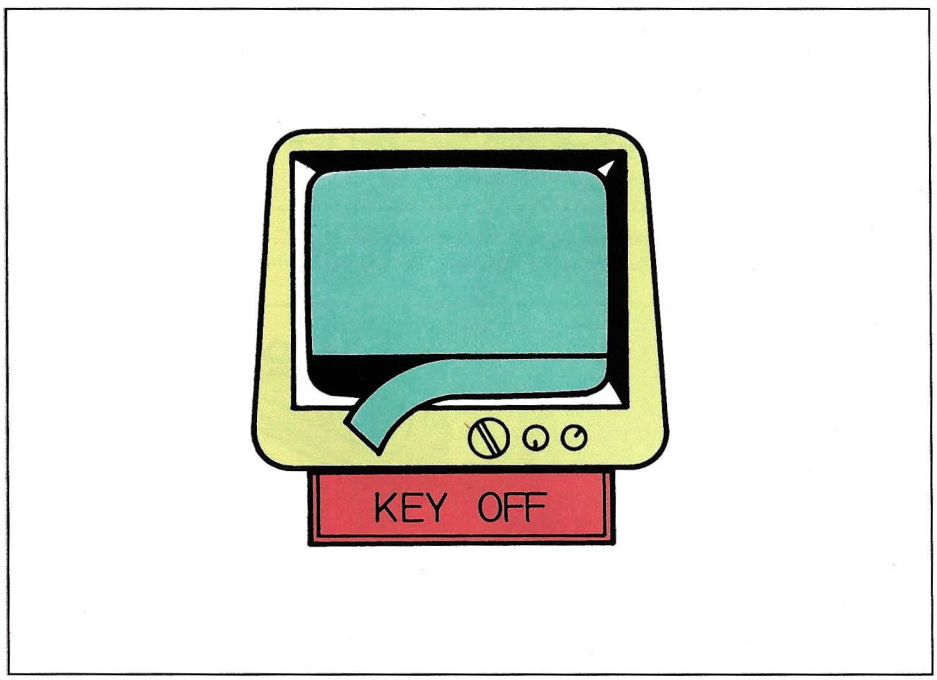


Para facilitar el movimiento del personaje a través del campo minado, el jugador cuenta con el apoyo de un detector que avisa de la proximidad de minas.

posean esta instrucción será necesario sustituirla por su equivalente. Al efecto, es conveniente recordar la existencia de los códigos de control, alguno de los cuales permitirá realizar esta misma acción.
Los códigos de control son propios de cada aparato; en consecuencia, no es posible precisar en estas páginas el código específico asociado a cada ordenador. No obstante, si cabe mencionar que uno de los más corrientes es el 26, de manera que ejecutando la siguiente instrucción:



La primera tarea a programar consiste en el borrado de la pantalla. Para ello puede hacerse uso del comando CLS, u otro equivalente en el dialecto BASIC del ordenador que se utilice, por ejemplo PRINT CHR$(26).



Si el ordenador utilizado es un IBM-PC o compatible, o un equipo MSX, será preciso ejecutar el comando KEY OFF para eliminar de la pantalla la franja inferior que revela las órdenes asociadas a las teclas de función.

```
PRINT CHR$(26)
```

se obtendría como resultado el borrado de la pantalla.
También es posible que no se emplee un único código para el borrado de la presentación visual, sino que el equipo exija la puesta en práctica de una combinación de códigos; por ejemplo:

```
PRINT CHR$(27)+"E"
```

En la propia línea 20 del programa también es posible seleccionar el modo gráfico adecuado para que en él se desarrolle el juego. En este caso, sí es totalmente obligado remitir al lector al manual de su ordenador, ya que las diferencias son drásticas de uno a otro equipo.
Si el equipo utilizado es un ordenador personal del tipo MSX, IBM-PC o compatible, será necesario en este punto eliminar de la pantalla la presentación que ocupa su zona inferior; presentación que recuerda los comandos disponibles en las teclas de función. Ello se logra, sencillamente, ejecutando la siguiente instrucción:

```
40 KEY OFF
```

Claro está que si esta franja de comandos no aparece en la pantalla de su ordenador, resultará inútil la inclusión de este comando.

MINADO DEL CAMPO

Para saber en qué puntos del campo se encuentran las minas, recurrimos al artificio de crear una matriz con tantos elementos como posibles casillas pueda recorrer nuestro personaje. Acto seguido, es preciso que el programa coloque las minas en el terreno en el que se va a desarrollar el juego, adoptando un criterio aleatorio:

```
 50 DIM A(31,21)
 60 REM COLOCACION DE LAS MINAS
 70 FOR I=1 TO 40
 80 X2=1+INT(RND*30)
 90 Y2=1+INT(RND*20)
100 IF X2=1 AND Y2=1 THEN GOTO 110
    ELSE A(X2,Y2)=1
110 NEXT I
```

Para el minado del campo se recurre al artificio de crear una matriz con tantos elementos como casillas pueda recorrer el personaje. Por supuesto, la colocación de las minas se realizará según un criterio aleatorio.

Esta rutina emplaza las minas de forma aleatoria a lo largo del escenario de juego. El número de minas depositadas se eleva a 40; sin embargo, difícilmente se podrán encontrar todas estas minas en el campo a cruzar, puesto que no se ha previsto ningún control para evitar que coincidan dos minas en el mismo lugar. Este hecho hará que aumente la aleatoriedad del juego.

La línea 50 se encarga de dimensionar la matriz A. La dimensión que realmente se va a utilizar se concreta en 30×20 elementos; no obstante, es preciso añadir un elemento más, por dimensión, para evitar inconvenientes que surgirán más adelante.

Las restantes instrucciones de la rutina definen un bucle que comienza en la línea 70 y termina en la 110. Este bucle colocará una mina en cada una de las pasadas; por supuesto, de forma aleatoria. Dentro del bucle las líneas 80 y 90 generan la posición de la mina; posición que se calcula a partir del número aleatorio entregado por la función RND. Dicho valor ha de someterse a un cambio de escala para que resulte un número comprendido dentro del margen 1 a 20 en el caso de Y2, y de 1 a 30 en el caso de X2.

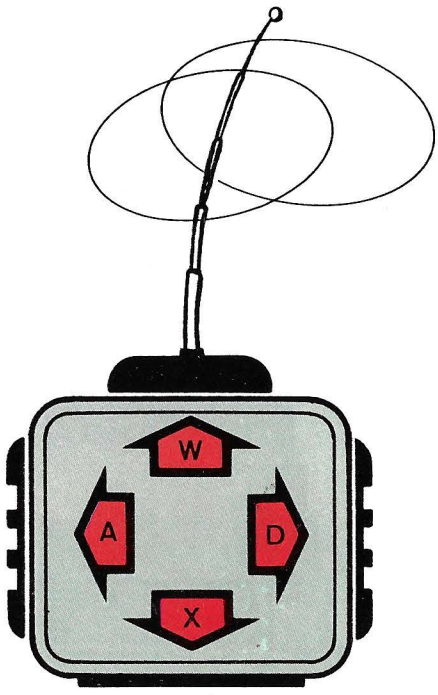
Por último, la línea 100 inspecciona la casilla de origen (1,1) para evitar que coloque en ella una mina. La referida línea 100 puede ocasionar dificultades en el caso de que el dialecto BASIC utilizado no incorpore la estructura IF/THEN/ELSE; en tal situación es recomendable cambiar la línea 100 por el par de instrucciones siguientes:

```
100 IF Y2=1 AND X2=1 THEN GOTO 110
105 A(X2,Y2)=1
```

Para delimitar el terreno de juego se trazará en la pantalla una línea que defina su perímetro. De ello se encargarán las instrucciones que siguen:

```
120 REM TRAZADO DEL CONTORNO
130 LINE (7,7)-(248,168),,B
```

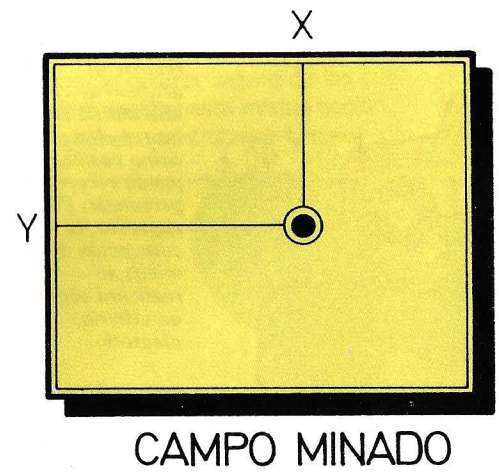
Esta última instrucción debe ser sustituida en cada caso por su equivalente en el ordenador que se utilice. En esencia, su misión es crear un rectángulo que delimite la superficie; la esquina superior izquierda de este rectángulo se encuentra en el punto 7,7 y la inferior derecha en el 248,168.



El movimiento del personaje a través del campo minado queda bajo el control de las teclas A, W, D y X.

DEFINICION Y MOVIMIENTO DEL PERSONAJE

Tras ello, resulta conveniente inicializar las variables que definen la posición del jugador dentro del campo. Estas variables, coincidentes con X e Y, contienen el valor actual de dicha posición. A su vez, las variables X1 e Y1 se emplean para conser-



Las variables X e Y contienen, respetivamente, las coordenadas horizontal y vertical de la posición que ocupa el personaje en el campo minado.

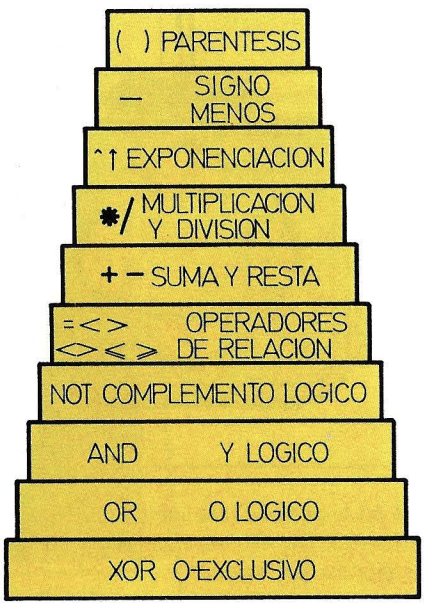
var el valor de la posición que ocupaba anteriormente el personaje. La inicialización de estas variables exige precisar cuál será la posición inicial del personaje.
Las instrucciones necesarias para esta operación pueden incluirse en una sola línea de programa, separando la inicialización de cada una de las variables por medio del signo dos puntos:

```
140 X=1 : Y=1 : X1=1 : Y1=1
```

El desplazamiento a través del rectángulo de juego ha de controlarse desde el teclado. Para ello es necesario incluir una serie de instrucciones que permitan al programa detectar las pulsaciones que efectúe el usuario sobre el teclado. La

Operadores lógicos

En BASIC existe un tipo de dato, denominado "lógico", cuya presencia es harto frecuente en el argumento del comando IF. En un capítulo precedente se vio que este comando era capaz de evaluar condiciones, entregando un resultado que podía ser "verdadero" o "falso". Pues bien, los valores que puede adoptar un dato de tipo lógico son precisamente los de "verdadero" y "falso".
Los datos lógicos se almacenan en el ordenador a modo de datos numéricos. Por regla general, se asocia al valor de "cierto" el dato numérico −1 y al de "falso" el 0. De esta forma se hace posible el almacenamiento de un valor lógico en el seno de una variable numérica normal.
Los datos lógicos se obtienen a partir de la evaluación de condiciones, o como resultado de ciertas funciones, y su empleo más común se encuentra en el control del flujo de ejecución (habitualmente a través de un comando IF). En todo caso, el uso de datos lógicos no se limita a este cometido; también se puede operar con datos de este tipo, de forma parecida a la operación con datos numéricos o alfanuméricos.
Así pues, dos o más datos lógicos pueden ser agrupados para producir un resultado también de tipo lógico. Para entender el significado de los operadores lógicos es preciso introducir algunos conceptos generales de lógica.
Tras evaluar una proposición puede deducirse que la misma es "verdadera" o "falsa". Por proposición se entiende cualquier tipo de expresión que afirma un hecho. Por ejemplo, "Madrid es la capital de Italia" es una proposición falsa.
Como ya se ha indicado, pueden



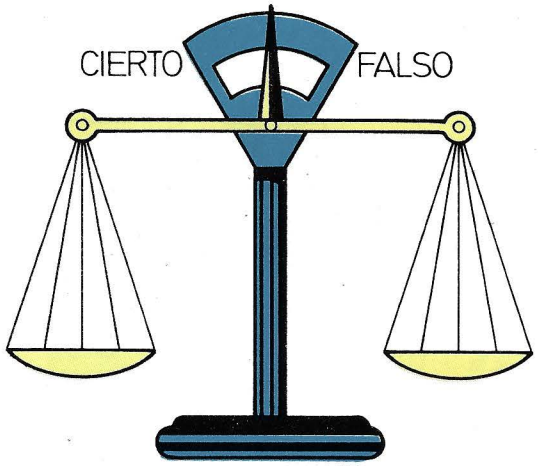
Jerarquía de las operaciones aritméticas, lógicas y de relación programables en BASIC.

agruparse más de una proposición sencilla para obtener una proposición compuesta. Así: "Madrid es la capital de Italia o de España" es una proposición compuesta por las simples: "Madrid es la capital de Italia" y "Madrid es la capital de España". En este caso, el nexo de unión de ambas se materializa en la palabra "o".
La proposición compuesta puede ser evaluada a partir del resultado de las simples. En el ejemplo, se tiene una proposición falsa y otra verdadera. Intuitivamente, se deduce que la proposición compuesta será verdadera cuando lo sea una de las simples (cuando lo sea la primera **o** lo sea la segunda). Es decir, basta que una de ellas sea cierta para que el total sea verdadero. Ello se debe al uso de la conjunción "o".
Por el contrario, si se unen las proposiciones con la conjunción "y" la cosa es bien distinta. La unión por medio de "y" determina que la veracidad de la proposición compuesta tiene lugar únicamente cuando ambas son ciertas (cuando lo es la primera **y** la segunda).
Generalizando, se puede decir que las palabras **o** e **y** agrupan dos datos lógicos y proporcionan un tercero que se deriva del valor de aquellos. O dicho de otra forma, **o** e **y** son dos operadores lógicos.
Para identificar con claridad el resultado de cada uno de estos operadores, se hace uso de las denominadas tablas de verdad.
Una tabla de verdad no es más que la representación de los resultados de cada operador para todas las combinaciones posibles de sus operandos. A modo de ejemplo, junto al texto se incluyen las tablas de verdad de los comentados operadores lógicos. En ellas se reflejan los resultados de operar los valores de entrada situados en las dos primeras columnas.
Existe un tercer operador lógico que corresponde a la "negación". Negar una proposición supone invertir su valor lógico: la negación de "hoy es jueves"

función BASIC más adecuada para este cometido es INKEY$, puesto que no exige pulsar la tecla de retorno de carro para que el ordenador entienda que el mensaje ha de ser aceptado y analizado:

```
160 B$=INKEY$: IF B$="" " THEN GOTO 160
```

Por supuesto, la próxima operación a realizar consiste en analizar lo que el operador ha introducido. Llegados a este punto hay que estudiar qué teclas se utilizarán para ordenar los movimientos a través del escenario de juego. Parece conveniente en este caso elegir teclas distribuidas de una forma lógica; por ejemplo, D, A, X y W. El siguiente bloque de instrucciones se encargarán de actualizar los valores de X e



A lo largo del programa se toman abundantes decisiones basadas en la evaluación de condiciones impuestas. Si la condición se verifica, el resultado obtenido es cierto (−1); mientras que si ésta no se cumple se obtendrá el resultado falso (0).

será, pues, "hoy **no** es jueves". A la vista del ejemplo se observan dos cosas: el operador **"no"** actúa sobre un único operando, y su función es la de cambiar verdadero por falso y viceversa. Su tabla de verdad coincide con la que aparece al margen.

Los operadores lógicos del BASIC se formulan recurriendo a la traducción inglesa de las mismas palabras españolas: AND (y), OR (o) y NOT (no). Por lo demás, funcionan tal y como se ha descrito más arriba.

El resultado de una operación lógica será un dato lógico y, por lo tanto, adecuado para el uso en el argumento de un comando IF.

Veamos a continuación un ejemplo de su uso dentro de un programa BASIC. El objetivo del programa es mostrar un mensaje de error cuando sea tecleado un número comprendido entre 10 y 20. El fragmento de programa que realiza tal acción es el siguiente:

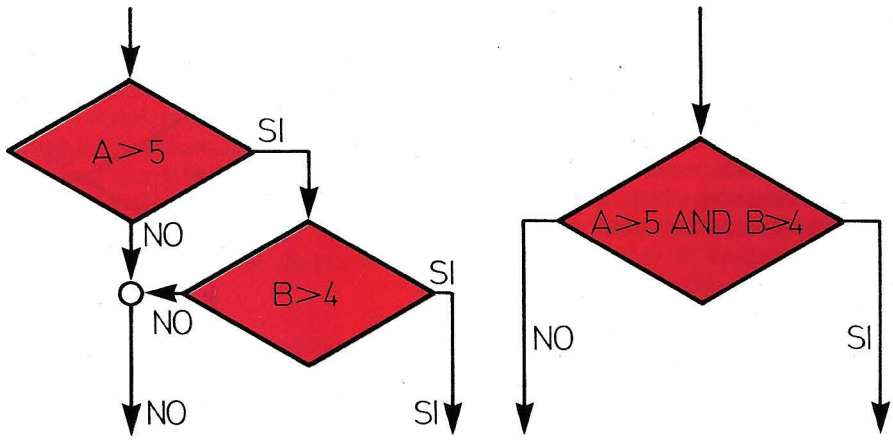
```
...
100 INPUT A
110 IF (A<20) AND (A>10) THEN PRINT
    "ESE NUMERO NO VALE"
...
```

En el argumento se ha formulado una proposición compuesta por A<20 y A>10. Como en este caso ambas condiciones se han unido por medio del operador AND (y), la proposición conjunta sólo será verdadera cuando se cumplan los dos términos: A menor que 20 **y** mayor que 10.

Al igual que ocurre con los operadores aritméticos (+, −, *, etc.), los operadores lógicos pueden encadenarse para construir expresiones complejas. Por ejemplo, si a los números erróneos del ejemplo anterior se desean unir también los comprendidos entre 50 y 80, bastaría con introducir lo siguiente:

```
...
100 INPUT A
110 IF (A<20 AND A>10) OR (A<80 AND
    A>50) THEN PRINT "ESE NUMERO NO
    VALE"
...
```

En este segundo ejemplo se han utilizado los resultados de los dos AND como operandos de la relación OR. Con ello se consigue la visualización del mensaje cuando se cumpla la primera **o** la segunda de las condiciones.



La combinación de condiciones *mediante operadores lógicos permite crear estructuras de decisión evolucionadas, definiendo condiciones múltiples en una sola línea de programa.*

TABLA DE FUNCIONES LOGICAS

DATOS		FUNCIONES LOGICAS			
A	**B**	**A y B**	**A o B**	**no A**	**no B**
verdadero	verdadero	verdadero	verdadero	falso	falso
verdadero	falso	falso	verdadero	falso	verdadero
falso	verdadero	falso	verdadero	verdadero	falso
falso	falso	falso	falso	verdadero	verdadero
		AND	**OR**	**NOT**	

Y en función de la tecla que se pulse en cada instante:

```
170 IF X<30 THEN X=X-(B$="D")
180 IF X>1 THEN X=X+(B$="A")
190 IF Y<20 THEN Y=Y-(B$="X")
200 IF Y>1 THEN Y=Y+(B$="W")
```

Es muy posible que las operaciones situadas tras cada condición resulten un tanto extrañas. De hecho, no es muy ortodoxo mezclar la evaluación de condiciones con instrucciones aritméticas; sin embargo, ello está permitido.
La evaluación de cada condición impuesta produce un resultado igual a cero si la condición es falsa, e igual a −1 cuando la condición resulta ser cierta. En este caso se aprovecha directamente el resultado de evaluar la condición para producir las modificaciones necesarias en las coordenadas que definen la posición del personaje.
Para controlar el número de movimientos que se realizan hasta terminar el juego se emplea una variable contadora: CON. Su valor se inicializa a cero en la línea 30 y se incrementa sucesivamente tras cada movimiento:

```
210 CON=CON+1
```

El siguiente paso consiste en colocar en la pantalla a nuestro personaje, representado por un asterisco, y borrar el asterisco de la posición que ocupaba anteriormente.

```
220 LOCATE Y1+1, X1+1
230 PRINT " "
240 LOCATE Y+1, X+1
250 PRINT "*"
```

Las instrucciones de las líneas 220 y 240 trasladan el cursor a la posición de pantalla adecuada, ya sea para imprimir un espacio en blanco encima de la posición que el personaje ocupaba anteriormente, o bien para visualizar el asterisco en la posición actual. La posición que anteriormente ocupaba el asterisco en la pantalla se almacena en las variables X1 e Y1. Para ello se asignan a estas variables los valores de X e Y, antes de actualizar estos últimos. Así, tras la actualización de X e Y, X1 e Y1 conservarán el contenido precedente de las variables X e Y. La instrucción adecuada para realizar esta asignación es la siguiente:

```
260 X1=X : Y1=Y
```

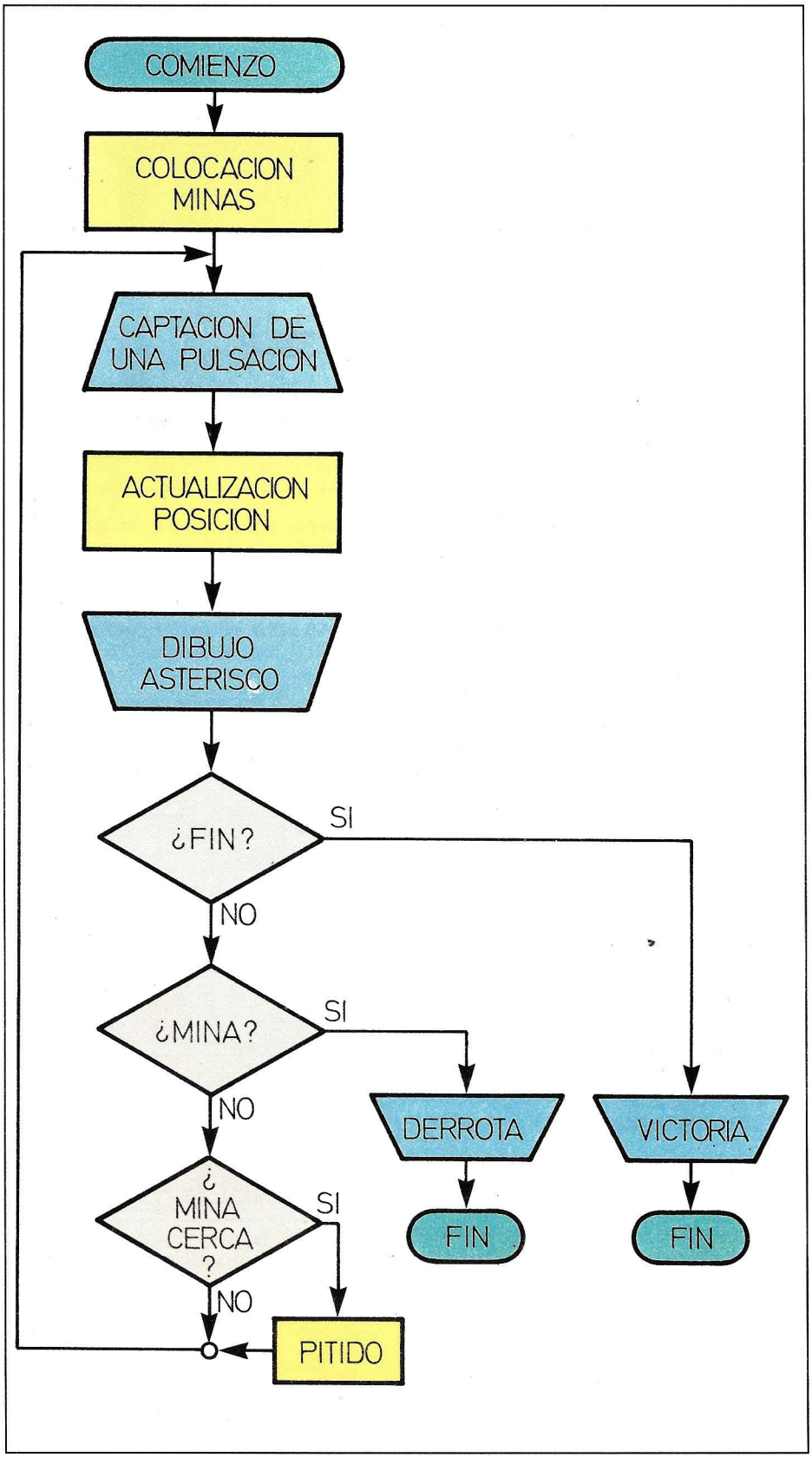
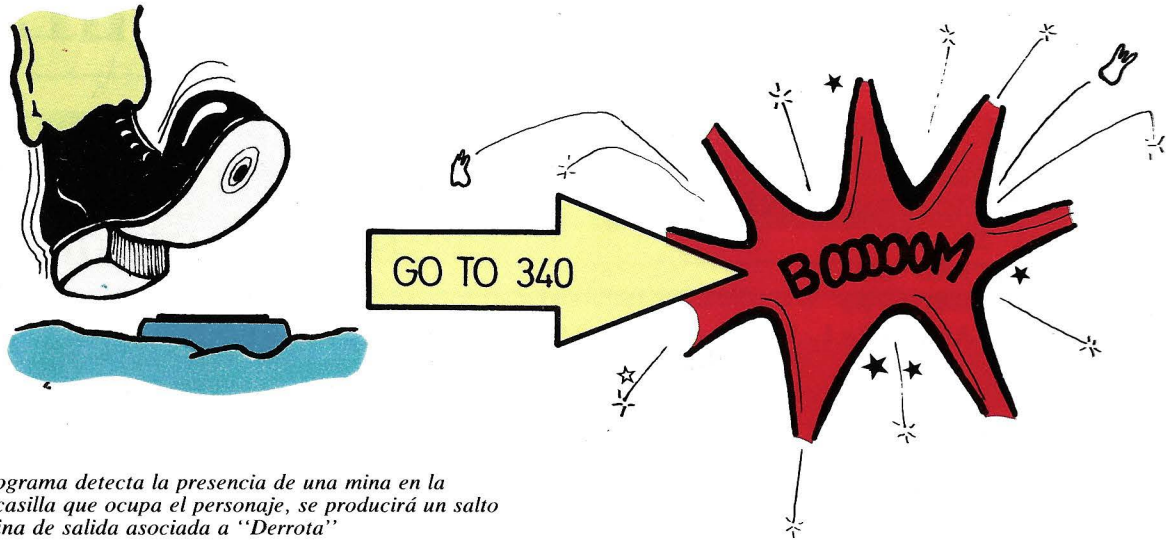


Diagrama de flujo correspondiente al programa "campo minado".



Si el programa detecta la presencia de una mina en la misma casilla que ocupa el personaje, se producirá un salto a la rutina de salida asociada a "Derrota"

¿EXITO O EXPLOSION?

Tras actualizar la posición, la próxima tarea que debe ocupar al programa es comprobar a dónde lleva el movimiento realizado. Al efecto, es necesario efectuar una serie de comprobaciones y tomar las oportunas medidas dependiendo de los resultados que se obtengan:

```
280 IF X=30 AND Y=20 THEN GOTO 360
```

La línea 280 comprueba si se ha alcanzado la esquina opuesta del campo minado, cuyas coordenadas son X=30 e Y=20. Si se ha llegado a dicho punto se producirá una bifurcación hacia la línea 360, donde se encuentra la rutina adecuada para tratar este supuesto.

```
290 IF A(X,Y)=1 THEN GOTO 340
```

Por su parte, en la 290 se comprueba si la posición que ocupa el personaje en este preciso instante coincide con una de las minas. De ser cierto el juego habrá terminado con una fatal consecuencia. De inmediato se bifurcará a una rutina situada a partir de la línea 340, la cual imprimirá el oportuno mensaje antes de poner fin al programa:

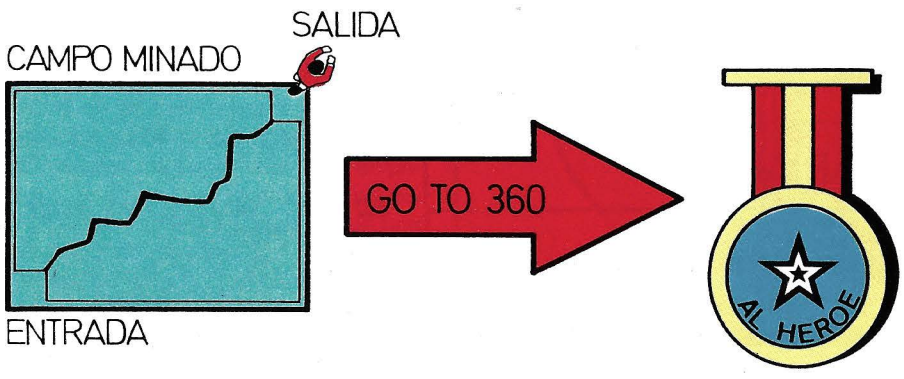
```
310 IF A(X+1,Y)=1 OR A(X-1,Y)=1 OR
    A(X,Y-1)=1 OR A(X,Y-1)=1 THEN BEEP
```

Esta línea de programa es la encargada de simular la actuación del detector de proximidad de minas. Genera un sonido a través del altavoz en el caso de que se encuentre una mina en las proximidades de posición que ocupa el personaje.
Por último, tan sólo queda retornar al comienzo de la rutina encargada de actualizar la posición:

```
320 GOTO 160
```

Al programa le faltan únicamente las dos rutinas de fin del juego: una a ejecutar en el caso de cruzar con éxito el campo minado, y otra que se ejecutará cuando el personaje tropiece con una de las minas. La rutina que tomará el control en caso de percance se reduce a dos líneas de programa:

```
340 PRINT "SE ACABO, HAS PERDIDO"
350 END
```



Si la travesía culmina con éxito, el programa bifurcará al punto de salida (GOTO 360) "victoriosa", reflejándose en la pantalla el número de movimientos realizados.

```
10 REM JUEGO DEL CAMPO MINADO
20 CLS:SCREEN 2
30 CON=0
40 KEY OFF
50 DIM A(31,21)
60 REM COLOCACION DE LAS MINAS
70 FOR I=1 TO 40
80 X2=1+INT(RND*30)
90 Y2=1+INT(RND*20)
100 IF X2=1 AND Y2=1 THEN GOTO 110 ELSE A(X2,Y2)=1
110 NEXT I
120 REM TRAZADO DEL CONTORNO
130 LINE (7,7)-(248,168),,B
140 X=1:Y=1:X1=1:Y1=1
150 REM ACTUALIZACION DE LA POSICION
160 B$=INKEY$:IF B$="" THEN GOTO 160
170 IF X<30 THEN X=X-(B$="D")
180 IF X>1 THEN X=X+(B$="A")
190 IF Y<20 THEN Y=Y-(B$="X")
200 IF Y>1 THEN Y=Y+(B$="W")
210 CON=CON+1
220 LOCATE Y1+1,X1+1
230 PRINT " "
240 LOCATE Y+1,X+1
250 PRINT "*"
260 X1=X:Y1=Y
270 REM COMPROBACION DE LA POSICION CON RESPECTO A LAS MINAS
280 IF X=30 AND Y=20 THEN GOTO 360
290 IF A(X,Y)=1 THEN GOTO 340
300 REM COMPROBACION DE PROXIMIDAD DE MINAS
310 IF A(X+1,Y)=1 OR A(X-1,Y)=1 OR A(X,Y+1)=1 OR  A(X,Y-1)=1 THEN BEEP
320 GOTO 160
330 REM RUTINAS DE FIN DEL JUEGO
340 PRINT "SE ACABO, HAS PERDIDO"
350 END
360 PRINT "GANASTE TRAS ";CON;" INTENTOS"
370 END
```

TABLA DE VARIABLES

A()	Matriz que almacena la posición de las minas.	I	Indice de bucle.
B$	Variable que contiene el carácter pulsado.	X,Y	Posición actual del personaje.
CON	Contador de movimientos.	X1,Y1	Posición anterior del personaje.
		X2,Y2	Coordenadas para la generación de las minas.

al igual que la asociada a un final victorioso:

```
360 PRINT "GANASTE TRAS ";CON;"
    INTENTOS"
370 END
```

Junto al texto se reproduce el listado completo del programa, salpicado por algunos comentarios (REM) que facilitarán la asimilación del mismo.

Forth (9)

Operación con números enteros y de punto flotante

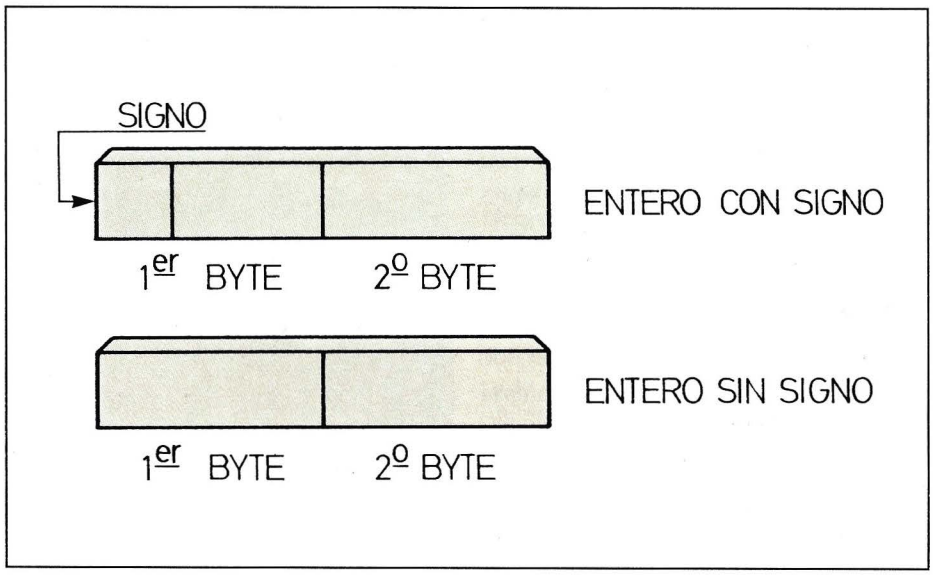
En el capítulo precedente se definieron los distintos tipos de números que admiten la mayor parte de las versiones del lenguaje FORTH. Ahora da comienzo el estudio pormenorizado de cada uno de estos tipos, empezando por los números enteros, con o sin signo, expresados en simple precisión.

NUMEROS ENTEROS

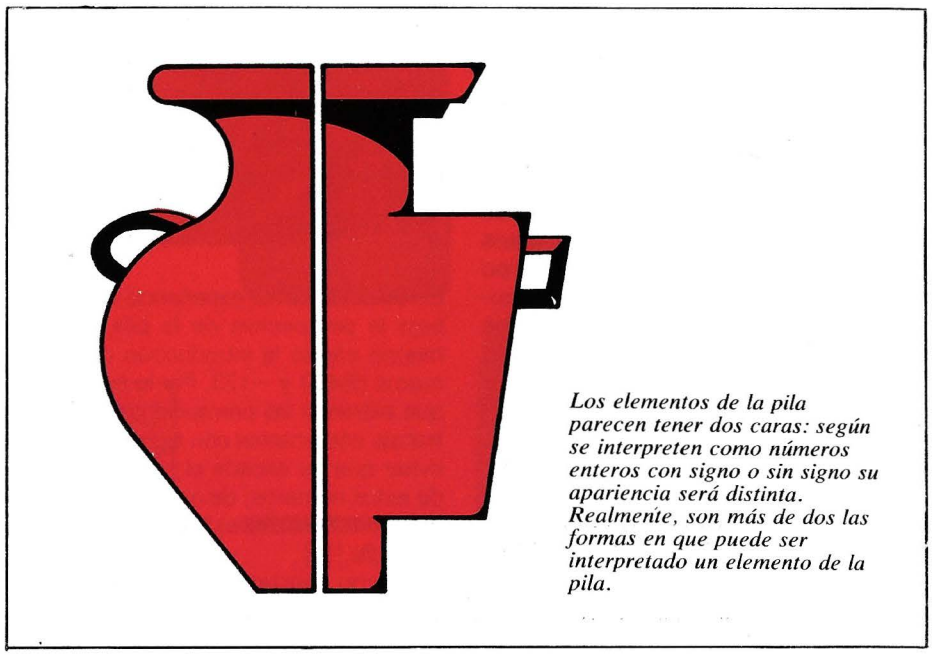
Los números enteros, con o sin signo, se almacenan por medio de dos bytes, lo que se traduce en la posibilidad de representar un total de 65536 combinaciones distintas. En el caso de los números enteros, estas combinaciones han de utilizarse para almacenar tanto los números con signo positivo como sus correspondientes valores negativos. La diferenciación entre unos y otros se establece almacenando los de signo positivo en binario, tal cual, mientras que los negativos se memorizan en complemento a dos.

Tal vez sea conveniente recordar que la expresión en complemento a dos se obtiene traduciendo el número a binario, complementando a uno su configuración (cambiar los ceros por unos y los unos por ceros) y sumando una unidad al resultado así obtenido.

La notación en complemento a dos permite saber si un número es positivo o negativo examinando simplemente su primer bit: si es un cero se trata de un número positivo, mientras que si es un uno el número será negativo. Por supuesto, si se prescinde del signo, el bit reservado al efecto quedará libre y, en consecuencia, dicho bit pasará a ocupar la posición de peso superior, permitiendo representar números de mayor magnitud.



Representación gráfica de la forma de almacenamiento de los números enteros con o sin signo.



Los elementos de la pila parecen tener dos caras: según se interpreten como números enteros con signo o sin signo su apariencia será distinta. Realmente, son más de dos las formas en que puede ser interpretado un elemento de la pila.

ENTEROS SIN SIGNO

Cuando se trata de trabajar con valores enteros comprendidos dentro del rango que va de 0 a 65535, se puede recurrir al tipo denominado: números enteros sin signo. Este tipo de notación resultará muy útil cuando no se contemple la posibilidad de operar con números negativos, puesto que con los mismos 2 bytes, es posible almacenar números de magnitud muy superior a la que permiten los enteros con signo.
Para visualizar un número entero sin signo es necesario emplear una palabra FORTH cuya denominación se reduce a un solo carácter: U. Veamos su efecto:

```
123 U. <CR>
```

Una vez ejecutada la orden, la pantalla mostrará el siguiente aspecto:

```
123 U. 123 OK
```

Una característica inherente a la pila es que en ella se almacenan los elementos uno encima del otro, sin distinción del tipo numérico asociado al dato que se deposita. Ello significa que el resultado será distinto dependiendo del modo de acceso y de la palabra empleada para visualizar los elementos de la pila.
Supongamos que se utiliza la instrucción:

```
-123 U. <CR>
```

Ello equivale a introducir en la pila un número con signo negativo, número que se almacenará en la misma en complemento a dos; y, acto seguido, se ordena su visualización como entero sin signo. La respuesta en pantalla será bastante curiosa:

```
-123 U. 65413 OK
```

Desde luego, también cabe realizar la operación contraria:

```
65413. <CR>

65413. -123 OK
```

El resultado de tal experiencia revela que, bajo la perspectiva de la pila, tiene un mismo efecto la introducción de los números 65413 y −123. Por lo tanto, habrá que extremar las precauciones cuando se trabaje con enteros con signo, en orden a evitar que se exceda el rango de validez de estos números; de ocurrir tal situación, se cometerán errores que el ordenador no advertirá.
Un hecho constatable es que la suma de un número en complemento a dos con otro, equivale a la resta de ambos números. Por ejemplo, el resultado de sumar los números 65413 y 123 —uno es el complemento a dos del otro— será igual a cero.
Veamos un nuevo ejemplo:

```
-123 56+DUP . U. <CR>
```

Esta cadena de palabras FORTH ordena a la máquina que calcule la suma de los números −123 y 56, y que muestre el resultado en pantalla como número entero con signo y como entero sin signo; obviamente, para realizar esto último es necesario duplicar previamente la cima de la pila.
Aquí está la respuesta:

```
-123 56+DUP . -67 U. 65469 OK
```

El primer resultado (−67) es el lógico, mientras que el segundo es la consecuencia de acceder a la pila con una palabra

Los números en punto flotante se almacenan en memoria codificados en cuatro bytes y, en consecuencia, ocupan dos elementos de la pila. Ello significa que para acceder a ellos no basta con tomar un solo elemento de la pila.

La zona de exponentes define la correcta situación del punto decimal dentro de un número expresado en punto flotante.

inadecuada. Para evitar errores de esta índole, conviene tener presente en todo momento qué tipo de números se encuentra en la pila.
El que sigue es un nuevo ejemplo. Ahora se depositan en la pila dos números enteros sin signo; en este caso, el resultado equívoco se producirá al visualizar el elemento de la pila como número entero sin signo.

```
65413 56+DUP . U. <CR>

65413 56+DUP . -67 U. 65469 OK
```

Cabe observar que las instrucciones aritméticas aplicables a los enteros con signo son también válidas para los números enteros sin signo, a condición de que sean visualizados en pantalla con la orden correcta. Para los enteros sin signo no es válida la palabra de comparación <, si bien, existe otra palabra alternativa perfectamente utilizable: U<.

NUMEROS EN PUNTO FLOTANTE

Los números en punto flotante ("coma flotante" en el caso español, que recurre a la coma en lugar del punto decimal) se almacenan en memoria mediante cuatro bytes; de ellos, tres se emplean para almacenar un valor numérico que puede ser positivo o negativo. Este número corresponde a las seis cifras significativas o mantisa del número en cuestión, mientras que el cuarto byte se reserva para almacenar la potencia de 10 por la que hay que multiplicar la mantisa para obtener el valor representado por el conjunto.

La mantisa es un número racional de punto fijo, que se supone con el punto decimal situado a la izquierda de su primera cifra.
Para ilustrar esta idea se muestran seguidamente algunos ejemplos de representación de números en punto flotante:

25.23 **Mantisa: 0.2523**
Exponente: 2

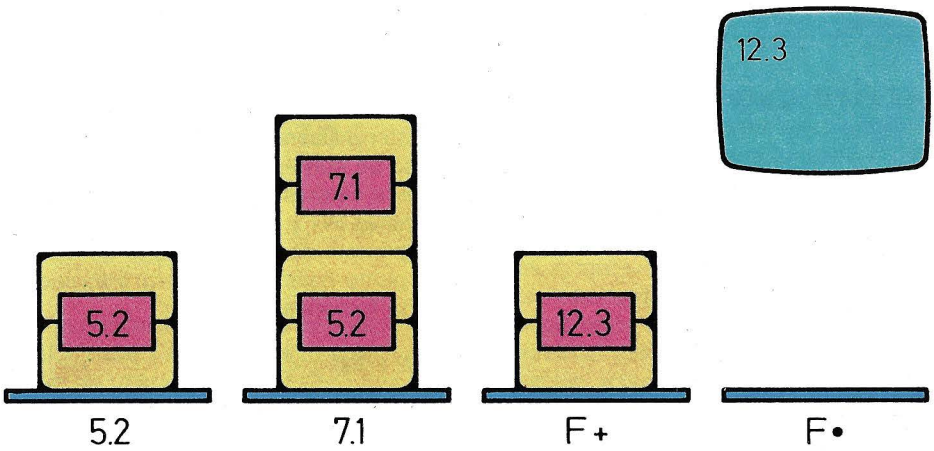
1234.0 **Mantisa: 0.1234**
Exponente: 4

Para trabajar con este tipo de números, el lenguaje FORTH brinda un repertorio de palabras especiales. En primera instancia cabe hablar de la palabra F, destinada a visualizar un número de punto flotante.
Veamos qué sucede cuando se intenta visualizar un número de este tipo depositado en la pila mediante la palabra ".":

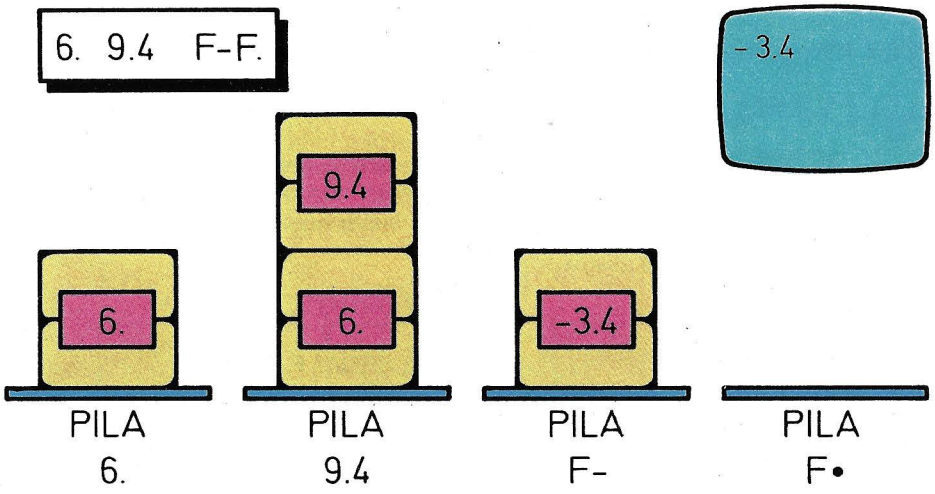
```
5.2 . . <CR>
```

(La presencia de dos puntos consecutivos obedece a que los números de punto flotante ocupan dos elementos de la pila).
Ahí está la respuesta:

```
5.2 . 16722 . 0 OK
```



Evolución de la pila al ejecutar la operación 5.2+7.1 y visualizar el resultado.

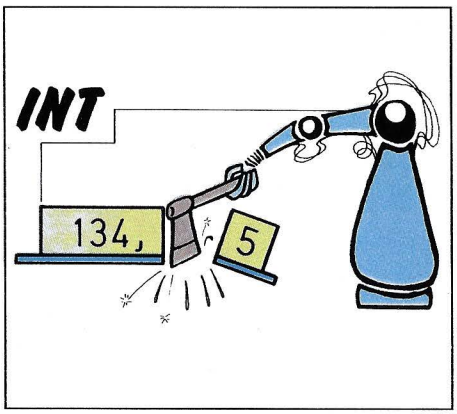


Manipulaciones de la pila asociadas a la operación: 6−9.4.

Por contra, si se utiliza la palabra adecuada, el resultado será bien distinto:

```
5.2 F. <CR>

5.2 F. 5.2 OK
```

TABLA DE ORDENES FORTH		
PALABRA	DESCRIPCION	TIPO
U.	Muestra por pantalla un número entero sin signo extraído de la cima de la pila.	Palabra de E/S
F.	Visualiza un número de punto flotante constituido por los dos elementos más cercanos a la cima de la pila.	Palabra de E/S
F+	Suma dos números de punto flotante, tomando los elementos más próximos a la cima de la pila.	Palabra aritmética
F−	Resta los dos números de punto flotante que se encuentran en las posiciones superiores de la pila.	Palabra aritmética
F*	Multiplica dos números de punto flotante.	Palabra aritmética
F/	Divide dos números de punto flotante.	Palabra aritmética
FNEGATE	Cambia el signo de un número de punto flotante.	Palabra aritmética
INT	Convierte un número de punto flotante a número entero, truncando la parte fraccionaria.	Conversión de tipo
UFLOAT	Tranforma un número entero en número de punto flotante.	Conversión de tipo



La palabra INT permite el paso de número en punto flotante a número entero, truncando para ello la parte fraccionaria.

Además de la palabra de visualización F, el vocabulario FORTH cuenta con un surtido adicional de palabras destinadas a la ejecución de operaciones aritméticas.

F+ Suma los dos números en punto flotante situados en las posiciones superiores de la pila, y deposita el resultado en la cima de la pila. Así, por ejemplo, el resultado de ejecutar la orden:

5.2 7.1 F+F. <CR>

coincidirá con:

5.2 7.1 F_F. 12.3 OK

F− Resta el segundo número en punto flotante introducido en la pila del introducido en primer lugar; el resultado se deposita en la cima de la pila.

6. 9.4 F−F. <CR>

6. 9.4 F−F. −3.4 OK

F* Multiplica los dos números de punto flotante situados más cerca de la cima de la pila y deposita el resultado en la misma.

2.1 1. F*F. <CR>

2.1 1. F*F. 2.1 OK

F/ Obtiene la división de los dos números de punto flotante situados en las posiciones superiores de la pila, y vierte el resultado en la cima de la misma.

3.4 5. F/F. <CR>

3.4 5. F/F. .68 OK

FNEGATE Cambia el signo del número en punto flotante que se encuentre en la posición superior de la pila.

12. FNEGATE F. <CR>

12. FNEGATE F. −12. OK

Todas las operaciones descritas actuarán tomando los elementos que se encuentren más cerca de la cima de la pila y considerándolos como números de punto flotante. Esta es una precisión muy a tener en cuenta en el caso de ejecutar operaciones aritméticas sobre números de distinto tipo.
Los números en notación científica son también de punto flotante, tal y como muestra el siguiente ejemplo:

4.7E1 4.7E3 F+F. <CR>

4.7E1 4.7E3 F+F. 4747. OK

CONVERSION DE NUMEROS DE UN TIPO A OTRO

Existen palabras FORTH que permiten el paso de números en punto flotante a números enteros y viceversa; éstas son las palabras INT y UFLOAT.
La primera de ellas convierte un número de punto flotante en entero, truncando la parte fraccionaria:

5. 2. F/INT . <CR>

3. 2. F/INT . 2 OK

A su vez, UFLOAT transforma un número entero en número de punto flotante:

2 UFLOAT 2.3 F+F. <CR>

2 UFLOAT 2.3 F+F. 4.3 OK

Esta última palabra presenta ciertos problemas, conducentes a error, cuando actúa sobre números negativos; por ejemplo:

−12 UFLOAT F. <CR>

−12 UFLOAT F. 65524 OK

En próximos capítulos se estudiarán los números de doble precisión, presentando las técnicas a aplicar para su visualización en el formato adecuado.

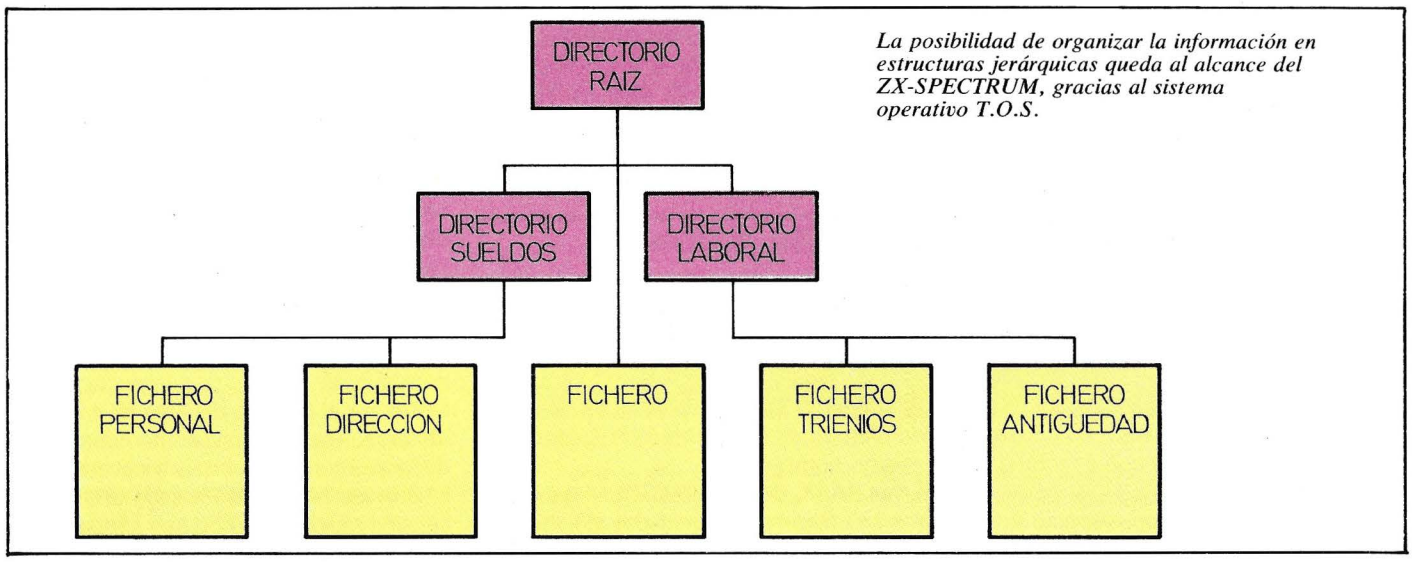
T.O.S. (y 2)

Localización y envío de información en el sistema operativo TOS

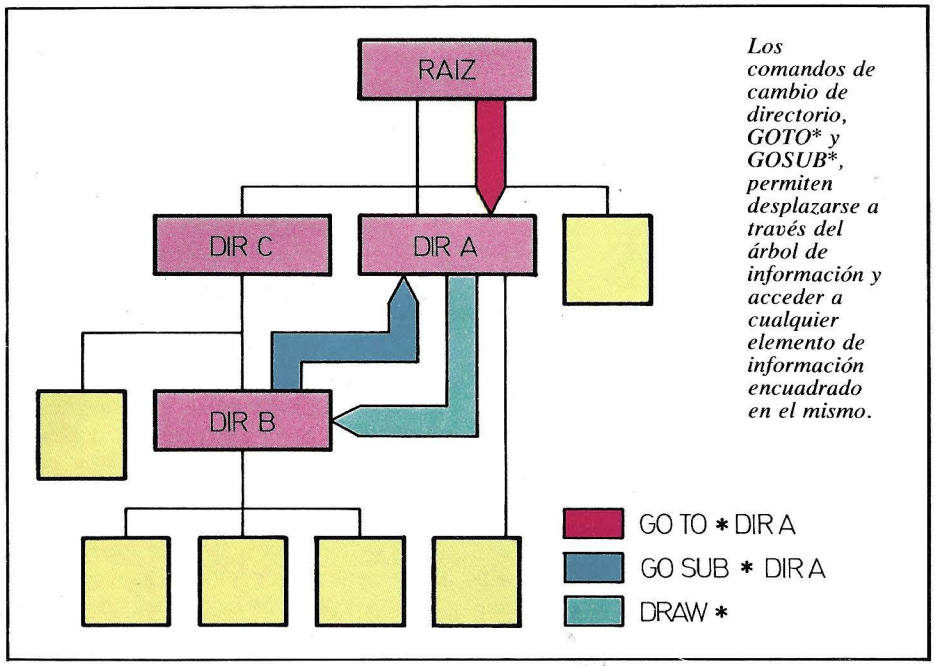
Cualquier sistema operativo que se precie, ha de disponer de una estructura eficiente que permita acceder a la información de una forma rápida y que economice los recursos del sistema; además, debe facilitar al usuario el acceso a dicha información con claridad y sencillez.
El sistema operativo TOS (Timex Operating System) lleva a cabo este cometido organizando la información en una estructura jerárquica: la estructura en árbol invertido.

LOS ARBOLES DEL T.O.S

El espectacular aumento de la cantidad de información accesible al usuario que deriva de utilizar el disquete en lugar de las clásicas cintas magnéticas, en casete o en microdrive, revierte en la necesidad de ordenar dicha información para que su acceso no se convierta en una ardua tarea. El sistema operativo TOS ha resuelto el tema optando por las estructuras arborescentes.
Una estructura en árbol invertido se caracteriza por la existencia de una única raíz (directorio raíz), a partir del cual crecen ramas (directorios secundarios o subdirectorios) y hojas (ficheros); e incluso cabe la posibilidad que de una rama crezcan nuevas ramas y hojas, en lugar de llevar directamente a hojas.
La ventaja de estructurar la información de esta forma reside en la agrupación de programas y datos en distintos directorios (ramas), de tal modo que el nombre del directorio en el que estén contenidos ayude a su identificación y evite así errores innecesarios.
Es importante establecer una clara diferencia entre directorios y ficheros. Los directorios contienen esencialmente información acerca de los ficheros que están bajo su dominio, al igual que el índice de un libro contiene información relativa a sus diferentes capítulos; mientras que en el interior de los ficheros es donde se reúne la información.
Bajo esta estructura es posible la existencia de ficheros con el mismo nombre en distitos directorios; de otro modo su acceso sería problemático, puesto que no cabría la opción de distinguir uno de otro.
Tal distinción se consigue en las estructuras de esta índole utilizando caminos para referenciar a los diversos ficheros contenidos en el árbol. Los caminos están formados por la concatenación de los nombres de sucesivos directorios por los que hay que transitar para llegar al fichero, además del propio nombre del referido fichero.
Existen algunas reglas sintácticas a observar; éstas son:
— Los nombres de los directorios han de ir separados por dos puntos (:).
— El directorio raíz se representa por dos puntos (:).
Nombres de caminos son:
" CALCULO VIGAS : IPN.BAS"
"VIGAS H BAS"



La posibilidad de organizar la información en estructuras jerárquicas queda al alcance del ZX-SPECTRUM, gracias al sistema operativo T.O.S.



Los comandos de cambio de directorio, GOTO y GOSUB*, permiten desplazarse a través del árbol de información y acceder a cualquier elemento de información encuadrado en el mismo.*

CAMINANDO POR EL ARBOL

Los cambios de uno a otro directorio se realizan en el TOS con la colaboración de los siguientes comandos:

GO TO*

Posicionamiento en el directorio indicado. Existen también movimientos relativos de directorios, expresandose estos con flechas (↑) Asi, por ejemplo GO TO* " ↑ " sube un nivel en el árbol. Pueden utilizarse varias flechas; en cuyo caso se ascenderá tantos niveles de directorio en el árbol como flechas se especifiquen.
Formato:

GO TO* nombre-camino

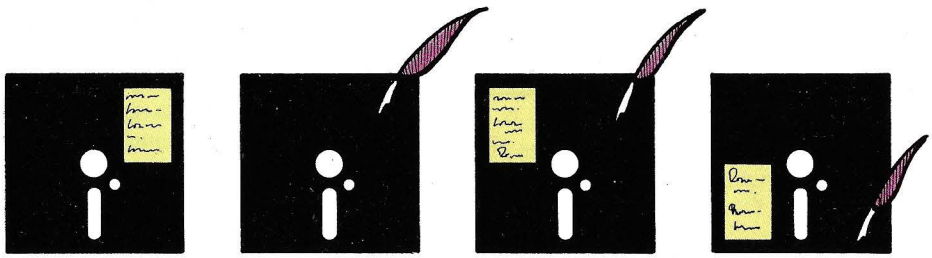
LIST*

Señala el lugar del árbol en el que nos encontramos actualmente.

GO SUB* y **DRAW***

Permiten el posicionamiento en un directorio, recordando desde qué directorio se operó el salto.
La ventaja de su uso con respecto a GO TO* reside en que, una vez que se han realizado las operaciones pertinentes en el directorio accedido, la vuelta al directorio anterior es automática con el uso de DRAW*; en consecuencia, no es necesario otro GO TO* para volver al punto inicial. Su uso es similar a sus homólogos GOSUB y RETURN del lenguaje BASIC.
Formato:

GOSUB* nombre-camino



Los posibles modos de acceso a un fichero bajo el control del T.O.S. son: sólo lectura, sólo escritura, lectura y escritura, y lectura y escritura comenzando por el final del fichero.

EL ACCESO A FICHEROS

El concepto de fichero es el de conjunto de elementos de información ligados a un mismo nombre.
En el BASIC del ZX-SPECTRUM los ficheros se utilizan como unidades indivisibles; es decir, se pueden almacenar datos en un vector, luego grabarlo, y más tarde volverlos a recuperar, pero siempre ha de hacerse con la totalidad de los datos.
Cuando el volumen de los datos es grande este método no es viable, puesto que exigiría cargar en memoria una gran cantidad de información, dejando un mínimo espacio para los programas. Incluso cabe la posibilidad de que los datos cargados excedan la capacidad de memoria disponible en el equipo.
El TOS resuelve este inconveniente con la facultad de leer datos de un fichero sin necesidad de cargarlo íntegramente en memoria; se carga tan sólo una parte mínima o "registro" del fichero en cuestión. Ello permitirá explotar aplicaciones que no estén limitadas por el tamaño de la memoria principal del ZX-SPECTRUM.
Las operaciones de lectura y escritura en los ficheros se realizan a través de canales. Un canal viene representado por un número asociado al fichero en uso. Con el canal se puede identificar unívocamente al fichero en las operaciones de lectura (INPUT*#) y escritura (PRINT*#); éste actúa además como un "buffer" o área auxiliar de memoria, a través, de la cual se va a establecer el flujo de información entre el disquete y la memoria principal del ordenador.
El número de canales disponibles se eleva a 16; lo que significa que se pueden mantener abiertos a la vez hasta un máximo de 16 ficheros.
Dentro de estos 16 canales se pueden distinguir dos tipos canales lentos y canales rápidos.
A los canales rápidos les corresponden los números del 1 al 4, y se diferencian de los lentos en el tamaño del área de memoria utilizada para el transvase de información de disquete a memoria y viceversa. Hay 512 bytes reservados a cada uno de los canales rápidos, y 512 bytes a repartir entre todos los canales lentos.

El sistema operativo TOS utiliza estas áreas de memoria para aumentar la velocidad de acceso al disquete, ya que cada lectura solicitada conlleva el transporte a la referida área de 256 bytes, independientemente de la cantidad de bytes implicados en la operación. Así pues, en la siguiente lectura la información solicitada estará ya en el buffer, no siendo necesario otro acceso al disquete, con lo que se gana en velocidad. En el caso de la escritura, se van almacenando en el buffer los distintos caracteres hasta completar 256 bytes; y en este preciso momento es cuando se realiza la escritura física en el disquete.

Cuando las operaciones de lectura y escritura se realizan por canales lentos, y existen varios ficheros abiertos al mismo tiempo, no se puede aplicar lo comentado en el párrafo anterior. En efecto, al ser compartida el área de memoria por todos los canales lentos, la información de un

La entrada del ZX-SPECTRUM en el entorno de control del T.O.S. llega con la incorporación al equipo de la unidad de disco INVESDISK-200.

Ficheros secuenciales y de acceso directo en el TOS

Con el sistema operativo TOS, el tratamiento de ficheros puede extenderse más allá de los límites marcados por el almacenamiento y extracción de programas, ya sea en BASIC o en código máquina. Apoyándose en el uso del disquete, el almacenamiento de la información gana en potencia y en alcance.

Los tipos de ficheros que soporta el TOS son dos: secuenciales y de acceso aleatorio o directo. En los primeros la información está almacenada igual que en una cinta magnética, siendo necesario pasar por toda la información precendente hasta llegar a la información deseada. No obstante, en los ficheros de acceso directo se puede acceder a cualquier elemento de información sin necesidad de pasar por todos los anteriores.

Los ficheros secuenciales pueden adoptar dos tipos de organización: en registros y en cadenas de caracteres o "stream". El tipo más utilizado es el último de ellos; en el que la información es almacenada en cadenas de caracteres, separadas por signos o códigos especiales que sirven para reconocer dónde empieza y termina una cadena de caracteres. Este tipo de organización está concebido para cuando la información a almacenar no tiene una estructura fija, y cada elemento puede referirse a cosas distintas. Los separadores reconocidos por el sistema operativo son: coma, retorno de carro (CHR$13), tabulador (CHR$6) o dobles comillas.

La organización en registros es más propia de los ficheros de acceso directo. Estos constan de registros de longitud fija —especificada en la sentencia de apertura del fichero— y numerados en orden creciente; de tal forma que para referirse a un registro hay que indicar el número de orden de dicho registro dentro del fichero. Cada operación de lectura/escritura se efectúa en base a bloques de información, de longitud constante, denominados registros.

A continuación se detallan algunas precisiones de importancia relativas al acceso a ficheros, tanto secuenciales como de acceso directo:

— Sólo se pueden utilizar cadenas de caracteres en la lectura o escritura de un fichero.
— La longitud máxima de estas cadenas es de 256 bytes.
— En las operaciones de lectura no se pueden utilizar cadenas de caracteres definidas con más de una dimensión.
— Es obligatorio incluir marcas de separación entre los diferentes elementos de un fichero para que, más tarde, puedan ser recuperados.
— Tan sólo se puede utilizar una cadena de caracteres en cada comando PRINT*#, siendo necesario incluir entre cada cadena el signo '+' si realmente se quieren escribir varias cadenas de una vez.

Los ficheros secuenciales están formados por cadenas de caracteres de distinta longitud y separados por marcas (stream), mientras que los de acceso directo están constituidos por registros o cadenas de longitud fija.

canal podría ser borrada al efectuarse una operación a través de otro canal diferente, por lo que cada lectura-escritura se plasma en un acceso a disco; el resultado es que el proceso será ahora bastante más lento que en el caso anterior.

COMANDOS DE ACCESO A FICHEROS

El acceso a la información contenida en los ficheros se realiza por medio de los siguientes comandos:

DIM*

Crea un fichero de datos.
Formato:

DIM* nombre-camino

OPEN#*

Abre un fichero; esto es: lo prepara para su posterior acceso asignándole un canal libre.
Formato:

OPEN#* canal; nombre-camino; modo, longitud-registro.

Los posibles modos son:
I — Sólo para lectura
O — Sólo para escritura
R — Acceso aleatorio en lectura y escritura
A — Acceso en lectura y escritura, escribiéndose la nueva información a partir de la ya existente.
La longitud del registro es un número de 1 a 256; siendo opcional en los modos I, O y A, y obligatorio en el modo R.

PRINT#*

Escribe información en el fichero.
Formato:.

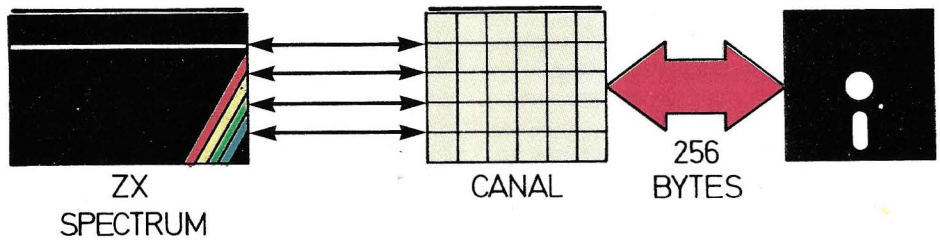
PRINT#* canal; "cadena de caracteres"

y para los ficheros de acceso directo:

PRINT#* canal; "cadena de caracteres"; AT número-registro

INPUT#*

Lee información del fichero.



El acceso a los ficheros se realiza a través de canales. Estos son áreas de memoria en las que se almacenan 256 bytes como resultado de una lectura o escritura en el disquete.

Formato:

INPUT#* canal; "cadena de caracteres"

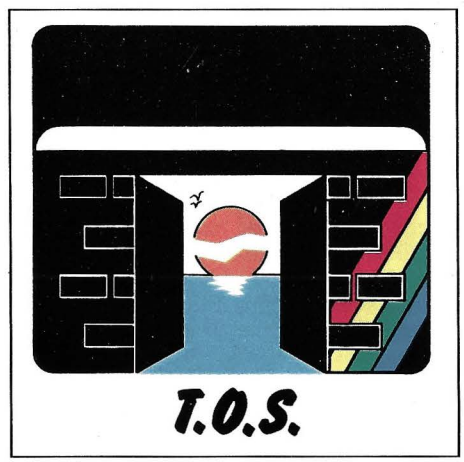
RESTORE#*

Hace que el puntero del fichero señale al principio de éste.
Formato:

RESTORE#* canal

LIST#*

Lista las características del archivo abierto



Los conectores para comunicación serie (RS/232) incorporados a la unidad INVESDISK-200, abren una puerta hacia el exterior a través de la que el ZX-SPECTRUM puede comunicarse con múltiples dispositivos periféricos.

con el número de canal especificado (número de canal, tipo de canal, modo de apertura, tipo de fichero, longitud del registro, puntero, tamaño).
Formato:

LIST#* canal

CLOSE#*

Da por terminado el acceso a un fichero, cerrándolo y liberando el canal.
Formato:

CLOSE#* canal

COMUNICACIONES EXTERNAS

La inclusión en el módulo de control de la unidad INVESDISK-200 de dos conectores para comunicación en serie con formato RS/232, abre al ZX-SPECTRUM una doble vía para dialogar con otros dispositivos.
Los dos accesos de comunicación son tratados por el sistema operativo como si se tratara de dos ficheros: CH.A.SCP para el acceso A y CH.B.SCP para el acceso B; por supuesto, éstos transmiten la información en lugar de almacenarla.
La transmisión de datos se realiza bit a bit, en serie, apoyándose en determinados protocolos de transmisión que, obviamente, han de ser iguales en los dos extremos de la línea de transmisión para que ambos dispositivos conectados puedan entenderse.
Las posibilidades del TOS no quedan restringidas a un único protocolo, sino que mediante el comando FORMAT* se pueden definir interactivamente las características del protocolo, seleccionándolas entre las contempladas por el sistema operativo. El formato de dicho comando es:

FORMAT* nombre-SCP

Una vez que se ha definido el protocolo de comunicación, se está ya en disposición de transmitir informaciones a través de estos canales. Cabe la posibilidad de enviar o recibir ficheros externos con los cuatro comandos especializados en tareas de intercambio: SAVE*, LOAD*, MERGE* y MOVE*. También es posible transmitir o recibir información no contenida en un fichero; al efecto hay que recurrir a los comandos de lectura y escritura en ficheros, formulados con las mismas reglas aplicables al acceso a ficheros convencionales.

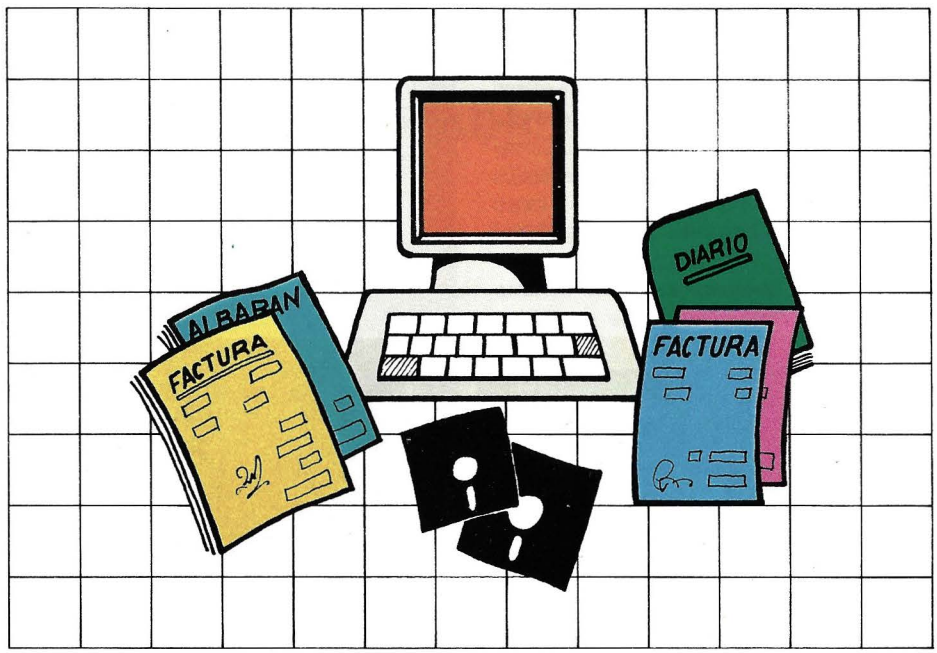
Software de administración

Programas para mecanizar las tareas de administración y contabilidad

Uno de los colectivos de profesionales cuyo trabajo es más susceptible de mecanización es el formado por economistas, administradores y contables. Por lo demás, dentro de las múltiples actividades en las que un ordenador puede colaborar en la empresa, las más frecuentes y generalizadas se encuentran, sin duda alguna, en el ámbito administrativo y/o contable.

Estos dos razonamientos iniciales, constatables desde hace muchos años, han convertido en habitual la presencia en la empresa de ordenadores convencionales y mini-ordenadores; su necesidad de explotación ha motivado el nacimiento paralelo de innumerables programas administrativos destinados a esta categoría de aplicaciones.

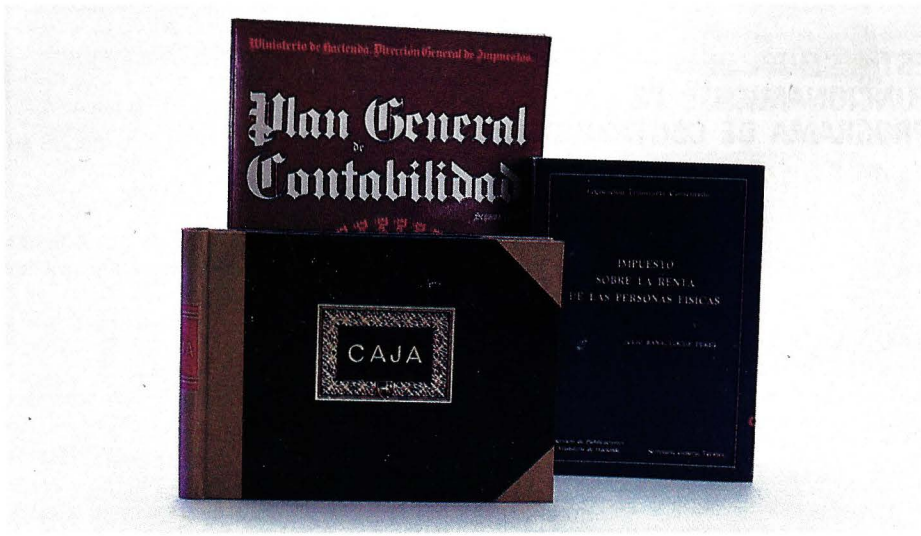
En la actualidad, el "boom" originado por la aparición de los ordenadores personales, ha hecho que esta abundancia se haya desplazado a un nuevo entorno más individual; entorno que ha aportado la posibilidad de mecanizar la administración de pequeñas empresas que jamás habrían podido acceder a un gran ordenador.



La llegada del ordenador personal ha hecho posible informatizar la administración de pequeñas empresas con equipos y programas de moderada economía.

PROGRAMAS ADMINISTRATIVOS Y PROGRAMAS CONTABLES

Evidentemente, el término administración es bastante más amplio que el término contabilidad; de hecho, cabe afirmar que la contabilidad es una parte de la administración, mientras que la administración es contabilidad más otros procesos. No obstante, el "corazón" de todo departamento administrativo de una empresa lo constituye la contabilidad; de ella se obtendrán



Dentro de las múltiples actividades en las que un ordenador puede colaborar en la empresa, las más frecuentes y generalizadas se encuentran, sin duda alguna, en el ámbito contable y administrativo.

los datos elementales para cumplimentar los restantes procesos, que serán muy distintos según la naturaleza de la empresa. Este planteamiento, hecho dentro del esquema general organizativo de una empresa, puede ser transplantado a su organización informática; así, el programa central será el encargado de la contabilidad, y como "satélites" suyos, existirán otros programas de carácter administrativo en general.

Dentro de la contabilidad podemos distinguir dos campos complementarios, pero independientes; tanto es así que podemos hablar de dos contabilidades distintas: Contabilidad Financiera y Contabilidad Analítica. La primera tiene carácter oficial y debe seguir ciertas reglas marcadas por los organismos correspondientes. Como consecuencia de esta normalización todas las empresas deben utilizar una misma "filosofía". En cambio, la Contabilidad Analítica tiene carácter privado y voluntario, de forma que cada empresa es libre de utilizar la metodología que considere más oportuna.

Tanto en este artículo como en general, cuando observemos la denominación de Contabilidad, sin ningún tipo de aclaración adicional, entenderemos que se trata de la Contabilidad Financiera. En la práctica apenas existen programas comercializados para mecanizar la Contabilidad Analítica; dada su heterogeneidad, cada usuario suele desarrollar sus propios programas. Sin embargo, para la Contabilidad Financiera se pueden encontrar en el mercado programas de todo tipo, precio y calidad.

Existen muchos programas para automatizar la contabilidad financiera. Uno de ellos, creado para los ordenadores IBM-PC y compatibles, es el denominado LINCE.

ESTRUCTURA DE FUNCIONAMIENTO DE UN PROGRAMA DE CONTABILIDAD

Existen dos conceptos fundamentales que dan lugar a sendos ficheros maestros presentes en todo programa de contabilidad: el plan de cuentas y los movimientos.

- Características del plan de cuentas.

Una cuenta puede definirse por un código formado por cierto número de caracteres, generalmente de 7 a 9, y por una descripción de dicho código. Por ejemplo la pareja <4301617, CLIENTE PEREZ> sería una cuenta en la que se reflejaría la situación contable del cliente Pérez respecto a nuestra empresa. Todas las cuentas del plan se organizan estructuradamente; así, si el identificador comienza por 4 se tratará de una cuenta de Acreedores y Deudores; si empieza por 43 será una cuenta correspondiente a un cliente, etc.

A los dos datos fundamentales, identificador y denominación, hay que añadir tres nuevos conceptos que permiten establecer valores concretos:

1. DEBE

Refleja la "deuda" contraída por la cuenta con nuestra empresa (indica de dónde se consigue el dinero).

2. HABER

Refleja la "deuda" contraída por nuestra empresa con una cuenta (indica en qué se gasta el dinero).

3. SALDO

Diferencia entre debe y haber.

Mediante estos tres atributos se obtendrá información a cualquier nivel; esto es: se puede conocer la situación de un cliente concreto mediante el debe, haber y saldo de su cuenta (4301617, para el cliente PEREZ), o la de todos los clientes en general mediante la parte común de todos los identificadores asociados (43)...

- Características de los movimientos

Para proceder a actualizar los valores de una cuenta se deben introducir movimientos o asientos contables. Todo movimiento debe ser cargado al debe de una cuenta y al haber de otra; de esta forma, la suma del debe y haber para todas las cuentas coincidirá (cuadrará, en terminología contable).

Por ejemplo: si emitimos una factura a nuestro cliente Pérez de 510.000 ptas., habrá que cargar 510.000 al debe de la cuenta del cliente (4301617) y la misma cantidad al haber de la cuenta de ventas (por ejemplo, la cuenta número 7 000 000). Cuando el cliente abone el importe de dicha factura, el movimiento se cargará al haber de la cuenta del cliente (4301617) y al debe de la cuenta correspondiente al ingreso (podrá ser a Caja o un Banco). A partir de estos dos conceptos fundamentales se obtendrán diversos informes, algunos normalizados según la ley vigente y otros de carácter voluntario. Las características de dichos informes se describirán adoptando como referencia el caso concreto de un programa de contabilidad denominado LINCE; programa cuyo desarrollo se debe a la compañía EIV (Ediciones de Informática Vicens-Vives, S. A.) y que está disponible para el ordenador personal IBM y compatibles.

PROGRAMA DE CONTABILIDAD LINCE

La aplicación está organizada en torno a un menú principal formado por veintiuna opciones diferentes. Cada una de ellas tiene una misión concreta y puede ser utilizada de forma interactiva sin más que teclear el número asociado a la opción y pulsar la tecla◂RETURN.

El programa está preparado para su explotación por parte de usuarios no expertos en informática; en consecuencia, guía al operador de forma que éste tan sólo se vea obligado a elegir entre varias opciones o responder a preguntas formuladas por el programa. Otra característica importante del paquete LINCE es la incorporación de determinados controles de erro-

El término administración es más amplio que el término contabilidad; de hecho, este último es un subconjunto de aquél, si bien es cierto que la contabilidad es el auténtico núcleo de la administración.

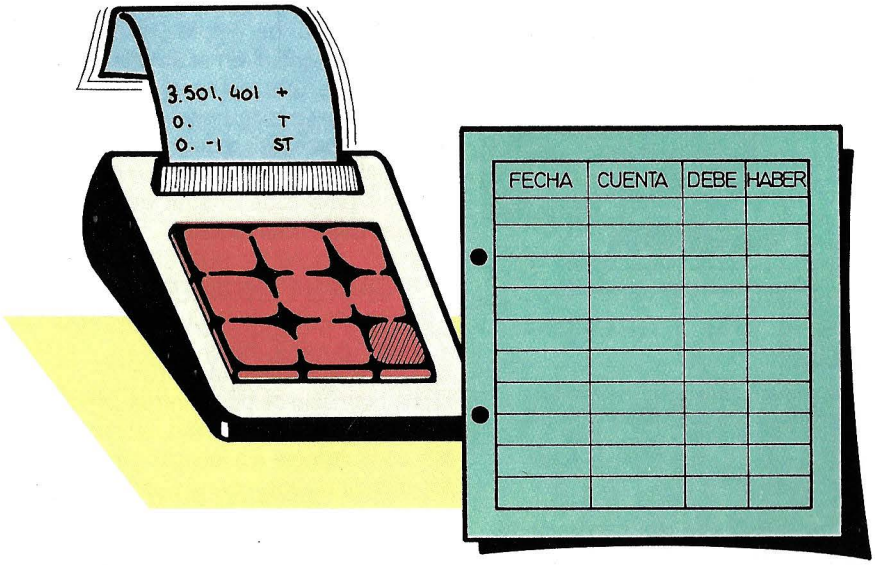
Técnicas de mecanización contable

Desde hace varias décadas se han buscado métodos, más o menos mecanizados, para aliviar el tedioso trabajo de asentar los movimientos contables. A continuación se describen tres de ellos que supusieron, cada uno en una época distinta, una importante novedad.

1. Método basado en plantillas (sistema de "calco")
Inicialmente, los libros contables se actualizaban de forma manual, y en algunos casos era necesario realizar más de una copia. Una novedad importante sobre estos métodos manuales fue la utilización de plantillas especiales, sobre las que se situaban impresos taladrados autocopiativos, de tal forma que escribiendo tan sólo una vez se producían varias copias. Por supuesto, para las operaciones aritméticas sólo se disponía de elementales máquinas de calcular que sumaban, restaban y, en el mejor de los casos, multiplicaban.

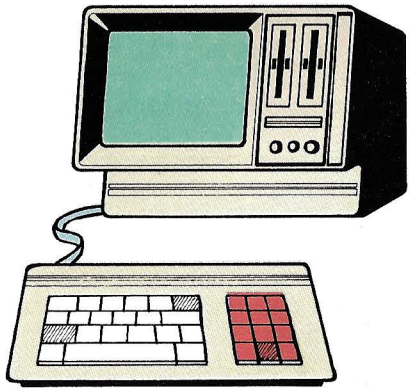
2. Método basado en máquinas especializadas
Un nuevo paso adelante en la mecanización de los procesos contables fue la aparición de máquinas especializadas. Eran susceptibles de ser programadas mediante métodos muy elementales; y además de permitir la obtención de documentos con escritura mecanizada, facilitaban los procesos de cálculo.

3. Método basado en la Informática
El tercer método revolucionario, el actual, se fundamentó en el uso de ordenadores junto con programas especializados en tareas de Contabilidad. La nueva tecnología Informática ha permitido, entre otras aportaciones, la aparición del concepto "Contabilidad en tiempo real"; es decir, los informes contables se obtienen con los datos actualizados, justo hasta el preciso instante en el que se produce el informe.

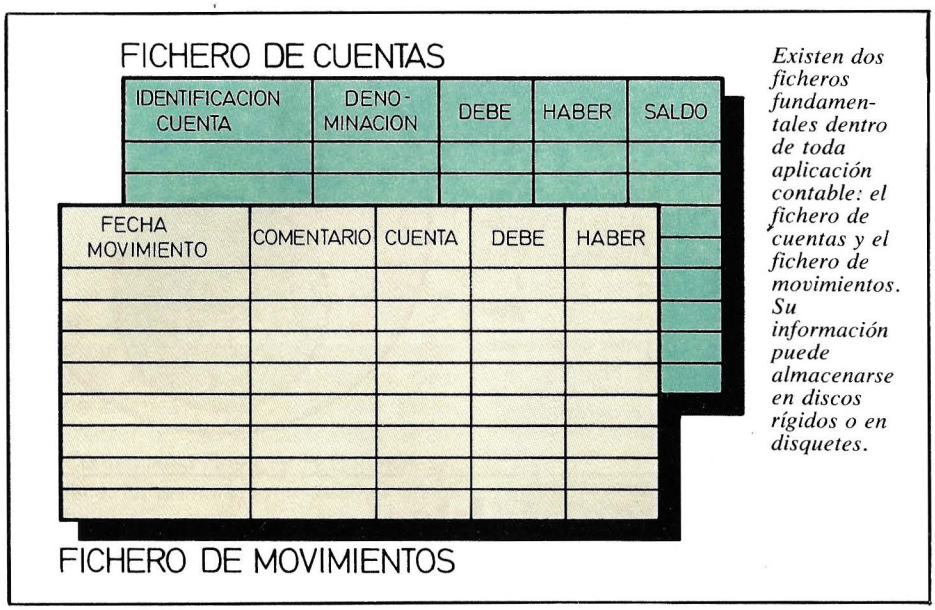


Sistema de mecanización contable basado en plantillas, hojas autocopiativas preimpresas y máquinas de calcular.

Sistema basado en máquinas especializadas.



Sistemas basados en equipos informáticos.



Existen dos ficheros fundamentales dentro de toda aplicación contable: el fichero de cuentas y el fichero de movimientos. Su información puede almacenarse en discos rígidos o en disquetes.

La gestión administrativa de la pequeña o mediana empresa, así como la contabilidad asociada al ejercicio de una profesión liberal, quedan hoy en día al alcance de un simple ordenador personal.

res para garantizar que la información introducida es correcta; así, por ejemplo, no permitirá dar un mismo identificador a dos cuentas distintas, introducir un movimiento sobre una cuenta inexistente, etc. Si a estas dos características se suma la completa documentación que lo acompaña, redactada en castellano, cabe tan sólo concluir que la utilización del programa LINCE resulta cómoda y sencilla.
Desde luego, no es posible resumir en un sólo capítulo la descripción completa de las 21 opciones que ofrece el programa; por lo tanto, en los próximos párrafos se enumerarán únicamente algunas de las más importantes:

- INICIALIZACION

Se suele utilizar una única vez, al comenzar el primer proceso de contabilidad. Permite definir características de la impresora a utilizar para los listados, precisar conceptos manejados en el programa y crear los ficheros que contendrán el plan de cuentas y los movimientos. Es importante destacar que este programa permite llevar la contabilidad de hasta 99 empresas distintas y para cada una de ellas se debe realizar una inicialización.

- ENTRADA PLAN DE CUENTAS

Permite dar altas, bajas, realizar consultas, y modificaciones sobre el fichero del plan de cuentas. En las altas, bajas y modificaciones tan sólo se podrá introducir el identificador y la denominación; en cambio, en las consultas se observarán, además de los dos datos anteriores, el debe, el haber y el saldo correspondientes.

- ENTRADA DE MOVIMIENTOS

Mediante esta opción, además de introducir los movimientos habidos, se podrá realizar un proceso de cuadre que garantice que las cantidades introducidas son correctas. Para ello, el programa evaluará la coincidencia de las sumas de debe y haber.

- CONSULTA DE MOVIMIENTOS

Su misión es obvia; permite visualizar por pantalla los movimientos introducidos mediante la opción anterior.

- LISTADO PLAN DE CUENTAS

Produce un informe por impresora de todo o parte del plan de cuentas; para ello hay que comunicar al programa los límites inferior y superior, de forma que éste sólo imprima las cuentas comprendidas en dicho intervalo. Otra posibilidad asociada a esta opción consiste en ordenar que se produzca un salto de página cada vez que finalice la impresión de un grupo de cuentas.

- LISTADO DE BALANCE

Obtiene un informe con los datos relativos al debe, haber y saldo de cada cuenta, para los movimientos efectuados en un período de tiempo variable (generalmente un mes) y acumulado a origen.

- LISTADO BALANCE ACTIVO/PASIVO

El informe obtenido en este caso es similar al anterior, pero agrupando las cuentas según pertenezcan al activo o al pasivo, y no por debe y haber.

- LISTADO DE DIARIO CLASIFICADO

Obtiene un listado, utilizable como "Diario Oficial", que contendrá todos los movimientos producidos, clasificados según la fecha en la que se introdujeron en el ordenador. Evidentemente, para producir este informe el programa debe realizar una clasificación previa; en este punto, el usuario puede decidir si desea que se realice tan sólo la clasificación, si desea obtener únicamente el listado, o si quiere que se efectúen ambos procesos consecutivamente.

- CIERRE PERIODO

Esta opción sirve para dejar a cero los datos de un período, acumulando previamente sus valores al plan de cuentas. Mediante la ejecución de este proceso se puede regular el plazo de tiempo que se incluirá en varios listados; entre ellos, el más importante es el diario oficial.

- CIERRE EJERCICIO

El objeto de esta opción es doble. Por un lado se encarga de dejar "saldadas" las cuentas de ciertos grupos del plan, acumulando los saldos de las mismas en la cuenta de resultados; y por otro lado, graba en el fichero de movimientos los asientos contables que justifican esta operación.

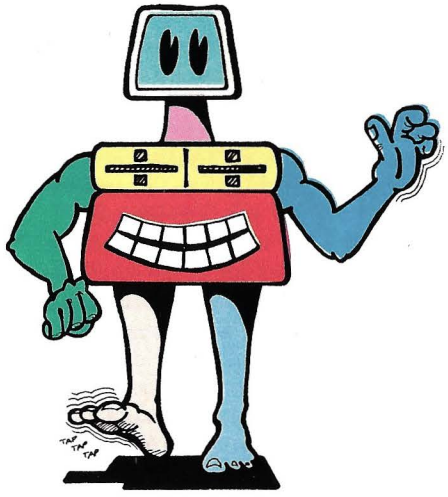
- APERTURA EJERCICIO

Se encarga de producir los saldos de apertura, para cada cuenta, en el debe si el saldo es positivo y en el haber si es negativo, además de grabar los movimientos correspondientes.

La pantalla como lienzo

Gráficos en color

En el primero de los capítulos dedicados al manejo de las capacidades gráficas del ordenador se vio cómo se trazan puntos y rectas. Utilizando esas dos herramientas se puede dibujar casi cualquier figura. Sin embargo, no basta con trazar el contorno. A menudo es preciso rellenar las superficies con color. Y, de todas formas, un gráfico sin color no es un gráfico completo.
En este capítulo se introducen los comandos adecuados para el uso del color. Junto con ellos se presenta el encargado de trazar circunferencias. Todo ello permite la creación de los más complejos dibujos en la pantalla del ordenador.



Sin lugar a dudas, el color es una de las características más importantes del mundo en que vivimos.

TRAZANDO LINEAS CURVAS

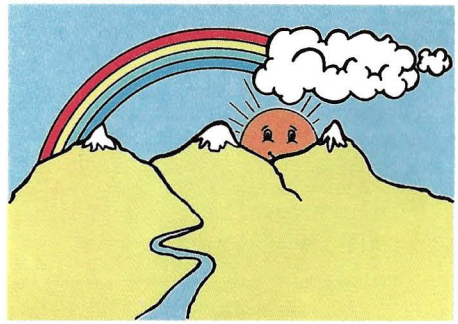
En un capítulo anterior se mostró la forma de trazar rectas "punto a punto"; método que permite también el trazado de circunferencias. Allí se mostraba un programa que permitía dibujar una circunferencia. El listado del referido programa es el que aparece a continuación:

```
10 FOR I=0 TO 6.28 STEP 0.01
20 PSET (100,100)
30 PSET – STEP (60*COS(I),60*SIN(I))
40 NEXT I
```

Tan sólo cuatro líneas son suficientes para dibujar en la pantalla una circunferencia con centro en el punto 100,100 y un radio de 60 pixels. El trazado se realiza punto a punto, calculando las coordenadas del siguiente punto a iluminar mediante el seno y el coseno del ángulo que forma con la horizontal.
Sin embargo, en muchos aparatos existe un comando BASIC especializado en estas labores. Este comando responde a la palabra clave CIRCLE.
Para definir una circunferencia basta con dar su centro y su radio. De la misma forma, el comando CIRCLE admite estos dos datos en su argumento. El primero de ellos, el centro, se indica mediante sus dos coordenadas: horizontal y vertical (para localizarlas adecuadamente es preciso conocer la resolución que ofrece el aparato y el origen de coordenadas que toma). El dato correspondiente al radio se mide en "pixels" o puntos luminosos, siendo por lo tanto un número entero.
En consecuencia, la instrucción BASIC que produce el trazado de una circunferencia de radio 40 y centro en el punto 50,50 es la que sigue:

```
100 CIRCLE (50,50),40
```

La formulación elegida es la del BASIC de Microsoft. Para establecer su equivalencia en otros dialectos es conveniente remitirse a la tabla de conversión incluida en estas páginas.
En la ejecución del comando CIRCLE es preciso tener en cuenta las dimensiones de la pantalla. Si durante el trazado de la circunferencia se exceden los límites de ésta se producirá un error y se detendrá la ejecución del programa.

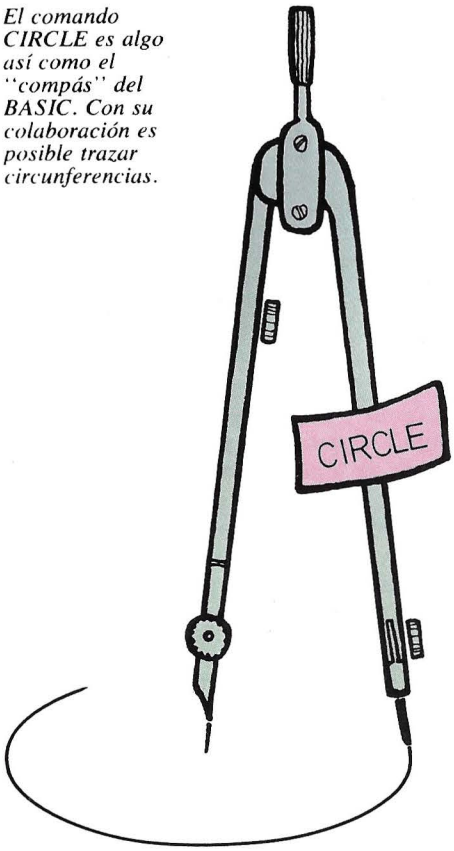


El vocabulario de comandos y funciones del lenguaje BASIC facilita la creación de gráficos en color con la pantalla como lienzo.

CIRCLE

Traza circunferencias, arcos de circunferencias y elipses en la pantalla.

Formato: CIRCLE(<X>,<Y>),<rad>,<col>,<ini>,<fin>,<asp>
Ejemplos: 10 CIRCLE (100,100),95
50 CIRCLE (90,85),70,,,,2



El comando CIRCLE es algo así como el "compás" del BASIC. Con su colaboración es posible trazar circunferencias.

Como ejemplo del uso de CIRCLE se muestra a continuación un sencillo programa.

```
10 SCREEN 2
20 FOR I=1 TO 90
30 CIRCLE (100,100),I
40 NEXT I
50 GOTO 50
■
```

En este nuevo ejemplo se han utilizado las líneas 10 y 50 para efectos secundarios: entrar en el modo de pantalla de gráficos y evitar que se pierda el dibujo al detenerse la ejecución (véase el primer capítulo de gráficos). El resto del programa aparece encerrado en un bucle FOR/NEXT; bucle que se utiliza para variar el radio de la circunferencia trazada en la línea 30. En conjunto, el programa dibuja 90 circunferencias concentricas contiguas. Ello da un aspecto de circulo lleno; aunque se aprecian puntos sin "pintar", debido a que las circunferencias no son perfectas.

ARCOS DE CIRCUNFERENCIA

Las circunferencias no son lo único que se puede trazar directamente con el comando CIRCLE Este comando permite también dibujar arcos, es decir, fragmentos de circunferencia. Para ello se necesitan dos datos más: los que indican el principio y el final del arco. En este punto es necesario conocer cómo se recorre la circunferencia.

El comando CIRCLE equivale al programa de trazado "manual" de circunferencias mostrado en el apartado anterior. La circunferencia se va recorriendo en sentido contrario al de las agujas del reloj. El punto inicial es el situado más a la izquierda; esto es, el que se encuentra a la misma altura que el centro. En la circunferencia entera, el punto final coincide con el inicial: se da una vuelta completa volviendo al mismo punto. Esto es lo que se hace en el ejemplo anterior al variar el contador del bucle de 0 a 6.28 (10 FOR I=0 TO 6.28 STEP 0.01); donde la cantidad 6.28 corresponde a $2*\pi$, que es el número de radianes que completa la circunferencia.

Para trazar arcos con el programa anterior basta con alterar los valores de los límites del comando FOR. De forma análoga, se han de indicar los límites en el trazado de arcos con CIRCLE. Esos dos datos se añaden en el argumento de dicho comando e irán precedidos de un tercero que indica el color del arco (o de la circunferencia completa). A título de ejemplo, la siguiente línea traza una circunferencia con el color número 3.

```
10 CIRCLE (120,75),25,3
```

Por el contrario, la línea que se muestra a continuación sólo presenta la mitad superior de dicha circunferencia.

```
10 CIRCLE (120,75),25,3,0,3.14
```

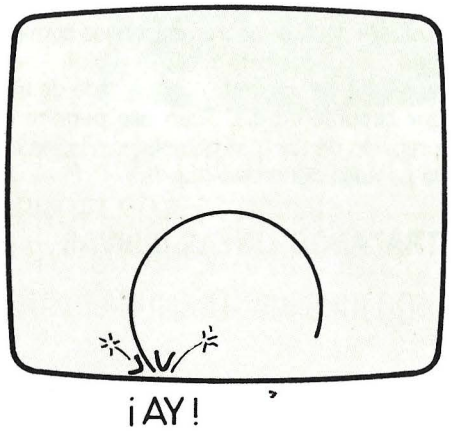
El arco de circunferencia especificado va desde los 0 a los 3.14 radianes (de 0 a π radianes); o lo que es lo mismo, se dibuja tan sólo media circunferencia. Utilizando estos dos nuevos parámetros se pueden construir, pues, toda clase de líneas curvas.

ELIPSES

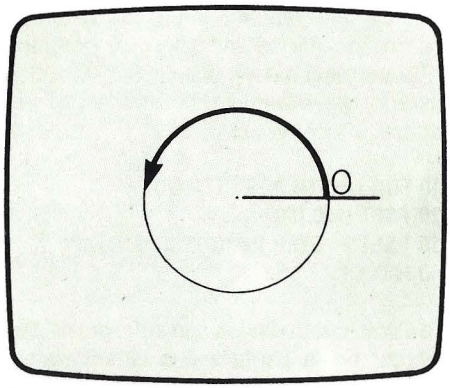
Una elipse es una figura geométrica muy relacionada con la circunferencia. Viene a ser como una circunferencia "aplastada". Dicha figura puede ser también creada por medio del programa "manual" de más arriba. Para ello se ha de alterar la línea en la que se calculan las coordenadas de cada punto. El truco consiste en "engañar" al ordenador dándole dos radios distintos.

```
30 PSET – STEP (60*COS(I),30*SIN(I))
```

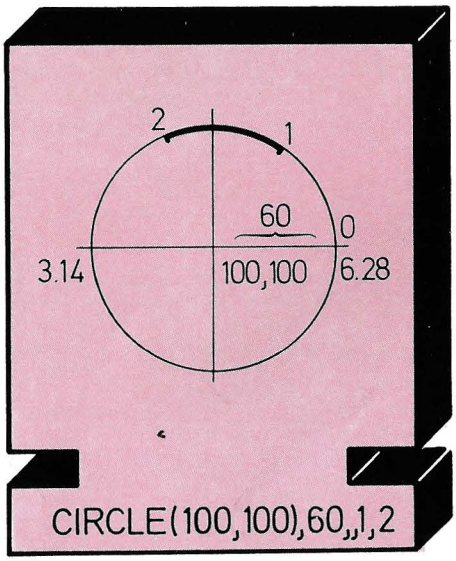
En este caso se ha variado el radio de la zona correspondiente a la coordenada Y. Ello producirá coordenadas verticales menores (la mitad de las horizontales); en otras palabras: una circunferencia "achatada". Esto mismo se puede codificar de forma más flexible utilizando una variable



Si durante la ejecución del comando CIRCLE se rebasan los límites de la pantalla, se producirá una situación anómala conducente a error.



Sentido en el que avanza el dibujo de las noventa circunferencias concéntricas ordenadas por el programa reproducido en el texto.



Ejemplo de trazado de un arco de circunferencia por medio del comando CIRCLE.

intermedia. Esta permitirá trazar distintos tipos de elipses, dependiendo del valor que adopte la mencionada variable. Este sería el aspecto de la nueva línea.

30 PSET – STEP (60*COS(I),R*60*SIN(I))

Si se deposita en R el valor 0,5, se obtendrá la misma elipse que con la línea anterior. Un valor de 1 en R producirá circunferencias. Esto indica que la variable intermedia R representa la "relación de aspecto" o relación entre ambos radios: eje mayor y eje menor.

Para trazar elipses con la colaboración de CIRCLE se emplea un método similar. De esta forma únicamente se necesita aportar un dato más. Este dato se añade al final de la lista situada en el argumento de CIRCLE. El nuevo parámetro proporcionará circunferencias cuando valga 1. Si su valor es menor que 1, el radio especifica la longitud del semieje vertical (esto es, del centro al punto más alto de la elipse). Cuando valga más de 1, el radio indicará el número de pixels que posee el semieje horizontal.

100 CIRCLE (100,100),90,3,,2

En esta línea se muestra un ejemplo del trazado de una elipse. Las posiciones empleadas para los puntos inicial y final del arco se han dejado en blanco. Esto significa que se dibujará la elipse entera. Obsérvese el empleo de las comas para no confundir la relación de aspecto con los límites del arco.

Como colofón a las líneas dedicadas a CIRCLE se muestra un pequeño programa que traza diferentes elipses con centro en el mismo punto:

```
10 SCREEN 2
20 A=8
30 FOR I=1 TO 5
40 CIRCLE (100,100),90,3,,A
50 A=A/2
60 NEXT I
70 GOTO 70
■
```

En este ejemplo se ha utilizado la variable A para almacenar el valor de la relación de aspecto. Dicho valor se reduce a la mitad en cada ejecución del bucle FOR/NEXT; de esta forma se consiguen relaciones de aspecto cada una la mitad de la anterior.

EL COLOR

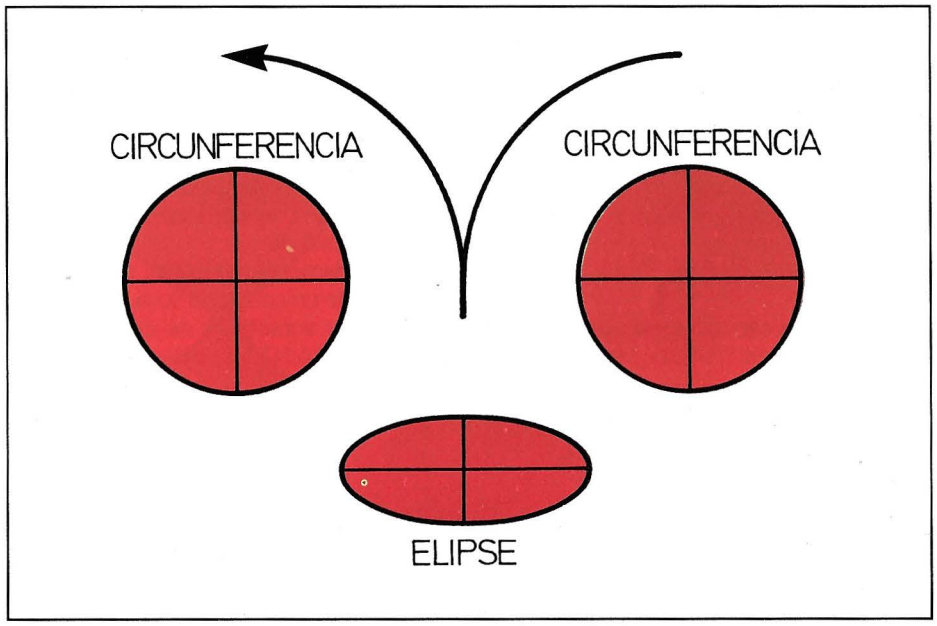
El color es algo fundamental en la vida de las personas. La superficie de la Tierra está llena de color. El color sirve para distinguir unas cosas de otras. A menudo se relacionan colores con conceptos abstractos (verde es el color de la esperanza, blanco el de la pureza, etc.). E incluso una determinada combinación de colores puede producir placer o aversión.

Los objetos que se tienen a la vista se identifican por su forma y su color. Los comandos explicados hasta ahora proporcionan formas. Es hora, pues, de entrar en el mundo del color de la mano del BASIC.

La forma de programar los colores a utilizar en el trazado de gráficos es algo muy particular de cada dialecto BASIC. Sin embargo, existen tres zonas específicas de la pantalla en las que se puede emplear el color. Estas corresponden a las siguientes: primer plano (o tinta), fondo (o papel) y borde.

Cuando se escribe texto en la pantalla, las letras son de un color distinto al del fondo. Las letras se consideran trazadas en el color de primer plano. El color de la pantalla vacía es el color de fondo; mientras que el tercer color se refiere a la zona que rodea a la zona utilizable de la pantalla (en aquellos aparatos en los que existe dicha zona).

Existen diferentes métodos para establecer el color que va a tomar cada una de estas zonas. Como ya es habitual, aquí se comentará el método empleado por el BASIC de Microsoft; las diferencias con otros dialectos se muestran en la correspondiente tabla de conversión.



Una elipse es semejante a una circunferencia "aplastada".

El BASIC de Microsoft utiliza el comando COLOR para especificar el color de cada zona en la pantalla (primer plano, fondo y borde). No obstante, los comandos gráficos llevan todos un parámetro que especifica el color que han de utilizar. Esto permite jugar con varios colores de primer plano. En resumen, el formato general del comando COLOR es el siguiente:

(Número de línea) COLOR [<primer plano>][,[<fondo>][,<borde>]]

en donde la ausencia de un parámetro revela que la zona afectada no se desea modificar. Así, por ejemplo, la siguiente sólo variará el color del borde de la pantalla.

```
120 COLOR,,2
```

El número de colores disponibles es también muy particular de cada máquina. Los diferentes conjuntos de colores pueden variar desde 2 (blanco y negro) hasta 256 o más. Por regla general, cada color utilizable lleva asociado un número o código. De esta forma, cada color se identifica por medio del respectivo código. Este es el método puesto en práctica en los comandos gráficos comentados.
Además de todo esto, en algunos modelos de ordenadores el número de colores utilizables depende del modo gráfico en el que se opere. En cualquier caso, el número de colores y el código de cada uno de ellos debe consultarse en las páginas del manual correspondiente.

LLENANDO LA PANTALLA DE COLOR

En el anterior capítulo de la obra dedicada a la creación de gráficos se presentó el comando LINE. Este permitía dibujar rectángulos, y con la opción BF rellenarlos del color especificado. De esta forma se conseguían zonas (rectangulares) cubiertas de un determinado color. Este método permite colorear porciones de la pantalla muy específicas en cuanto a su forma. Si la zona a colorear no presenta una distribución rectangular no será posible utilizar el comando LINE.
Para rellenar superficies redondas se puede emplear el subprograma incluido como ejemplo del uso de CIRCLE. Pero, como ya se vio, el resultado no es ni mucho menos perfecto. Y, si la superficie no se encuadra en ninguno de estos contornos regulares, la cosa se hace bastante más complicada.
Afortunadamente, el BASIC de Microsoft dispone de un comando cuya función es la de rellenar superficies de color. Este comando es PAINT (en inglés, pintar). El comando PAINT rellena con el color especificado una superficie cerrada de la pantalla. Para su empleo es preciso delimitar previamente la superficie a colorear para que ésta quede totalmente cerrada. Ello se puede conseguir con el uso de LINE y/o CIRCLE.
Una vez dibujado el contorno basta con ejecutar el comando PAINT indicando un punto interior de la superficie. Según la localización del punto especificado se rellenará una u otra superficie.
El empleo de un punto perteneciente a la superficie a rellenar es fundamental. Si se traza una circunferencia, se puede pintar tanto su interior como la parte exterior de la misma. Así pues, el punto de referencia identifica unívocamente la zona de la pantalla que se desea colorear. Por ejemplo:

```
30 CIRCLE (100,100),90,2
40 PAINT (100,100),2
■
```

PAINT

Rellena una superficie con el color especificado.

Formato: PAINT (<X>,<Y>),<color>
Ejemplos: 20 PAINT (10, 50), 3
70 PAINT (100,100)

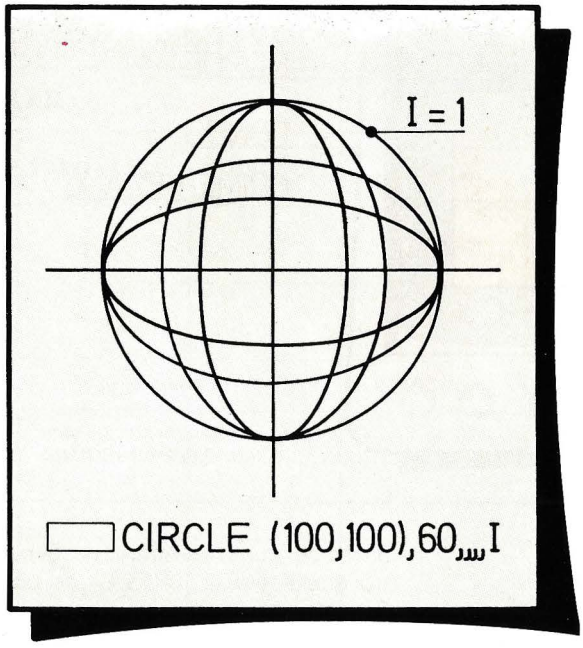
Estas dos líneas trazan una circunferencia y la llenan del color número 2. Se ha empleado el centro de la circunferencia como punto de referencia para PAINT, pero hubiera valido cualquiera de los situados en el interior de la misma; por ejemplo:

```
40 PAINT (100,50),2
```

Si el punto especificado se encuentra en la zona externa, ésa será la parte coloreada. Esto mismo es lo que sucedería con la ejecución de la siguiente línea:

```
40 PAINT (120,100),2
```

En los tres ejemplos mostrados se observa que el color del "relleno" coincide con el de la circunferencia inicial (concretamente con el número 2). Ello no se debe a una coincidencia. El comando PAINT considera como límites válidos aquellos trazos que son del mismo color que el indicado en su argumento. Si no existen en la pantalla líneas del color especificado se rellenará toda la superficie de la misma.



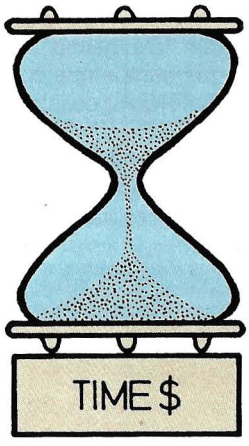
Dibujo de varias elipses superpuestas por medio del comando CIRCLE. Cada una de ellas se obtiene asignando un valor distinto al parámetro I.

Variables de tiempo

En un capítulo precedente se habló de las variables del sistema, o lo que es lo mismo: de posiciones de memoria reservadas para uso del sistema. Entre ellas se encuentra una variable que puede resultar de utilidad en múltiples casos. Se trata de una variable cuyo contenido se incrementa automáticamente mientras el aparato está encendido. Por lo tanto, dicha variable puede ser empleada para medir el tiempo transcurrido entre dos acontecimientos. Muchos dialectos BASIC permiten el acceso directo a esta variable, la cual suele estar asociada a la palabra clave TIME o TIME$ (en el primer caso se trata de una variable numérica).
Dependiendo del ordenador, esta variable se incrementará con mayor o menor frecuencia; por ejemplo, es habitual que se incremente en una unidad cada sesentavo de segundo.
La referida variable permite medir el tiempo de ejecución de un programa. Por ejemplo, el siguiente programa calcula el tiempo invertido por el ordenador en ejecutar 10.000 sumas:

```
10 LET T=TIME
20 FOR I=1 TO 10000
30 LET A=B+C
40 NEXT I
50 PRINT 60*(TIME-T)
```

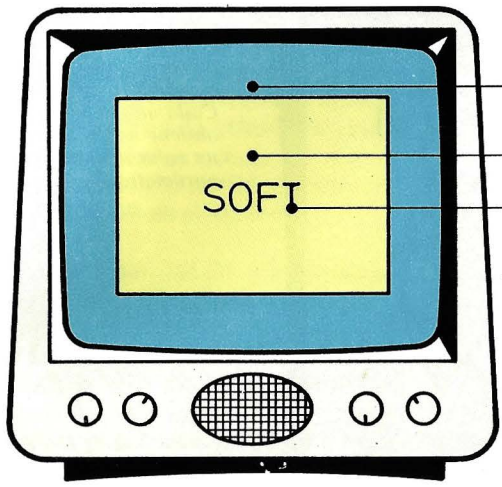
El valor inicial de TIME es almacenado en la variable T. Una vez realizado el proceso, se calcula el tiempo transcurrido por medio de una simple resta (TIME−T). Si el ordenador cuenta en sesentavos de segundo, habrá que multiplicar la cantidad obtenida por 60. Este es, precisamente, el cometido de la línea 50.



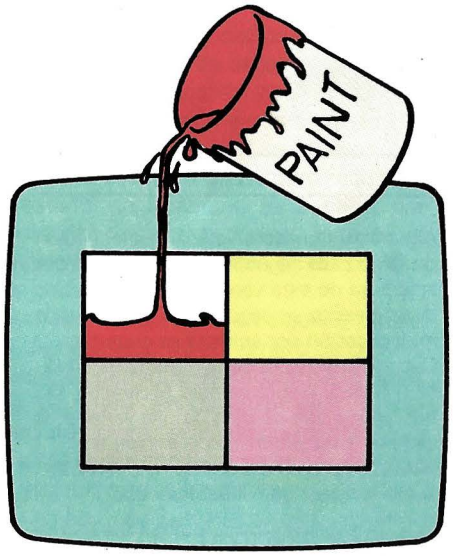
Algunos ordenadores actualizarán el valor de TIME cada segundo, lo cual significa que la lectura de esta variable vendrá directamente en segundos. En tal caso, la multiplicación por sesenta es superflua. Cuando la variable de tiempo no es TIME sino TIME$, la cosa cambia. El carácter "$" final indica que se trata de una variable de cadena. Esta circunstancia impide su tratamiento aritmético. Además, TIME$ suele emplear un formato del tipo *hh:mm:ss*. Esto se verá más claro con un ejemplo:

```
PRINT TIME$<CR
0:23:35
```

Dicha variable puede ser actualizada mediante una sentencia de asignación del tipo LET TIME$="12.25:30".
Otros aparatos disponen además de una variable que almacena la fecha. Esta última suele estar asociada a la palabra DATE$.



En la pantalla cabe diferenciar tres zonas que, habitualmente, pueden tomar colores distintos.



El comando PAINT permite dar color a superficies encerradas por líneas.

Según lo dicho, el formato en el que se presenta este comando es el que figura a continuación.

(Número de línea) PAINT (<X>, <Y>)[,<color>]

En donde <X> e <Y>indican las coordenadas del punto de referencia. El parámetro que especifica el color es optativo. Como en el resto de los comandos gráficos, si no se adjunta este parámetro se utilizará el color actual de primer plano.

COMO AVERIGUAR EL COLOR DE UN PUNTO

Se ha visto la forma de dar color a los puntos de la pantalla. Sin embargo, es posible que en algún momento se necesite conocer el color de un determinado punto. Esto se puede hacer tomando buena nota de los puntos que se colorean. Si se utiliza PAINT para rellenar una superficie compleja, el cálculo de los puntos afectados puede resultar extremadamente arduo. De hecho la localización exacta de los puntos que componen una simple circunferencia constituye una tarea nada fácil.

Conocer el estado de un punto luminoso resulta útil para localizar trazos en la pantalla. Una vez localizados, éstos pueden ser borrados o emplear el dato que identifica su situación para posteriores cálculos. En el BASIC de Microsoft existe una función que devuelve el código de color de un punto dado. Dicha función es POINT (no confundir con PAINT) y, como es de suponer, admite en su argumento las coordenadas de un punto de la pantalla. Por lo tanto, su formato general es el siguiente:

(Número de línea) [LET] <variable>= POINT(<X>,<Y>)

En el formato se ha incluido POINT en el seno de una sentencia de asignación. Al tratarse de una función, POINT deberá situarse en una expresión numérica o donde se pueda hacer uso del dato que proporciona.

Como ejemplo del uso de POINT se propone un programa que localiza un punto de color 1 en el interior de la línea 100 (horizontal) de la pantalla.

```
300 FOR I=1 TO 256
310 IF POINT(I,100)<>1 THEN NEXT I
320 PRINT I
```

Se ha supuesto que la pantalla contiene 256 puntos horizontales (del 1 al 256). El bucle FOR/NEXT que recorre la línea finaliza cuando se encuentra un punto de color número 1. En ese momento se escribe el valor de I, el cual corresponde a la coordenada X del punto hallado.

Como se ha indicado, POINT proporciona el código de color de un punto. Además de ello, si el punto rebasa de los límites de la pantalla, el valor devuelto será −1. Como los códigos de color son siempre números positivos, el valor negativo indica claramente que se ha inspeccionado un punto que no pertenece a la pantalla. Esta posibilidad puede emplearse para medir, de una forma efectiva, la superficie de la pantalla. El programa que proporciona esta información podría codificarse en BASIC de la siguiente forma.

```
10 REM LIMITES DE LA PANTALLA
20 SCREEN 2
30 X=0
40 X=X+1
50 A=POINT (X,2)
60 IF A <>−1 THEN GOTO 40
70 Y=0
80 Y=Y+1
90 A=POINT (2,Y)
100 IF A <>−1 THEN GOTO 80
```

POINT

Proporciona el código de color del punto indicado en su argumento.

Formato: POINT(<X>,<Y>)
Ejemplos 20 PRINT POINT(125,24)
70 LET COL=POINT(125,24)

TABLA DE CONVERSION

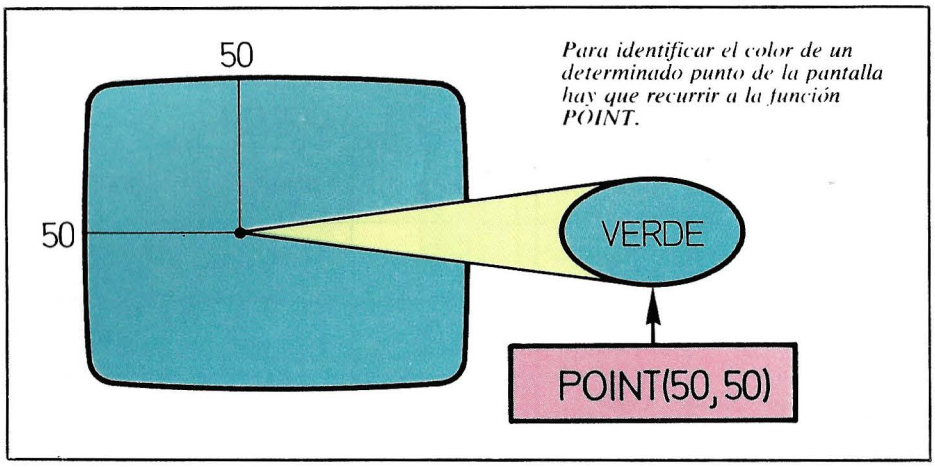
ORDENADOR	DIBUJO				COLOR		
	CIRCLE (X,Y), R,C,I,F,A	Arcos	PAINT (X,Y),C	POINT (X,Y)	TINTA	PAPEL	BORDE
APPLE II (APPLESOFT)	—	—	—	—	HCOLOR (7)	—	—
APRICOT (M-BASIC)	—	—	—	—	—	—	—
ATARI	—	—	—	—	COLOR <n>	—	—
CBM 64	—	—	—	—	(8)	(8)	—
DRAGON	CIRCLE (X,Y), R,C,A,I,F	CIRCLE	PAINT (X,Y), C,CC (3)	PPOINT (X,Y)	COLOR T,P		—
EQUIPOS MSX	CIRCLE (X,Y), R,C,I,F,A	CIRCLE	PAINT (X,Y), C,CC (3)	POINT (X,Y)	COLOR T,P,B		
HP-150	—	—	—	—	—	—	—
IBM PC	CIRCLE (X,Y), R,C,I,F,A	CIRCLE	PAINT (X,Y), C,CC (3)	POINT (X,Y)	COLOR T,P,B		
MPF	—	—	—	—	HCOLOR (7)	—	—
NCR DM-V (MS-BASIC)	—	—	—	—	—	—	—
NEW BRAIN	—	—	—	—	—	—	—
ORIC	CIRCLE R,C (1)	—	FILL lin, Col, byte (4)	POINT (X,Y) (5)	INK <n>	PAPER <n>	—
SHARP MZ-700 (MZ-BASIC)	—	—	—	—	COLOR X,Y,T,P		
SINCLAIR QL	CIRCLE X,Y,R	ARC X_1,Y_1 TO X_2, Y_2, ANG	—	—	INK <n>	PAPER <n>	BORDER <n>
SPECTRAVIDEO	CIRCLE (X,Y), R,C,I,F,A	CIRCLE	PAINT (X,Y),C	POINT (X,Y)	COLOR T,P,B		
ZX-SPECTRUM	CIRCLE X,Y,R	DRAW X,Y, ANG (2)	—	POINT (X,Y) (6)	INK <n>	PAPER <n>	BORDER <n>

X, Y: Coordenadas del centro de la circunferencia o del punto a tratar. R: Radio. C: Color. I: Angulo inicial del arco. F: Angulo final del arco. A. Relación de aspecto. ANG Angulo medido en radianes que corresponde al arco a trazar. T Código de color de primer plano (tinta). P: Código de color de fondo (papel). B: Código de color del borde

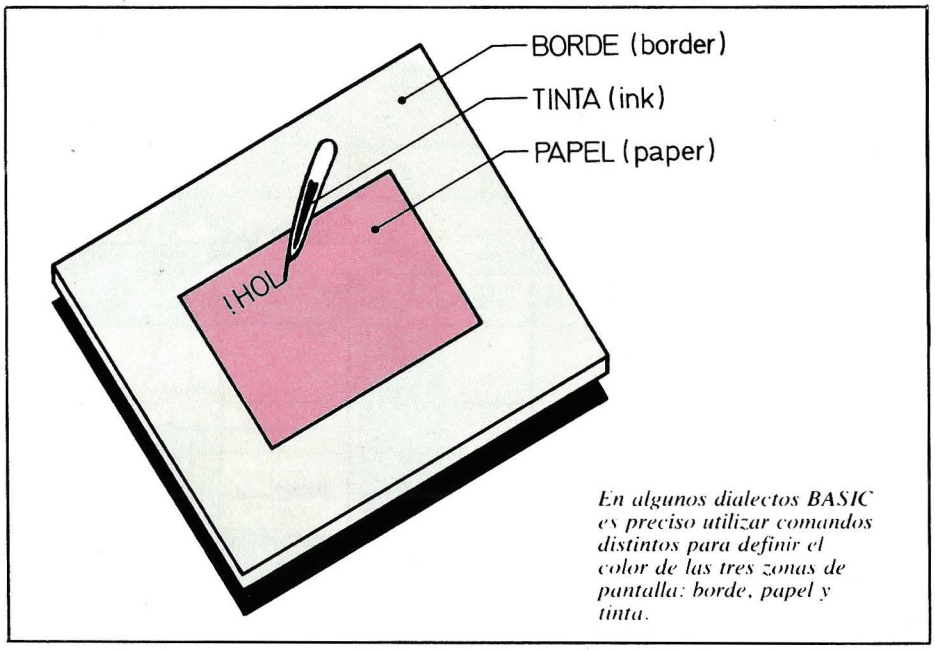
(1) El color se indica como 0 si es el de fondo ó 1 si es el de primer plano (2) Actúa como DRAW (ver primer capitulo de gráficos) pero trazando una linea curva. La curvatura viene impuesta por el ángulo en radianes a trazar (3) CC especifica el color del contorno a rellenar. (4) Sirve para rellenar las lineas y columnas de la pantalla indicadas con el byte que se adjunta. (5) Devuelve el valor – 1 si es color de primer plano y 0 si es fondo. (6) Proporciona el valor 0 si corresponde a "papel" o 1 si corresponde a tinta. (7) Es una variable que se actualiza con el código de color en una sentencia de asignación. (8) El color se cambia mediante códigos de control en el argumento de PRINT.

```
110 SCREEN 0
120 PRINT "PIXELS: HORIZONTALES
    "X;",VERTICALES ";Y
```

En este ejemplo se crean dos bucles que exploran una línea vertical y otra horizontal. Al alcanzar ambos límites de la pantalla finalizan los bucles. La variable que sirvió de contador dará la medida en pixels de cada lado de la pantalla. La línea 110 se utilizará en aquellos aparatos que no permitan escribir texto en la pantalla de gráficos (SCREEN 2). En el ejemplo se ha considerado que el modo de texto se obtiene por medio del comando SCREEN 0.



Para identificar el color de un determinado punto de la pantalla hay que recurrir a la función POINT.



En algunos dialectos BASIC es preciso utilizar comandos distintos para definir el color de las tres zonas de pantalla: borde, papel y tinta.

COLOR Y TEXTO

Se ha indicado más arriba que el comando que controla el color es precisamente COLOR. Este comando establece los colores de primer término, fondo y borde. Su acción afecta tanto a gráficos como a texto. En el caso del texto los caracteres se escriben en el color de primer plano.

Otros dialectos BASIC, diferentes al de Microsoft, utilizan un comando para cada zona de color. Estos suelen corresponder con: INK (tinta) para el primer término, PAPER (papel) para el fondo y BORDER para el borde. Los mencionados comandos (o similares) pueden actuar de dos formas diferentes para el texto: en toda la pantalla o sólo en parte de la misma. Si su efecto repercute en la totalidad de la pantalla éste será idéntico al de COLOR. Cuando el efecto es sólo parcial no existe una correspondencia con el mencionado comando.

Un efecto parcial del cambio de color es el que no altera el texto escrito con anterioridad. Esto quiere decir que sólo se cambiará el color del texto que se escriba a continuación. Esta característica permite obtener zonas de la pantalla con distinto color de fondo o de primer término (o ambos). Así se consigue destacar uno o varios fragmentos de texto.

La utilización de los comandos de efecto parcial va pues relacionada con el uso del comando PRINT. En algunos equipos esta relación es tan estrecha que el comando de color se ha de situar en el argumento de PRINT. En tal caso, el cambio de color (de papel o tinta) puede adoptar la forma de código de control o carácter especial. Un posible ejemplo es el que se muestra a continuación.

```
50 PRINT "<ESC>M5 CURSO DE PROGRA-
   MACION"
```

Donde <ESC> representa la pulsación de la tecla ESCAPE u otra tecla especial. Los caracteres que se añaden tras este código indicarán la acción a realizar.

Si la formulación del cambio de color se asemeja a un comando podrá tomar un aspecto similar al siguiente.

```
50 PRINT INK 5;"CURSO DE PROGRA-
   MACION"
```

En este segundo caso se ve claramente que el efecto producido es el cambio del color de primer plano.

Otras posibilidades utilizables en la representación de texto son el video inverso y el parpadeo. El primero se refiere a la permutación del color de papel y de la tinta. Ello significa que el carácter se representará en el color del anterior fondo y el fondo en el del anterior primer término; esto es, invertidos. El otro efecto, el de parpadeo, produce el paso repetido de video normal a video inverso, lo que implica una serie de destellos que se repetirán con una frecuencia fija.

Los comandos que controlan estos dos efectos suelen situarse en el argumento de PRINT; habitualmente se formulan con las palabras INVERSE y FLASH (o similares).

COLOR

Establece el color de primer plano, fondo y borde de la pantalla:

Formato: COLOR <primer plano>,<fondo>,<borde>
Ejemplos: 20 COLOR 1,2,2
70 COLOR 4, 15, 5

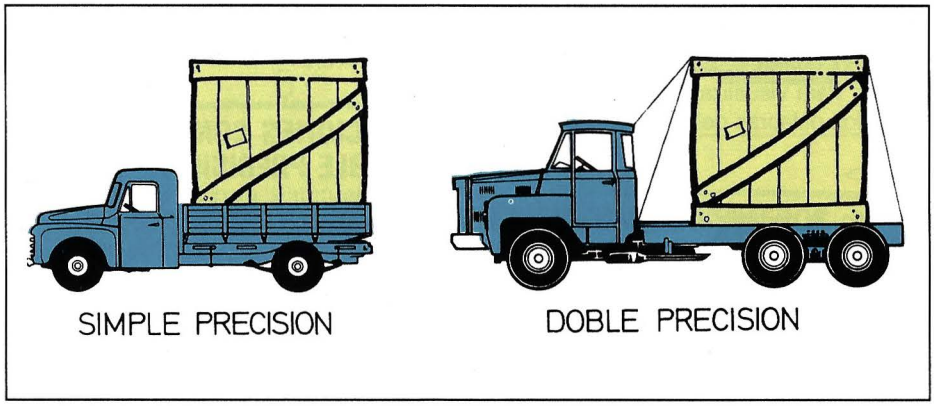
Forth (10)

Números de doble precisión y presentación de datos numéricos en pantalla

Cuando es necesaria una mayor precisión en los cálculos a realizar en FORTH, se puede recurrir al empleo de variables de doble precisión. Para representar un número de doble precisión hay que contar con dos elementos de pila; ello significa que un número de esta categoría se almacena en cuatro bytes de memoria (32 bits).

Estos números son relativamente difíciles de manejar. Si bien existen instrucciones adecuadas para realizar algunas operaciones con ellos, no es posible trasladarlos directamente a la pila; por lo demás, no existen en todas las versiones de FORTH palabras adecuadas para mostrarlos directamente por pantalla (aunque más tarde se estudiará un método que permite visualizarlos realizando una serie de operaciones con el PAD).

El almacenamiento en memoria de los números de esta índole se realiza en binario y fraccionados en dos partes. La primera coincide con un valor numérico que representa los 16 bits menos significativos del equivalente binario del número a introducir y, tras ésta, se introduce el valor correspondiente a los dieciséis bits más significativos.



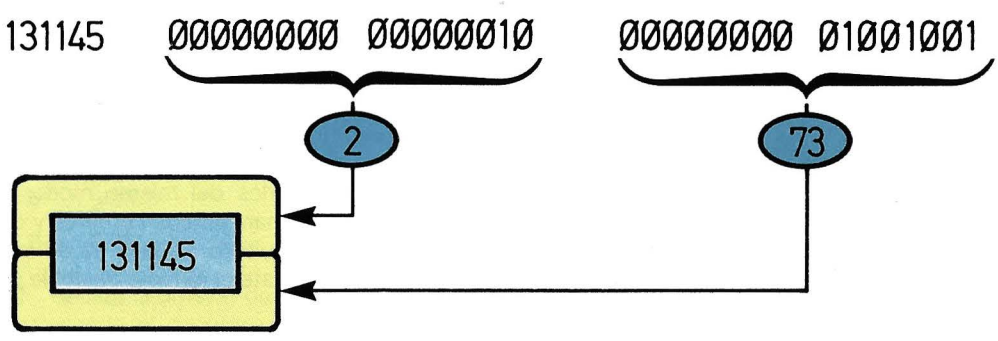
Cuando es preciso lograr una mayor exactitud en los cálculos en FORTH, es necesario recurrir a los números de doble precisión.

Como quiera que en principio, no siempre existe una palabra específica para mostrarlos por pantalla, es preciso recurrir a la palabra "."; ésta sólo permite la visualización de un elemento de la pila, luego habrá que ejecutar dos palabras de este tipo para leer un número de doble precisión. Para introducir el número 127 en formato de doble precisión, hay que proceder a la siguiente instrucción de datos:

```
127 0 <CR>
```

Una forma más eficiente de trabajar con estos números la proporciona el sistema hexadecimal. Al operar en este sistema, los dos elementos de la pila que se visualizan sí corresponden exactamente a las cifras que conforman el número, aunque en un orden un poco peculiar.

Un número de doble precisión se representa por medio de un número hexadecimal de 8 cifras; para introducirlo en el ordenador hay que empezar por las cuatro cifras situadas más a la derecha (las me-



Para introducir en el ordenador un número de doble precisión, hay que empezar calculando su equivalente binario. Acto seguido, se introducirá en primer lugar el número representativo de los 16 bits menos significativos, y luego el correspondiente a los 16 bits más significativos.

nos significativas) y, a continuación, se introducirán las cuatro de la izquierda (las más significativas). En el siguiente ejemplo, se asume que el sistema de numeración en el que se trabaja es el hexadecimal.
Suponga que se desea introducir el número:

25AC7645

Para ello habrá que depositar en la pila los dos números siguientes:

7645 25AC <CR>

Tras accionar el retorno de carro, el número en cuestión pasará a la memoria interna. Para visualizarlo será preciso ejecutar dos palabras del tipo "."; esto es:

.. <CR>

Tras ello, la pantalla mostrará los dos números hexadecimales que fueron introducidos en su momento:

```
.25AC .7645 OK
```

Los números negativos de doble precisión se almacenan en complemento a dos. Normalmente, cuando se trabaja con otros tipos de números, para almacenar un número negativo es suficiente con anteponer el signo, y el propio ordenador se encarga de realizar la complementación y almacenarlo en el formato adecuado. En este caso, como quiera que el ordenador no introduce el número sino que lo hace el propio usuario (quien debe introducirlo como dos números separados), también éste último se verá obligado a tener en cuenta el signo y complementar el número para obtener el resultado adecuado.
Así, para introducir el número −234 en formato de doble precisión, es necesario introducir la siguiente secuencia:

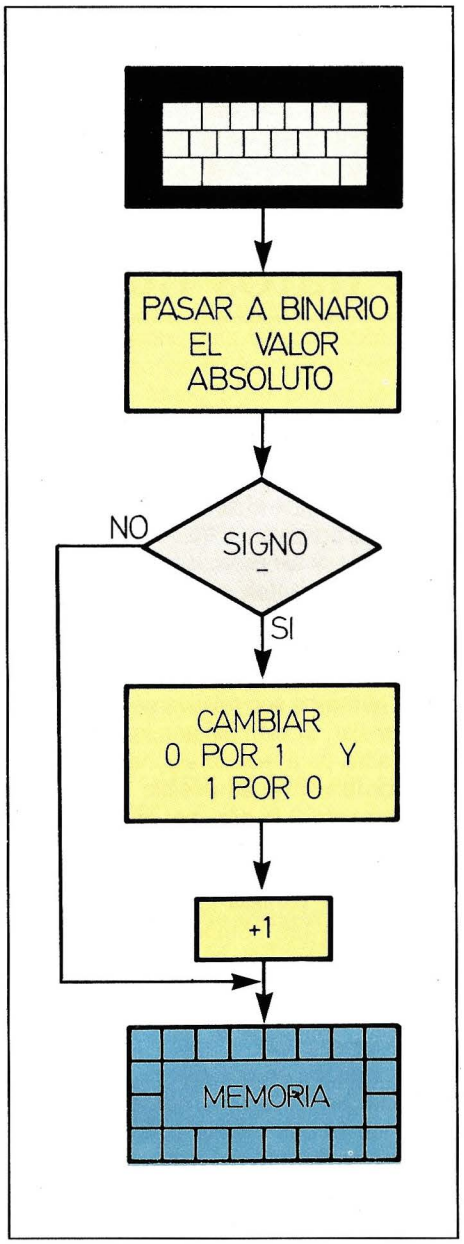
−234 −1 <CR>

OPERACIONES CON NUMEROS DE DOBLE PRECISION

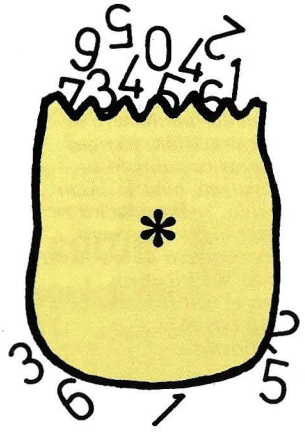
Existen una serie de operaciones en las que intervienen números de doble precisión, ya sea como operadores, constituyendo el resultado, o incluso como ambas cosas. A continuación se exponen las palabras más importantes relacionadas con los números de doble precisión.

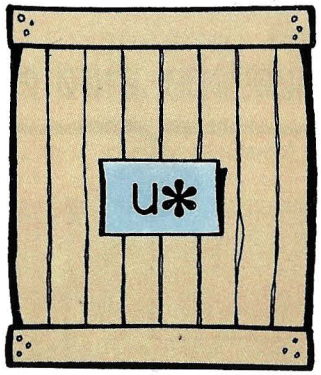
D+ Palabra que se emplea para sumar dos números de doble precisión.
Como el ordenador no es capaz de distinguir qué número de los que se encuentran en la pila son de doble precisión y cuáles no, entenderá que lo que debe hacer es tomar los dos elementos situados más cerca de la cima de la pila e interpretarlos como uno de los números y, a continuación, tomar los dos elementos siguientes e interpretarlos del mismo modo. Acto seguido sumará ambos números y depositará al resultado en la cima de la pila, adoptando éste el formato de doble precisión.
Por ejemplo, a la orden:

8 0 6 0 D+ . . <CR>



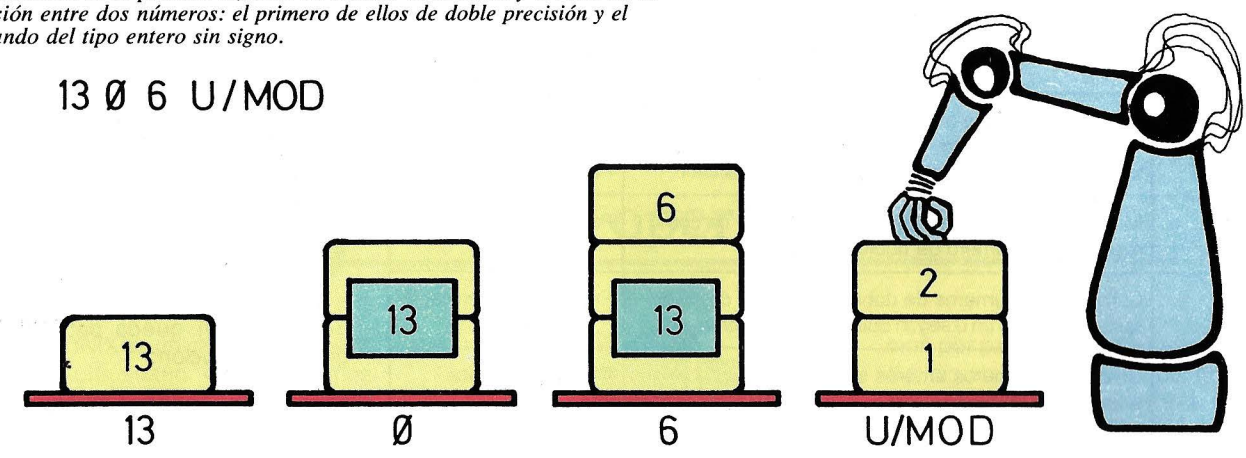
Al operar con números de doble precisión, el propio usuario debe calcular el complemento a dos de los números negativos a introducir en el ordenador. El diagrama de flujo refleja los pasos necesarios para obtener el complemento a dos de un número.





La necesidad de utilizar la palabra U deriva del hecho de que el producto de dos enteros sin signo suele exceder la capacidad de representación de valores de este tipo de números.*

La finalidad de la palabra U/MOD es obtener el cociente y el resto de la división entre dos números: el primero de ellos de doble precisión y el segundo del tipo entero sin signo.



el ordenador dará la siguiente respuesta:

```
8 0 6 0 D+ . 0 . 14 OK
```

DNEGATE Permite cambiar el signo de un número de doble precisión. Al ejecutar la instrucción:

```
3 0 DNEGATE . . <CR>
```

se obtendrá el siguiente resultado en pantalla:

```
3 0 DNEGATE . −1 . −3 OK
```

U* Se trata de una palabra que multiplica dos números; sin embargo, estos dos números no son de doble precisión, sino enteros sin signo. La existencia de esta peculiar orden se debe a que resulta frecuente que el produto de dos numeros enteros de simple precisión desborde la capacidad de representación de este tipo numérico. Por ejemplo:

```
3 4 U* . . <CR>
```

La respuesta del ordenador será

```
3 4 U* . 0 . 12 OK
```

D< Palabra FORTH que compara dos números de doble precision y comprueba si uno de ellos es menor que el otro Ejemplos

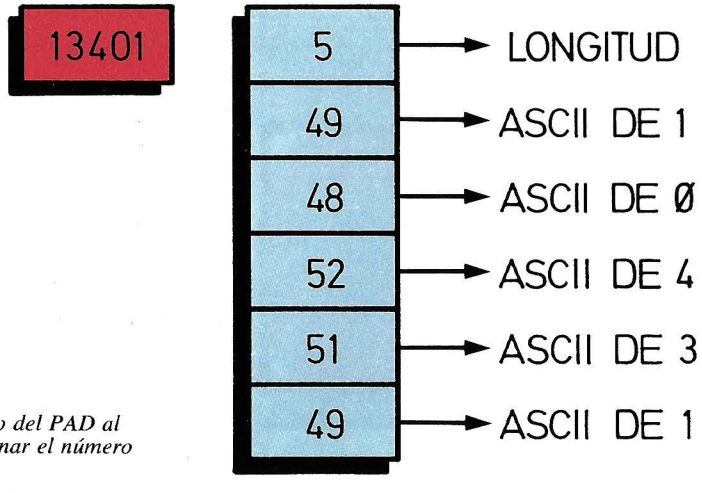
```
8 0 6 0 D< : <CR>
8 0 6 0 D< . 0 OK
```

U/MOD Realiza la división de dos números, entregando el cociente y el módulo de la división. Toma el primer elemento de la pila como entero sin signo, y agrupa los dos siguientes para sintetizar un número de doble precisión. Es una variante de la palabra /MOD que trabaja con enteros sin signo, con la salvedad de que el dividendo es ahora un número de doble longitud.
Ejemplos:

```
13 0 6 U/MOD . . <CR>
13 0 6 U/MOD . 2 . 1 OK
```

IMPRESION DE NUMEROS EN PANTALLA

La filosofía fundamental del formateado de números para su visualización en pantalla consiste en pasar los dígitos al PAD en orden inverso, para más tarde visualizarlo con el formato deseado. Estos números han de ser de doble precisión y positivos.
Existen una serie de palabras FORTH especializadas en el transporte de los números de la pila al PAD; éstas son las siguientes:



Aspecto del PAD al almacenar el número 13401.

TABLA DE ORDENES FORTH

PALABRA	DESCRIPCION	TIPO
D+	Permite sumar dos números de doble precisión.	Palabra aritmética
DNEGATE	Cambia el signo de un número de doble precisión.	Palabra aritmética
U*	Multiplica dos números enteros sin signo y da como resultado un número de doble precisión.	Palabra aritmética
D<	Comparados números de doble precisión y da como resultado un 1 ó un 0 según que uno sea mayor que otro.	Palabra de comparación
U/MOD	Divide dos números enteros sin signo, proporcionando el resto y el cociente.	Palabra aritmética
<#	Inicia el paso de dígitos al PAD.	Manejo del PAD
#<	Termina el paso de dígitos al PAD.	Manejo del PAD
#	Pasa un dígito al PAD.	Manejo del PAD
#S	Pasa todos los dígitos que queden al PAD.	Manejo del PAD
HOLD	Guarda un carácter ASCII en el PAD.	Manejo del PAD
ASCII	Deja en la pila el código ASCII del carácter siguiente.	Manejo de caracteres
SIGN	Pasa al PAD el signo de un entero situado en la cima de la pila.	Manejo del PAD

<# Empieza a pasar el número al PAD.

#> Termina de pasar el número al PAD; elimina el entero de doble precisión de la pila y coloca en ella la dirección y longitud del PAD.

#→# Extrae un dígito del número y lo lleva al PAD.

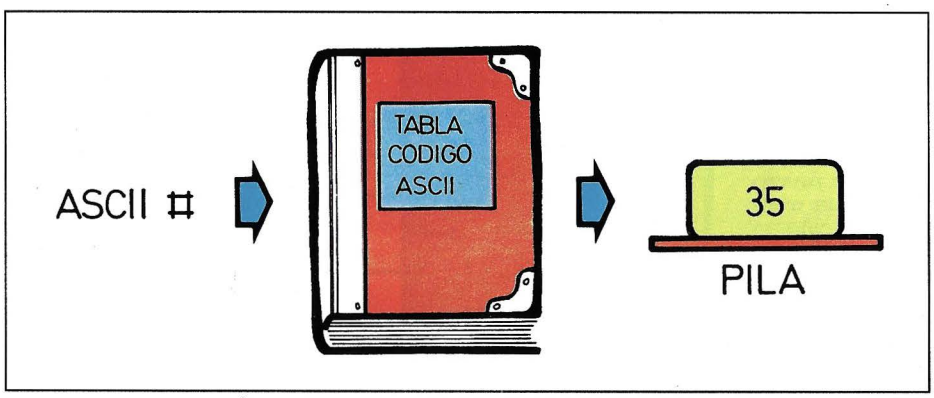
#S Extrae todos los dígitos hasta que el siguiente sea cero.

HOLD Guarda un carácter ASCII en el PAD.

ASCII Deposita en la pila el código ASCII del siguiente carácter.
Por ejemplo:

```
ASCII # .<CR>

ASCII # . 35 OK
```

A continuación se define una palabra que trabaja adoptando este sistema:



ASCII es una palabra FORTH que busca en la tabla de códigos ASCII el correspondiente al carácter que se especifique; dicho código será depositado en la pila.

```
: NUM
5 0
DO
CR 0 <###
####>TYPE
LOOP
;
```

Su actuación queda patente en la siguiente ejecución:

```
2 3 4 12 12 NUM <CR>

00012
00012
00004
00003
00002 OK
```

Desde luego, esta rutina parte del supuesto de que en la pila se encuentran cinco números que se desean visualizar; si no fuera así, el proceso conduciría a error.
La palabra SIGN permite pasar al PAD el signo de un número entero residente en la pila. Mediante esta nueva palabra FORTH, se puede crear la palabra "D.", la cual permite visualizar en pantalla un número de doble precisión con su signo correspondiente:

```
: D.
DUP >R DUP 0<
IF
   DNEGATE
THEN
<# #S R> SIGN #>
TYPE
;
```

Al ejecutarla:

```
−12 −1 D. <CR>
```

se obtiene la siguiente respuesta:

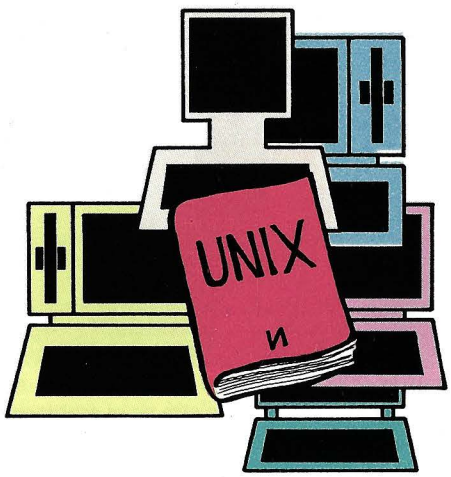
```
−12 −1 D. −12 OK
```

UNIX (1)

En busca de un sistema operativo estándar

El UNIX es un sistema operativo multiusuario (soporta varios usuarios actuando al mismo tiempo sobre el ordenador) y multitarea (ejecución de más de un programa a la vez) diseñado inicialmente para miniordenadores. No obstante, debido a su facilidad de uso, independencia del hardware y adaptabilidad a las exigencias de cada usuario, el UNIX está penetrando cada vez más en el mundo de los microordenadores

UN POCO DE HISTORIA



La fácil compatibilidad del UNIX con casi cualquier tipo de ordenador, lo convierte en un firme candidato a sistema operativo estándar.

El sistema operativo UNIX se gestó a finales de los años sesenta en los laboratorios de la Bell Telephone, pertenecientes a las compañías norteamericanas American Telephone and Telegraph y Western Electric, de reconocido prestigio en el mundo científico

Un programador llamado Ken Thompson, no muy satisfecho con el sistema operativo de que disponía (Multics) —aunque en aquellos años éste era uno de los primeros sistemas operativos interactivos, arrastraba bastantes secuelas de sus antepasados orientados a procesos por lotes—, decidió escribir su propio sistema operativo.

Inicialmente, el UNIX fue escrito en el lenguaje ensamblador del miniordenador PDP-7 de Digital Equipment Corporation.

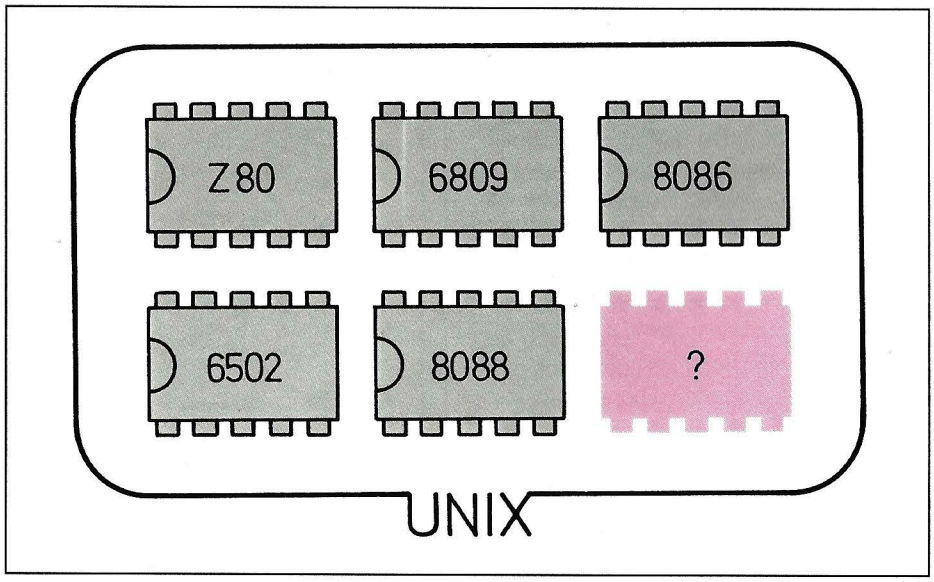
En el año 1971, el UNIX fue trasladado al más famoso ordenador de la gama PDP, el PDP-11, reescribiendo parte del sistema operativo en un nuevo lenguaje, el B (precursor del actual lenguaje C).

Ese mismo año llegó la versión del UNIX a otro programador de los laboratorios Bell, Dennis Ritchi, padre del lenguaje C, que junto con Ken Thompson tradujo enteramente el UNIX a este lenguaje.

Dada la imposibilidad de su comercialización por parte de la AT&T, esta firma decidió distribuirlo con fines puramente filantrópicos entre los Colegios y Universidades que lo solicitaran; a cambio de un pago simbólico, estas entidades recibían una cinta con el sistema operativo UNIX.

Semejante decisión causó principalmente dos efectos de distinto signo. El primero fue la rápida extensión y uso del UNIX al entrar en contacto con multitudes de estudiantes y laboratorios de investigación, lo que contribuyó a que se convirtiera en uno de los sistemas operativos más conocidos dentro del mundo científico. El segundo efecto se concreta en la gran diversidad de versiones que han ido surgiendo a partir del UNIX primigenio; ello tiene su razón principal en el hecho de que no existió una única mano que dirigiese su desarrollo, así como a la gran facilidad que presenta el UNIX para recibir nuevas aplicaciones.

Para combatir este mare mágnum de versiones, y aprovechando su separación de



Como quiera que el UNIX es un sistema operativo escrito en lenguaje de alto nivel, resulta plenamente independiente del microprocesador que constituya la CPU del equipo.

los laboratorios Bell en el año 1984, AT&T ha lanzado al mercado la versión de UNIX que pretende ser la estándar: UNIX versión V.

LAS BAZAS DEL UNIX

- *Transportabilidad*

La implantación del sistema operativo UNIX en cualquier ordenador queda garantizada por el hecho de que está escrito en un lenguaje de alto nivel: el C. Ello significa que no está ligado a una única familia de microprocesadores, como ocurre con los sistemas operativos escritos en ensamblador; tal es el caso del CP/M o del MS/DOS.

Los equipos encuadrados en el rango que va desde los microordenadores a los grandes ordenadores, están perfectamente capacitados para poder soportar el sistema operativo UNIX sin complicación alguna. Basta con un tiempo aproximado de dos o tres semanas para introducir el UNIX en cualquier ordenador de nuevo cuño.

No hace falta recalcar la importancia que este hecho tiene en el actual mundo de la informática, en el que cada par de semanas aparece un nuevo modelo de ordenador, cada par de meses un nuevo chip especializado y cada par de años una nueva familia de microordenadores.

- *Software potente*

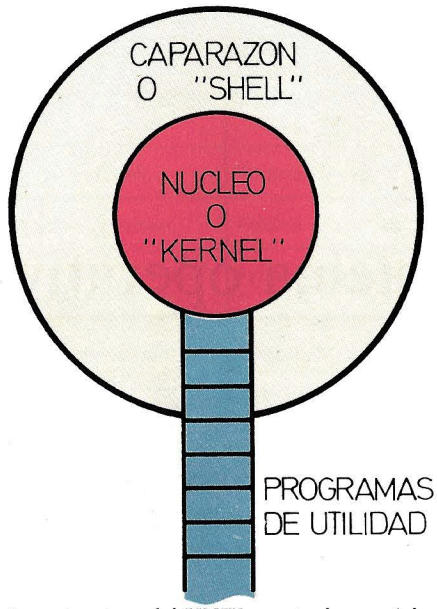
Los programas que dan cuerpo al sistema operativo UNIX están divididos funcionalmente en los siguientes grupos:

— El núcleo o "kernel"; en él residen los programas que efectúan las tareas propias del sistema operativo y gestionan el almacenamiento y recuperación de la información.

— El caparazón o "shell", encargado de reconocer e interpretar los comandos dirigidos al sistema operativo por el usuario.

— Programas de utilidad que realizan funciones de ayuda al mantenimieno del sistema y facilitan el trabajo del usuario.

En el caso del UNIX, la abundancia y calidad de los programas de utilidad lo convierten en uno de los sistemas operativos dotados de un mejor software de respaldo. Pueden contabilizarse hasta un total de 200 programas de utilidad; no es exagerado afirmar que raro es el programa de utilidad que necesite el usuario para el que el UNIX no ofrezca al menos dos variantes.



La estructura del UNIX consta de un núcleo, encargado de realizar las tareas propias del sistema operativo, un caparazón que establece la comunicación con el usuario, y multitud de programas de utilidad.

Así mismo, al disponer de facultades multilaterales y multiusuario, el UNIX permite desarrollar un entorno capaz de afrontar aplicaciones complejas.

- *Adaptabilidad al usuario*

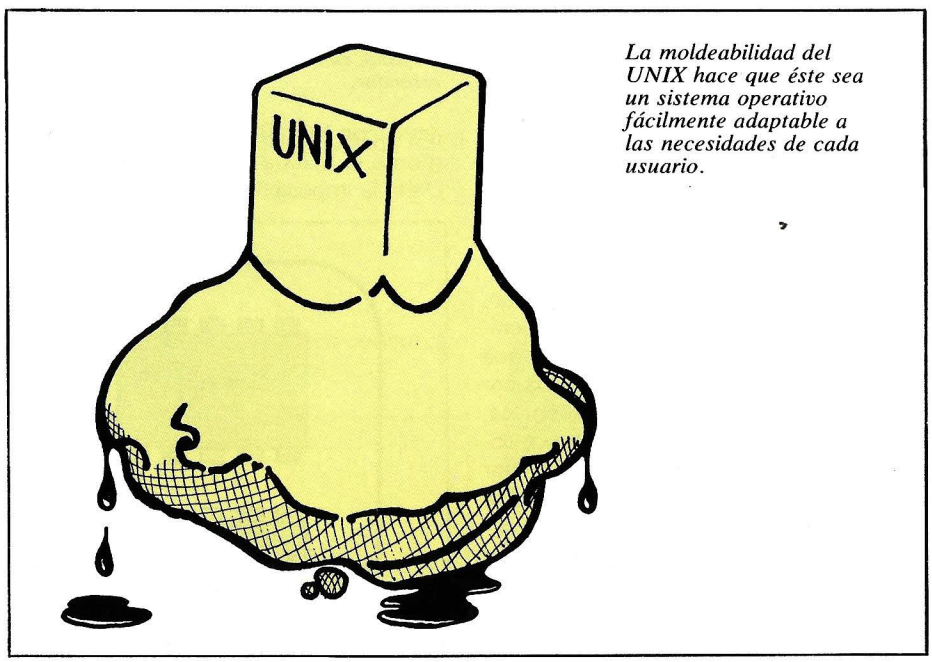
Aunque un dominio completo del UNIX no se alcanza en poco tiempo, debido a la extensión de campos que abarca, sí es posible no obstante que un usuario que no haya entrado nunca en contacto con él sea capaz de manejarlo en un alto porcentaje de sus capacidades en un tiempo record. En efecto, el UNIX está especialmente diseñado para que la zona del sistema que comunica con el usuario, "shell", pueda ser modificada fácilmente. Ello permite acondicionar el sistema operativo a medida del usuario, estableciéndose así una comunicación amena y fácil entre éste y el sistema operativo.

- *Información estructurada*

Una de las bases más sólidas de la potencia del UNIX reside en su estructura de ficheros. Bajo el control del UNIX todo tipo de información puede ser tratada como un fichero; ficheros que son manejados todos ellos de forma idéntica.

Como resultado de esta regularidad en el tratamiento de los ficheros, es posible el hecho de que el contenido de un fichero o de salida de un programa puedan ser enviados automáticamente a otro fichero o programa. El nombre que recibe esta conexión es "pipe" o tubería; un término que resalta muy a las claras la actividad de canalizar la información.

Volviendo a los ficheros, cabe mencionar que éstos están organizados en una es-



La moldeabilidad del UNIX hace que éste sea un sistema operativo fácilmente adaptable a las necesidades de cada usuario.

tructura jerárquica en forma de árbol invertido. El elemento que ocupa la cúspide y del cual se ramifican los demás recibe el nombre nodo o raíz o directorio raíz. Los dos tipos de ficheros existentes en esta organización son los ficheros propiamente dichos y los directorios: estos últimos son ficheros especiales que contienen la información que precisa el sistema relativa a todos los ficheros que se encuentran bajo su dominio.
Esta estructura está especialmente diseñada para poder identificar con sencillez a los elementos de información inmersos en el amplio marco procesable por medio del UNIX.

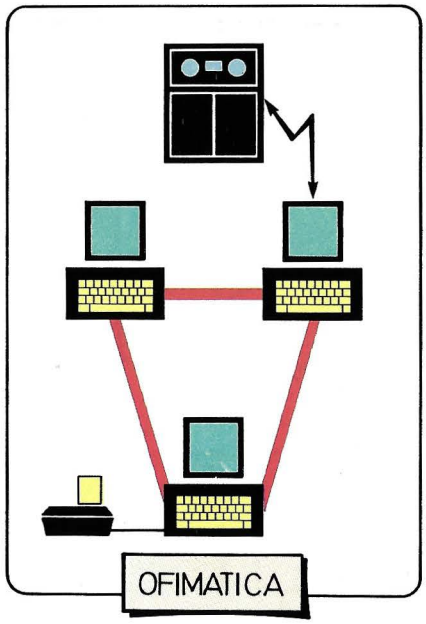
ALGUNOS PUNTOS DEBILES

La característica más relevante del UNIX hay que buscarla en su plena disposición a convertirse en la inteligencia básica de casi cualquier modelo del ordenador. Tal facultad tiene su origen en el hecho de estar escrito en un lenguaje de alto nivel que lo hace independiente de la intimidad de la máquina.

Aunque, en su conjunto, el UNIX es un sistema operativo muy completo y destacable en la mayoría de sus aspectos, pre-

La ofimática

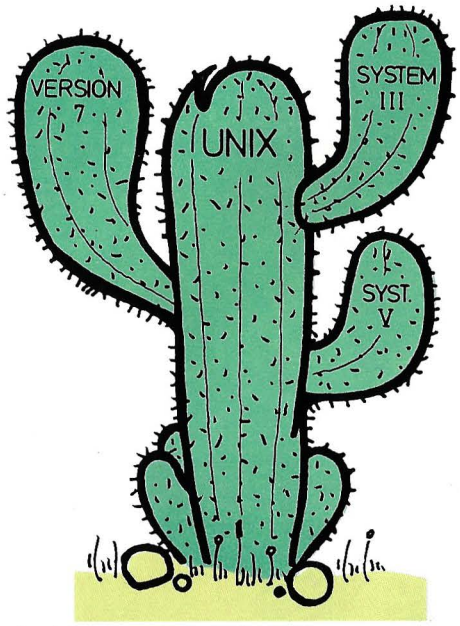
El fenómeno de desplazamiento desde una economía industrial hacia una economía de la información es bien conocido en todo el mundo, especialmente en estos últimos años. Las empresas, al igual que los gobiernos dependen cada vez en mayor grado de los sistemas utilizados para recoger, procesar, analizar y diseminar la información de forma rápida y eficaz.
Estas nuevas necesidades han afectado profundamente a la filosofía de la oficina tradicional, naciendo un nuevo concepto de entender el trabajo en estos lugares: la ofimática.
Con la ofimática se mejora la transferencia de información, reemplazando a los elementos clásicos (papel, bolígrafo, máquinas de escribir, teléfonos, etc.) por equipos informáticos que aumentan la productividad gracias a la mayor velocidad y eficiencia del trabajo desarrollado.
Una oficina automatizada no es —en contra de lo que cabría imaginar— un lugar repleto de máquinas, sino que debe entenderse como un entorno que permite a los empleados utilizar la interconexión de los equipos electrónicos para realizar tareas administrativas, de gestión, organizativas y analíticas; y todo ello aprovechando la facilidad ofrecida por estos equipos para el tratamiento de la información.



La ofimática está cambiando rápidamente la fisonomía de la oficina tradicional, convirtiéndola en un potente centro de información.

Con frecuencia, una compañía que cambia sus máquinas de escribir por procesadores de texto es capaz de doblar su productividad. Si embargo, la generación de papel por métodos más rápidos y en mayores cantidades no conduce necesariamente a mejores resultados. Para que la aplicación de la ofimática resulte efectiva es necesario establecer una labor previa de formación del personal de la oficina, para que se acople a los nuevos equipos y procedimientos, y sin que ello suponga un trauma.

Las futuras oficinas tendrán un aspecto muy distante a las actuales. Los terminales de ordenador serán tan familiares como hoy lo pueden ser las máquinas de escribir o los teléfonos. Estos terminales permitirán un empleo de procesadores de texto y darán acceso a bases de datos. La información, tanto gráfica como en forma de texto, será transmitida de unos puestos a otros y almacenada en soportes masivos, tales como discos magnéticos u ópticos, reemplazando a los archivadores repletos de papel. Por último, los diversos equipos estarán interconectados en redes locales, y tendrán acceso a grandes redes remotas que abrirán una inmensa vía de comunicación en el mundo exterior.



La falta de una dirección única que encaminara el desarrollo del UNIX, ha provocado la existencia actual de múltiples versiones, algunas de ellas incompatibles entre sí.

senta ciertos inconvenientes originados, principalmente, por su génesis en el campo científico y de enseñanza. Los que siguen son sus óbices más relevantes.

● *Proliferación de versiones*
El hecho de que el UNIX se difundiera masivamente sin estar sujeto a una compañía que centralizara su evolución y modificaciones, ha contribuido a que el desarrollo del sistema haya seguido vías muy diversas. De ahí la carencia de compatibilidad total entre sus dialectos, lo que supone un obstáculo para la aplicación del software creado bajo distintas versiones. Cuando el UNIX llegó al mercado, AT&T disponía de varias versiones: Sexta Edición, Versión 7, PWB/UNIX, UNIX System III y UNIX System V. Para complicar más aún el panorama, la mayoría de las versiones comerciales están basadas en el UNIX System III, mientras que la firma AT&T está promoviendo el UNIX System V.
Este ejemplo es un buen reflejo del galimatías de versiones existentes. En todo caso, la reciente salida del UNIX al mercado comercial está obligando a optar por una versión específica, la System V, con la cual desarrollar el software de aplicación.

● *Relativa escasez de software comercial*
Como quiera que, en los primeros años, la mayoría de las versiones del UNIX se orientaban a la enseñanza y a las aplicaciones científicas, las aplicaciones comerciales quedaron desatendidas; principalmente, en los temas relativos a tipos de ficheros y al acceso concurrente a la información en ellos contenida, de forma que no hubiese interferencia entre los diversos usuarios. Dada esta situación, las compañías de software comercial han tenido que añadir sus propias extensiones para cubrir importantes aspectos.

● *Consumo del C.P.U.*
Su concepción inicial, orientada a miniordenadores, hace que el UNIX esté acostumbrado a disponer de abundantes recursos de máquina, con lo que su implantación en sistemas de reducida potencia puede saturar en cierto grado a la CPU.
La tendencia actual a utilizar microprocesadores cada vez más potentes, hace que este inconveniente se atenúe día a día.

Las características inherentes al UNIX permiten lograr uno de los anhelos de todo usuario: evitar que el software quede obsoleto en un corto intervalo de tiempo.

RESUMEN

A título de conclusión cabe destacar la ventaja importantísima de su transportabilidad y el hecho de permitir el uso del mismo software con nuevas generaciones de ordenadores. En cuanto a la multiplicidad de versiones del UNIX, cabe esperar que la entrada en el mercado de la firma AT&T imponga una norma a seguir e impulse un rápido desarrollo de programas de aplicación que llenen el hueco actual.
La opinión de analistas experimentados y constructores de ordenadores, apuntan en la dirección de que el UNIX tiene grandes posibilidades de convertirse, en los años venideros, en el sistema operativo estándar de los micro y miniordenadores, especialmente en el ámbito de estos últimos.

Paquetes integrados

La síntesis de las aplicaciones horizontales

Dentro de esta sección de la obra se han estudiado fundamentalmente cuatro tipos de aplicaciones horizontales: hojas electrónicas, bases de datos, tratamientos de textos y gráficos de gestión. Otro grupo importante, dentro del software de aplicación horizontal, es el formado por los programas de comunicación, destinados a facilitar el diálogo entre distintos sistemas informáticos.

En el presente capítulo da comienzo el estudio de un nuevo tipo de aplicación, que sintetiza en un mismo paquete a varios de los programas apuntados anteriormente.

EL SOFTWARE INTEGRADO

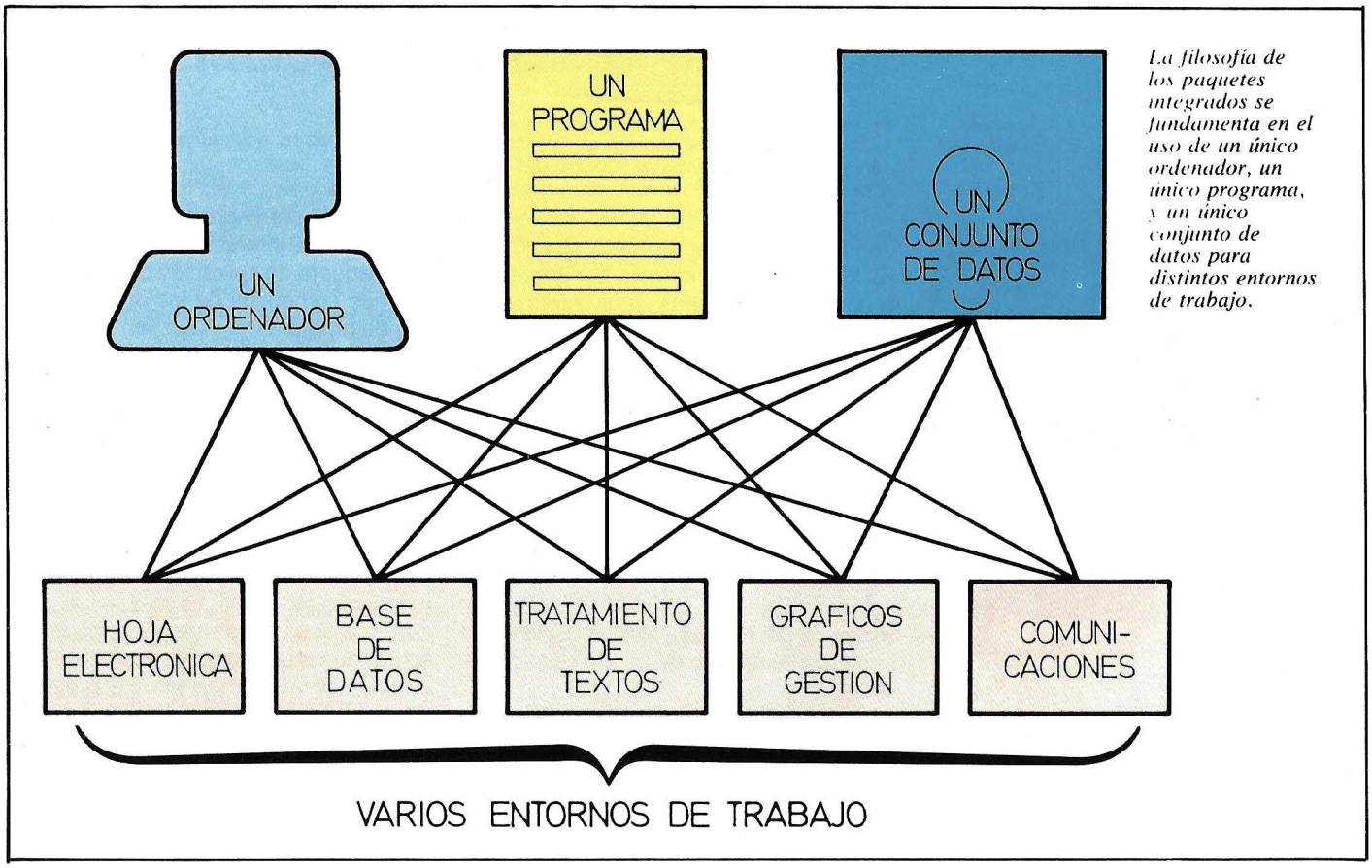
Cada uno de los cinco grandes grupos en los que cabe catalogar a los programas de gestión y productividad utilizables en ordenadores personales tiene una misión específica, tal y como se resume a continuación:

- Hojas electrónicas

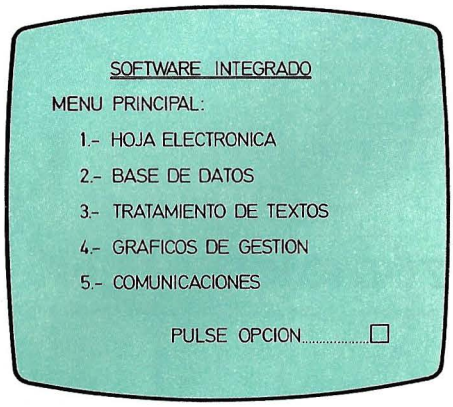
Fundamentalmente se encargan de facilitar los cálculos numéricos. Para ello ofrecen una técnica de programación extraordinariamente sencilla, basada en una estructura matricial, que permite que el propio usuario prepare los modelos oportunos.

- Bases de datos

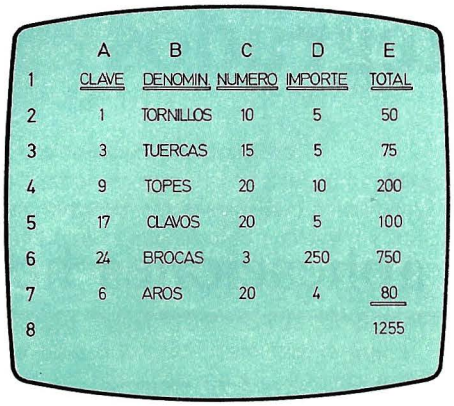
Su objetivo consiste en gestionar y automatizar el manejo de bases de datos o ficheros de información. Para ello, deben



La filosofía de los paquetes integrados se fundamenta en el uso de un único ordenador, un único programa, y un único conjunto de datos para distintos entornos de trabajo.



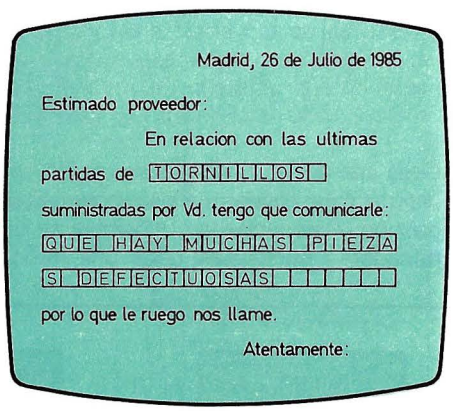
Para facilitar la "navegación" del usuario a través de los distintos entornos, los paquetes integrados suelen estar organizados en base a un sistema de menús.



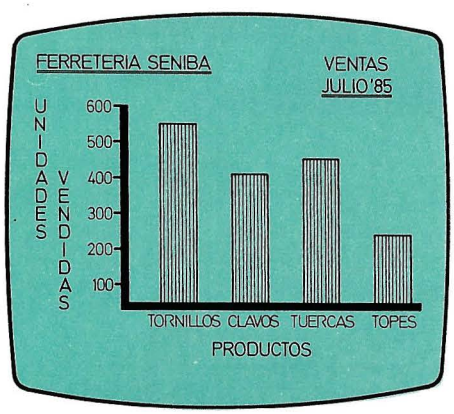
	A	B	C	D	E
1	CLAVE	DENOMIN.	NUMERO	IMPORTE	TOTAL
2	1	TORNILLOS	10	5	50
3	3	TUERCAS	15	5	75
4	9	TOPES	20	10	200
5	17	CLAVOS	20	5	100
6	24	BROCAS	3	250	750
7	6	AROS	20	4	80
8					1255

En el ejemplo propuesto, el empleo de la hoja electrónica permitirá mecanizar la producción de facturas.

El entorno asociado a la base de datos se utiliza en nuestro ejemplo para la gestión del almacén.



La mecanización de la correspondencia en nuestra ferretería será una tarea encomendada al entorno de proceso de textos.



Los gráficos de gestión permitirán resumir con claridad y sencillez la información relativa a las ventas.

permitir cuatro operaciones elementales: introducción de nueva información en la base de datos (ALTAS), borrar información ya existente (BAJAS), modificar algunos de los datos existentes en la base (CAMBIOS), y obtener informes por impresora o por pantalla, a partir de la información residente en la base de datos (INFORMES).

- Procesadores de textos

Las hojas electrónicas ofrecen al usuario la posibilidad de procesar cómodamente números, los programas para el tratamiento de textos persiguen un objetivo complementario: procesar cómodamente palabras. En esencia, estos programas asumen todas las funciones típicas de una máquina de escribir, con las facilidades suplementarias que derivan del uso de un ordenador.

- Gráficos de gestión

En términos generales, las aplicaciones informáticas de gestión tienen una finalidad común: ayudar al usuario a manejar información. Las posibles formas en las que ésta se puede representar son muy variadas; entre ellas, por su sencillez y fácil interpretación, cabe destacar las representaciones gráficas. El objetivo de las aplicaciones de gráficos de gestión consiste en representar en sencillos diagramas ciertos datos alfanuméricos, de forma que queden patentes sus propiedades fundamentales.

- Comunicaciones

El último gran grupo dentro de los programas de aplicación para ordenadores personales, lo constituyen los encargados de establecer comunicaciones entre distintos ordenadores, o incluso entre un ordenador y otros dispositivos del mundo exterior.

Una vez resumidas las características de cada uno de los cinco grandes grupos de aplicaciones horizontales, estamos ya en disposición de definir qué se entinde por software integrado: se dice que un programa es un Sistema Integrado si se incluye la posibilidad de utilizar todos o la mayor parte de los cinco entornos de trabajo descritos anteriormente.

JUSTIFICACION DEL SOFTWARE INTEGRADO

En una primera impresión puede parecer escasamente interesante disponer de un solo paquete de aplicación que asuma los mismos cometidos de los otros cinco tipos de programas... ¿Qué ventajas se obtiene con ello? ¿Qué más da utilizar una base de datos integrada en un programa amplio, que una base de datos implementada en un programa autónomo?

Desde luego, si contemplamos la ejecución independiente en un único entorno, la respuesta a las preguntas anteriores no permite establecer ventajas importantes del software integrado sobre los programas independientes. Es más, en algunos casos, incluso puede resultar más interesante utilizar un programa específico, que ofrecerá alguna ventaja sobre el programa integrado al estar especializado y disponer de todos los recursos del ordenador para él solo.

No obstante, al estudiar el trabajo global desempeñado por una empresa o por un particular nos encontramos con un pro-

blema muy importante: la migración de los datos. En el caso de disponer de distintos programas de aplicación, cada uno de ellos manejará una información independiente de los restantes; de esta forma si, por ejemplo, los mismos datos que se almacenan en una base de datos se quieren utilizar en una hoja electrónica, el usuario deberá repetir la entrada dos veces: una para la aplicación de base de datos y otra para la aplicación hoja electrónica. Si el usuario dispone de un paquete de software integrado no será necesario que realice más que una única entrada de datos, ya que la información incluida en un entorno estará disponible desde cualquiera de los restantes.

Normalmente, las aplicaciones de software integrado tienen como auténtico "corazón" la hoja electrónica; a través de ella se realiza la introducción de los datos y, más adelante, estos mismos datos se utilizarán en distintos tipos de entornos; en el procesador de textos, si a partir de ellos se desean producir cartas u otros tipos de documentos, en el sistema gráfico si se desean obtener diagramas, etc.

Un paquete integrado sintetiza, en una misma aplicación, los entornos necesarios para automatizar íntegramente las tareas habituales en muchos ámbitos de actividad.

Evaluación de ordenadores personales

Cuando una empresa o un particular deciden adquirir un ordenador personal, suelen tropezar con la más que importante dificultad de decidir la marca y características de equipo más apropiado a sus necesidades.

Desde luego, no se pretende resolver este problema en un pequeño cuadro; no obstante, sí se pueden apuntar aquí algunas reglas que faciliten la toma de una decisión.

Los proveedores ofrecerán normalmente sistemas completos, incluyendo el hardware y el software básico para su funcionamiento. El comprador debe intentar hacer un análisis detallado y comparativo de forma que se evalúen los distintos conceptos por separado, y como síntesis de estas evaluaciones parciales se obtendrá una evaluación global. Algunos de los aspectos más significativos para la evaluación son los siguientes:

- CONDICIONES ECONOMICAS

Además del precio de adquisición del sistema, hay que tener en cuenta otros aspectos económicos como la financiación, la posibilidad de compra o alquiler, el leasing, etc.

- CAPACIDAD DE CRECIMIENTO

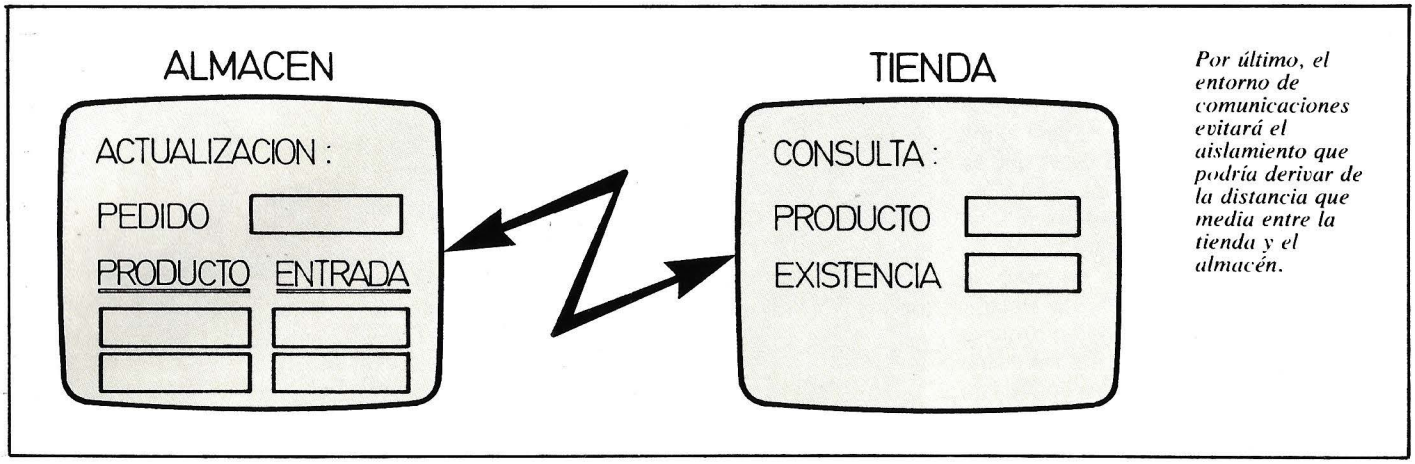
Casi con toda seguridad, con el paso del tiempo resultará necesario ampliar el equipo adquirido; por ello, resulta muy importante tener en cuenta la capacidad de crecimiento del sistema.

- RENDIMIENTO DEL EQUIPO

Teniendo en cuenta la máquina y los programas que permitirán explotarla, se debe medir en alguna unidad homogénea el rendimiento de los distintos equipos que se estén comparando.

- FIABILIDAD DEL PROVEEDOR

Para garantizar el buen funcionamiento de los productos adquiridos, es importante cerciorarse de que el proveedor es una entidad seria que garantiza el mantenimiento y conservación tanto de equipos como de programas.



Por último, el entorno de comunicaciones evitará el aislamiento que podría derivar de la distancia que media entre la tienda y el almacén.

UN SENCILLO EJEMPLO

Dado que el concepto de integridad resulta de vital importancia para comprender las ventajas aportadas por un paquete integrado, vamos a ilustrar con un sencillo ejemplo cómo podría organizarse la gestión de una pequeña empresa:
Supondremos que se trata de una ferretería, con un único punto de venta y un almacén situado en un lugar distinto al de la tienda. El objetivo consiste en mecanizar de forma integrada la facturación, la gestión del almacén, la emisión de cartas a proveedores, la obtención de informes gráficos sobre las ventas y la comunicación entre la tienda y el almacén.

- *Facturación*

El proceso de facturación se basará en dos operaciones aritméticas elementales: la primera consiste en multiplicar el número de unidades vendidas de un producto determinado por el importe unitario de venta de dicho producto; de esta forma se obtendrá el importe total a facturar por la venta de un producto. La segunda operación consiste en sumar los importes totales de todos los productos incluidos en una misma factura, para así obtener el importe neto a facturar.
Dado el carácter aritmético del problema, su entorno natural de resolución es una hoja electrónica. Si se utiliza un paquete de software integrado, el sistema tan sólo debe exigir al usuario la entrada de la clave del producto y del número de unidades vendidas, encargándose el programa de localizar (probablemente en la base de datos) el importe unitario por el que se multiplicará y la propia descripción del producto.

- *Gestión del almacén*

Precisamente en esta fase se debe facilitar al usuario el control de los productos con que trabaja. El objetivo, en este caso, consiste en permitir actualizar la información relativa a los productos almacenados, dando altas, bajas, modificaciones y visualizando el estado final.
Sin duda el entorno de trabajo será ahora la base de datos, ya que su especialidad coincide con las tareas descritas anteriormente.

- *Correspondencia con proveedores*

Ahora se trata de facilitar la producción de documentos, más o menos normalizados, que se utilizarán para la emisión de cartas. Lo más habitual es que el usuario disponga de varios modelos de cartas: uno para reclamaciones a proveedores, otro para solicitar un suministro, etc., en los que existan "huecos" libres donde se incluirá el concepto variable: la reclamación concreta, los productos solicitados, etc.
Evidentemente, en este caso resulta necesario utilizar el entorno que ofrece el procesador de textos.

- *Informes gráficos*

Para obtener diagramas comparativos sobre las ventas producidas se pueden elegir diversas opciones: mostrar unidades vendidas por cliente, productos vendidos por cliente, etc. En cualquier caso, los informes gráficos van a tener como misión principal mostrar de una forma muy sencilla el comportamiento de las ventas, dando apoyo para que el usuario tome decisiones: dejar de vender un producto, dejar de servir a un cliente, etc.
Resulta trivial dedicir que, en este caso, debe utilizarse como entorno de trabajo la opción de gráficos de gestión.

- *Comunicación tienda/almacén*

También se apuntó la necesidad de establecer una comunicación entre el almacén y la tienda, localizadas en puntos distantes. Esta comunicación puede servir, entre otras muchas cosas, para que un vendedor desde la tienda compruebe inmediatamente las existencias del producto solicitado por un cliente; también puede utilizarse en sentido contrario; el encargado del almacén puede visualizar las existencias en la tienda y de esta forma saber qué producto debe enviar.
En este caso resulta imprescindible utilizar un programa de comunicaciones.

CONCLUSIONES FINALES

Desde luego es posible resolver cada una de las cinco soluciones del ejemplo anterior con una aplicación distinta; no obstante, las relaciones existentes entre todos los supuestos hace recomendable utilizar un paquete integrado que facilite el trabajo con datos comunes.
En los tres próximos capítulos se acometerá el estudio de un producto de software integrado con el que se podría resolver eficientemente el problema: se trata del paquete SYMPHONY. Más adelante se analizarán otros paquetes integrados presentes en el mercado informático.

Archivos aleatorios (1)

Creación y uso de archivos de acceso directo

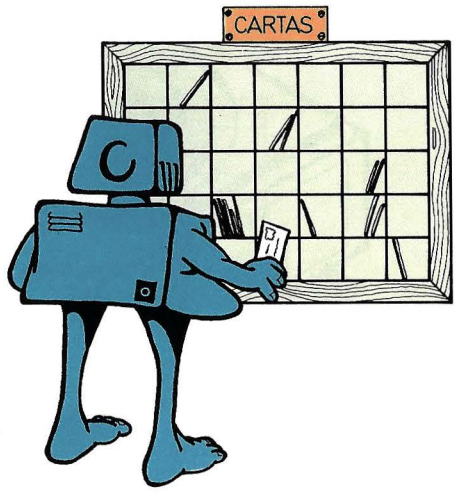
Los elementos constitutivos de un archivo secuencial (los registros) se graban uno detrás de otro, y para acceder a uno de ellos es preciso pasar por todos los anteriores. Evidentemente, esto plantea algunas dificultades a la hora de manejar la información contenida en ese tipo de archivos. Así pues, en muchos casos es necesario recurrir a los archivos de acceso directo que obvian algunas de estas dificultades. Dos de estas características ventajosas de los archivos de acceso directo son las siguientes:

- En primer lugar, en un archivo directo no es necesario grabar los registros uno tras otro, sino que se pueden grabar en la posición y en el orden que se desee.
- En segundo lugar, a la hora de acceder a uno de los elementos del archivo, no es necesario acceder previamente a todos los demás.

Un archivo en general es un conjunto de registros. En el caso de los archivos aleatorios, cada registro está formado por una serie de campos que también han de tener una longitud fija, de manera que el registro tendrá una estructura fija. Esta estructura del registro ha de ser elegida a la vista de la información que se va a grabar en el archivo. Dentro de estos campos, existe uno muy importante: la clave, que decide la posición relativa de cada registro dentro del archivo. Por lo general, la clave suele ser numérica (en el caso del BASIC siempre lo será). La clave puede equipararse a un índice que permite el acceso a un determinado registro.

El primer paso para construir un archivo de acceso directo es crear la estructura de los registros que contendrán la información. Así, si se desea crear un archivo con los números de teléfono y las direcciones de los amigos, será necesario que cada registro contenga, por ejemplo, las siguientes informaciones:

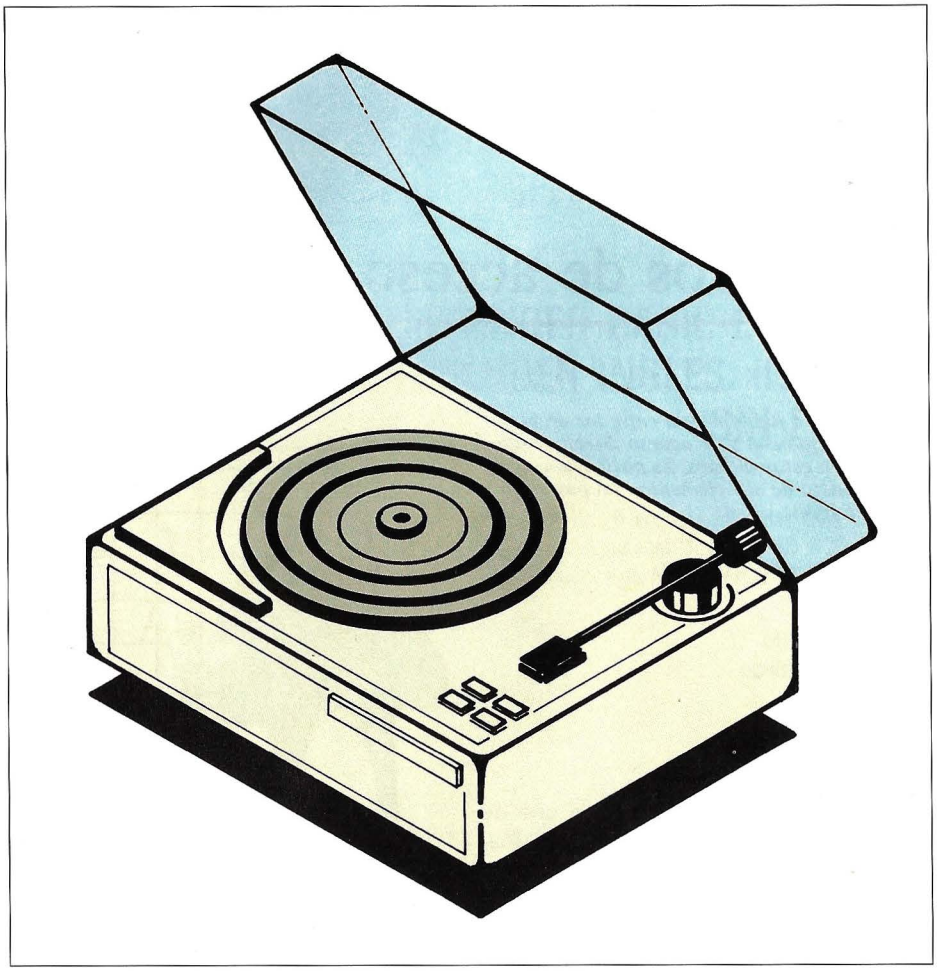
La diferencia fundamental entre los archivos de tipo secuencial y de acceso directo reside en que, en estos últimos, es posible acceder a cualquiera de sus elementos sin pasar por los anteriores.



Nombre
Primer apellido
Segundo apellido
Dirección
Ciudad
Teléfono

La longitud reservada para cada uno de estos campos ha de ser definida y permanecerá fija a lo largo de todo el proceso.

El casete es un soporte de almacenamiento externo capaz exclusivamente de albergar archivos de tipo secuencial.

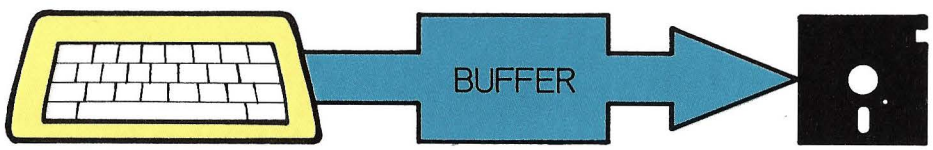


Al igual que ocurre con la reproducción de un disco musical, los archivos de acceso directo permiten el libre acceso a cualquiera de sus registros. Tomando la analogía del disco, el acceso a un registro equivale a posicionar la cabeza en cualquier parte de la superficie del mismo.

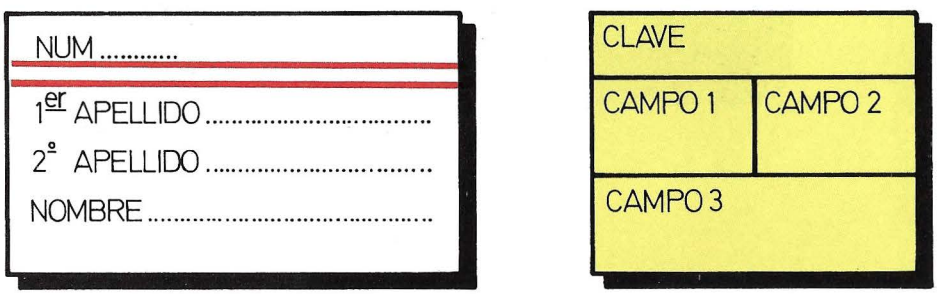
Esta longitud se elegirá de forma que quepa en el campo la información que se desea almacenar. Por ejemplo, para almacenar el campo nombre será suficiente con reservar 20 caracteres; la mayor parte de los nombres cabrán en ese sitio. Desde luego, en este punto puede ser preciso llegar a una solicitud de compromiso; no todos los nombres caben en 20 caracteres, pero si reservamos mayor espacio nos encontraremos con que la mayor parte de los nombres no llegarán a ocupar gran parte del espacio disponible y, por tanto, ese espacio se perderá. Hay que tener en cuenta que si reservamos mucho espacio para cada uno de los campos, el registro en su conjunto resultará mas voluminoso y, en consecuencia, cabrán menos registros en el soporte de memoria sobre el que reside el archivo. En el ejemplo propuesto, una distribución más o menos racional de las longitudes de los registros puede ser la siguiente:

Nombre	(20 caracteres)
Primer apellido	(15 caracteres)
Segundo apellido	(15 caracteres)
Dirección	(30 caracteres)
Ciudad	(10 caracteres)
Teléfono	(9 caracteres)

OPERANDO DESDE EL BASIC



El "buffer" actúa como intermediario entre los datos generados por el proceso y el soporte donde se almacenan.



Un registro es muy semejante a una ficha. Los espacios reservados en la ficha para los distintos conceptos reciben en los registros el nombre de campos.

Las operaciones que es necesario efectuar para crear uno de estos archivos desde el BASIC son las siguientes:

1) Abrir el archivo en modalidad de acceso directo. A diferencia de lo que sucedía con los archivos secuenciales, aquí no es preciso abrir el archivo en modo de lectura o de escritura. Una vez abierto el archivo, en él se pueden efectuar operaciones tanto de lectura como de escritura.

La siguiente instrucción opera abriendo el archivo en caso de que exista; y en caso de que no exista, lo crea:

```
OPEN "R",#n1,"<nombre>",n2
```

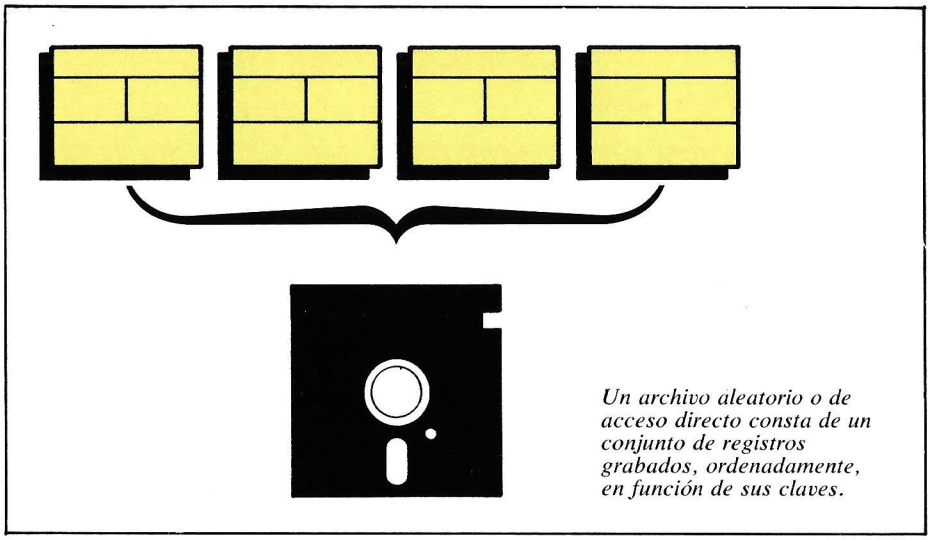
Al comando OPEN le siguen una serie de parámetros que es necesario especificar. En primer lugar, la R entre comillas indica

que se trata de un archivo de acceso directo. A continuación, n1 indica el número de canal que se desea abrir. El nombre que se especifique a continuación es el del archivo con el que se desea trabajar. Por último, n2 indica la longitud total del archivo.
Existen limitaciones en cuanto al tamaño máximo del registro, pero éstas dependen del sistema con el que se esté trabajando.
2) Especificar la estructura de cada registro (esta operación no es necesaria en todas las versiones del BASIC).

FIELD #n1, n3 AS <var1>, n4 AS <var2>, ...

Mediante esta instrucción se define el campo asociado a un canal determinado, que se identifica mediante el número n1. A continuación se indica la estructura mediante la enumeración de las variables asociadas a cada campo: var1, var2, etc., siendo precedidas estas variables por un número que indica el espacio que se les desea reservar.
Un detalle importante a tener en cuenta es que en un archivo secuencial todos los campos que componen el registro han de ser alfanuméricos. El hecho de que estos campos hayan de ser obligatoriamente alfanuméricos no quiere decir que en un archivo no se pueden grabar números, sino que sobre ellos habrá que realizar las transformaciones adecuadas para convertirlos en cadenas de caracteres.
3) Por lo general, los datos que se van a introducir en el archivo no tienen la misma longitud que los campos que les fueron reservados. Esta es la razón de que sea preciso realizar una adaptación de los datos a los campos donde se alojarán dentro del archivo. Este ajuste puede realizarse mediante dos instrucciones existentes a tal efecto. Estas instrucciones se encargan de ajustar los caracteres bien a la derecha o bien a la izquierda del campo disponible, y rellenan el resto del campo con espacios en blanco. Por lo general, se suele emplear la instrucción LSET con cadenas de caracteres alfanuméricos, mientras que RSET se suele emplear para cadenas de caracteres numéricos.
La transformación de un número en una cadena de caracteres es fácil de realizar. De hecho, ya se conoce una función capaz de realizar esa operación: se trata de la función STR$. Sin embargo, esta función, que en principio funciona correctamente, puede ser mejorada. Hay que observar qe

Los archivos de acceso directo exigen soportes de memoria que permitan un acceso aleatorio a la información almacenada; tal es el caso de las unidades de disco flexible.



Un archivo aleatorio o de acceso directo consta de un conjunto de registros grabados, ordenadamente, en función de sus claves.

OPEN

Realiza la apertura de un archivo en un dispositivo de almacenamiento externo.

Formato: OPEN "R",#<n1>,<nom>,<long>

Ejemplos: 10 OPEN "R",#1,"DATOS",123
50 OPEN "R",#2,"ARCHIVO.DAT",50

FIELD

Especifica el formato del registro

Formato: FIELD#<n1>,<long1> AS <var1> ,...

Ejemplos: 20 FIELD#1, 20 AS A$, 10 AS B$
70 FIELD#2, 5 AS TEL$, 15 AS NOM$

LSET/RSET

Ajustan los datos a izquierda o derecha, respectivamente, dentro del campo especificado.

Formato: LSET <campo>=<variable>
RSET <campo>=<variable>

Ejemplos: 20 LSET A$=TEL$
70 RSET B$=NOM$

PUT

Sitúa un registro de clave especificada en un archivo de acceso directo.

Formato: PUT # <n1>, <reg>

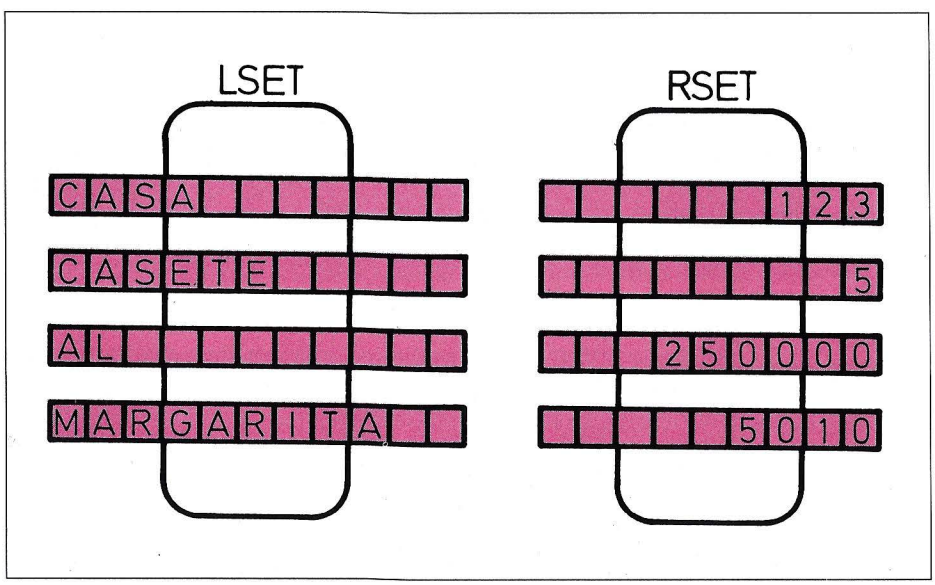
Ejemplos: 20 PUT #1, REG%
70 PUT #2,15

GET

Lee el registro especificado de un archivo de acceso directo.

Formato: *GET* #<n1>,<reg>

Ejemplos: 10 GET #1, REG%
50 GET #2,25



Mediante las funciones LSET y RSET, es posible ajustar los datos a izquierda o derecha, respectivamente, dentro de los campos del registro.

un número entero se almacena en la memoria del ordenador ocupando dos bytes, mientras que el campo preciso para almacenar ese mismo número en un archivo mediante la función STR$ debería tener 5 caracteres.
Para evitar este desperdicio del espacio de almacenamiento de los soportes externos, dispone de una función que permite almacenar un número de simple precisión en una cadena, empleando para ello tan solo dos bytes. El sistema seguido es el mismo que se emplea para almacenar ese número en memoria. La función a utilizar es MKI$.
Existen también funciones equivalentes para almacenar números de simple precisión en memoria mediante cuatro bytes (MKS$) y números de doble precisión mediante ocho bytes (MKD$).
Por supuesto, existen funciones para realizar la operación inversa, funciones que obtienen el número primitivo a partir de la cadena generada por MKI$, MKS$ y MKD$. Estas funciones serán estudiadas más adelante al explicar como se puede leer un archivo.
La formulación de LSET es la que sigue:

LSET <var. dest.>=<var. origen>

En donde:

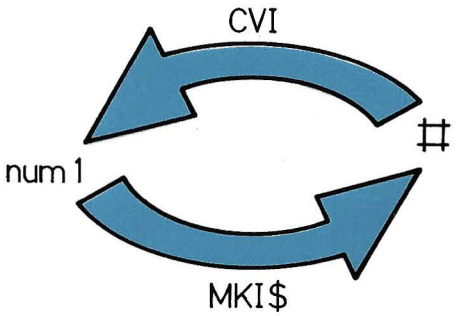
<var. dest.>: Variable donde se desea que la cadena quede ajustada.
<var. origen>: Variable donde se encuentra la cadena original.

4) Introducir los datos en el buffer, de donde pasarán en el momento oportuno al archivo en disco.
La posición relativa dentro del archivo queda definida por POS%, que es la clave del registro y que en este caso es numérica. Esta clave ha de ser un número entero. Si existen más datos a introducir, se ha de volver al punto 3; si no, se pasa al siguiente punto.

PUT #n1,POS%

5) Cerrar el archivo que se ha empleado. Esta operación es muy importante ya que los elementos del archivo se van grabando en el buffer y van pasando al disco en el momento oportuno. La operación de cierre del archivo consiste en disociar el canal del archivo, transfiriendo antes el contenido del buffer al archivo.

CLOSE #1 ó END



Las funciones MKI$ y CVI permiten convertir un número en una cadena de caracteres, y viceversa, facilitando su almacenamiento en un espacio mínimo.

CREACION DE UN ARCHIVO

Un ejemplo de creación de un archivo de este tipo puede ser el que se detalla en los próximos párrafos.
La primera operación se realiza en la línea 1000. Concretamente, en este caso se abre el canal 1; el nombre del archivo será DATOS y la longitud reservada para cada registro es de 98 bytes:

```
1000 OPEN "R", #n1, "DATOS", 98
```

A continuación, es necesario especificar la estructura de los registros que se van a emplear.

```
1010 FIELD #n1, 20 AS NOM$, 15 AS AP1$,
15 AS AP2$, 30 AS DIR$, 10 AS CIU$#, 9 AS
TEL$
```

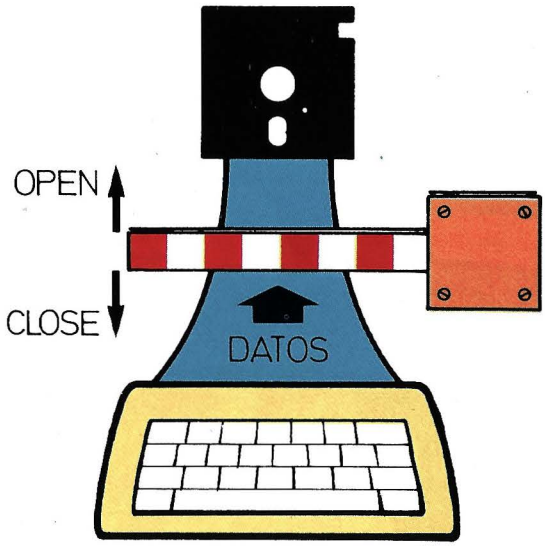
Con la línea 1020 se pretende leer el número de registro en el que será grabada la información que se introducirá más adelante:

```
1020 INPUT "QUE REGISTRO ";REG%: IF
     REG%=0 THEN CLOSE #n1: END
```

Para salir del proceso basta con responder a la orden INPUT que el registro que se desea grabar es el registro 0; en tal caso se cerrará el archivo y se detendrá el proceso.
Evidentemente, antes de almacenar los datos en el archivo es preciso disponer de dichos datos. Por lo general, estos datos suelen proceder del exterior y son contados a través de la ejecución de instruccio-

```
1000 OPEN "R", #n1, "<nombre>", n2
1010 FIELD #n1, 20 AS NOM$, 15 AS AP1$,
     15 AS AP2$, 30 AS
DIR$, 10 AS CIU$, 9 A$ TEL$
1020 INPUT "QUE REGISTRO ";REG%: IF
     REG%=0 THEN CLOSE#n1: END
1030 INPUT "NOMBRE: ";N$
1040 INPUT "APELLIDO 1: ";A1$
1050 INPUT "APELLIDO 2: ";A2$
1060 INPUT "DIRECCION :";D$
1070 INPUT "CIUDAD :";C$
1080 INPUT "TELEFONO :";T#
1100 LSET NOM$=N$
1110 LSET AP1$=A1$
1120 LSET AP2$=A2$
1130 LSET DIR$=D$
1140 LSET CIU$=C$
1150 RSET TEL$=MKD$(T#)
1200 PUT #1,REG%
1210 GOTO 1020
```

PROGRAMA 1: listado del programa ejemplo para la creación e introducción de datos en un archivo.



Los comandos OPEN y CLOSE abren y cierran, respectivamente, los canales de comunicación entre el ordenador y un archivo en disco.

CVI

Convierte una cadena que representa un número en formato comprimido en uno de precisión entera.

Formato: CVI (<cadena>)

Ejemplos: 20 N=CVI(N$)

CVD

Convierte una cadena que representa un número en formato comprimido en uno de doble precisión.

Formato: CVD(<cadena>)

Ejemplos: 20 P=CVD(LL$)

CVS

Convierte una cadena que representa un número en formato comprimido en uno de simple precisión.

Formato: CVS(<cadena>)

Ejemplos: 20 KG=CVS(MIL$)

nes de tipo INPUT. También podrían derivar de procesos ejecutados con anterioridad, pero este caso es menos frecuente. Lo que sí sucede a veces essque la información recogida es tratada antes de ser almacenada en el archivo.
Aquí está, en definitiva, el bloque de instrucciones para la captación de los datos:

```
1030 INPUT "NOMBRE: ";N$
1040 INPUT "APELLIDO 1: ";A1$
1050 INPUT "APELLIDO 2: ";A2$
1060 INPUT "DIRECCION :";D$
1070 INPUT "CIUDAD :";C$
1080 INPUT "TELEFONO :";T#
```

A continuación, es preciso ajustar los valores que se desean grabar en el archivo al formato disponible:

```
1100 LSET NOM$=N$
1110 LSET AP1$=A1$
1120 LSET AP2$=A2$
1130 LSET DIR$=D$
1140 LSET CIU$=C$
1150 RSET TEL$=MKD$(T#)
```

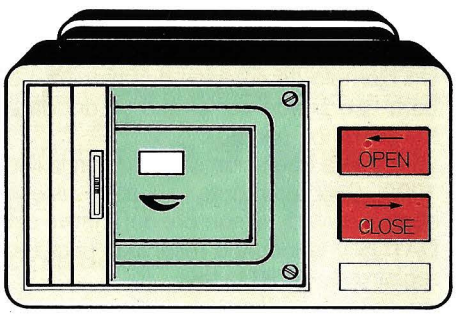
Tras ello, puede ya grabarse la información relativa al registro que nos ocupa en el archivo:

```
1200 PUT #1,REG%
```

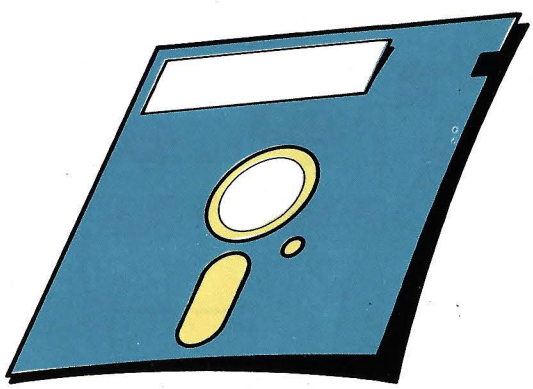
Por último, puede regresarse a la linea 1020 para tratar más entradas de datos:

```
1210 GOTO 1020
```

Junto al texto se reproduce el listado completo del programa ejemplo, destinado a la creación e introducción de datos en el archivo.



La apertura y el cierre de un archivo creado en un dispositivo de almacenamiento externo corre a cargo, respectivamente, de las órdenes OPEN y CLOSE.



La operación con archivos en disco permite al ordenador almacenar y manipular con eficacia grandes cantidades de información.

LECTURA DE UN ARCHIVO DE ACCESO DIRECTO

Son cuatro los pasos a seguir para la lectura de los requisitos de un archivo de acceso directo o aleatorio; justamente, los que se detallan a continuación.
1) Abrir el archivo.
La apertura es análoga a la que se realiza al crear el archivo. Ante todo, al abrir el archivo para leer datos del mismo es preciso que este exista, esto es, que haya sido creado previamente. El nombre utilizado ha de ser el correcto y la longitud del registro ha de ser la misma con la que fue creado.

```
OPEN "R", #n1, " nombre ", n2
```

TABLA DE CONVERSION

ORDENADOR	OPEN	FIELD	AJUSTE	ESCRITURA/LECTURA		CONVERSION		APERTURA
	OPEN "R",≠n1, <nom>,n2	FIELD≠n1, n3AS<var1>..	LSET/RSET	PUT≠n1, <num>	GET≠n1, <num>	CVI/CVS/ CVD	MKI$/MKS$/ MKD$	OPEN<nom> [FOR<modo>] AS[≠]n1 [LEN=n2]
APPLE II (1) (APPLESOFT)	OPEN<nom>, Ln1,Sn2,Dn2,Vn4	—	—	WRITE nom, R1	READ Nom, R1	—	—	—
APRICOT (M-BASIC)	OPEN "R",≠n1, <nom>,n2	FIELD≠n1, n3AS<var1>...	LSET/RSET	PUT≠n1, <num>	GET≠n1, <num>	CVI/CVS/ CVD	MKI$/MKS$/ MKD$	OPEN<nom> [FOR<modo>] AS[≠]n1 [LEN=n2]
ATARI	OPEN≠n1,aux1, aux2,nom	—	—	—(2)	—(2)	—	—	—
CBN 64 (3)	—	—	—	—	—	—	—	—
DRAGON	—	—	—	—	—	—	—	—
EQUIPOS MSX	—	FIELD≠n1, n3AS<var1>...	LSET/RSET	PUT≠n1, <num>	GET≠n1, <num>	CVI/CVS/ CVD	MKI$/MKS$/ MKD$	OPEN<nom> [FOR<modo>] AS[≠]n1 [LEN=n2]
HP-150	OPEN "R",≠n1, <nom>,n2	FIELD≠n1, n3AS<var1>...	LSET/RSET	PUT≠n1, <num>	GET≠n1, <num>	CVI/CVS/ CVD	MKI$/MKS$/ MKD$	OPEN<nom> [FOR<modo>] AS[≠]n1 [LEN=n2]
IBN PC	OPEN "R",≠n1, <nom>,n2	FIELD≠n1, n3AS<var1>...	LSET/RSET	PUT≠n1, <num>	GET≠n1, <num>	CVI/CVS/ CVD	MKI$/MKS$/ MKD$	OPEN<nom> [FOR<modo>] AS [≠]n1 [LEN=n2]
MPF	—	—	—	—	—	—	—	—
NCR DM-V (MS-BASIC)	OPEN "R",≠n1, <nom>,n2	FIELD≠n1, n3AS<var1>...	LSET/RSET	PUT≠n1, <num>	GET≠n1, <num>	CVI/CVS/ CVD	MKI$/MKS$/ MKD$	— —
NEW BRAIN	—	—	—	—	—	—	—	—
ORIC	—	—	—	—	—	—	—	—
SHARP MZ-700 (MZ-BASIC)	—	—	—	—	—	—	—	—
SINCLAIR QL	—	—	—	—	—	—	—	—
SPECTRAVIDEO	—	—	LSET/RSET	PUT≠n1, <num>	GET≠n1, <num>	CVI/CVS/ CVD	MKI$/MKS$/ MKD$	OPEN<nom> [FOR<modo>] AS[≠]n1 [LEN=n2]
ZX-SPECTRUM	—	—	—	—	—	—	—	—

(1) Para manejar estos comandos es preciso utilizar un CTRL+D e introducirlos en los argumentos de sucesivas instrucciones PRINT.
(2) Los archivos aleatorios en el ATARI con unidad de disco no funcionan de la forma comentada en el texto con los comandos PUT y GET, sino que para leer un determinado registro de un archivo, es necesario posicionarse sobre él previamente y, más tarde, realizar la operación de lectura mediante INPUT≠O de escritura con PRINT≠.
(3) Se comentará en un posterior capítulo de la obra.

MKI$

Convierte un número de precisión entera en una cadena en formato comprimido.

Formato: MKI$(<num>)

Ejemplos: 10 LET A$=MKI$(A)

MKS$

Convierte un número de precisión simple en una cadena en formato comprimido.

Formato: MKS$(<num>)

Ejemplos: 10 LET H$=MKS$(HS)

MKD$

Convierte un número de precisión doble en una cadena en formato comprimido.

Formato: MKD$(<num>)

Ejemplos: 10 LET K$=MKD$(PESO)

```
1000 OPEN "R", #1, "DATOS", 99
1010 FIELD #1, 20 AS N$, 15 AS A1$,
     15 AS A2$, 30 AS D$, 10 AS C$, 9 AS T$
1020 INPUT "QUE REGISTRO ";REG%:
     IF REG%=0 THEN CLOSE#1: END
1030 GET#1, REG%
1130 PRINT "NOMBRE: ";N$
1140 PRINT "APELLIDO 1: ";A1$
1150 PRINT "APELLIDO 2: ";A2$
1160 PRINT "DIRECCION :": D$
1170 PRINT "CIUDAD :";C$
1180 PRINT "TELEFONO :"; CVD(T$)
1210 GOTO 1020
```

PROGRAMA 2: programa para la lectura de los datos introducidos en el archivo creado por medio del programa 1.

2) Especificar la estructura de cada registro.
El registro ha de tener la misma estructura que tenía cuando el archivo fue creado. Lo único que ha de mantenerse es la estructura, ya que los datos están grabados en el archivo en registros de ese tipo. Sin embargo, los nombres de las variables asignadas a los campos del buffer pueden ser distintas:

FIELD#n1, n3 AS <var1>, n4 AS var2, ...

3) Leer el registro deseado del archivo, con lo que dicha información quedará a la disposición del usuario para manejarlo según sus necesidades:

GEN#n1, REG%

Los datos alfanuméricos, como ya se ha dicho, están directamente a disposición del usuario una vez que el registo es extraído del archivo. Pero no hay que olvidar que los datos numéricos fueron tratados para convertirlos en alfanuméricos, por lo que ahora será preciso realizar la operación inversa. Si la función empleada para la conversión fue STR$, bastará ahora con aplicar la función VAL. Pero en el caso de que la función utilizada fuese una de las siguientes: MKI$, MKS$ o MKD$, sería preciso recurrir a las funciones CVI, CVS o CVD, respectivamente. Su cometido en el que sigue:

CVI Convierte una cadena en un número de precisión entera.
CVS Convierte una cadena en un número de simple precisión.
CVD Convierte una cadena en un número de doble precisión.

Téngase en cuenta que la cadena ha de ser convertida al mismo tipo de número que constituía su origen. Es decir, si se convierte un número de precisión entera en cadena mediante el empleo de la función MKI$, no es posible reconvertir la cadena en número mediante la función CVS ni mediante la función CVD. Tan solo la función CVI nos permitirá obtener el número del que se partió.
4) Cerrar el archivo para dejar libre el canal correspondiente:

CLOSE#n1

El programa ejemplo que figura junto al texto (programa 2), puede utilizarse para leer los datos que fueron introducidos en el archivo mediante el programa 1.

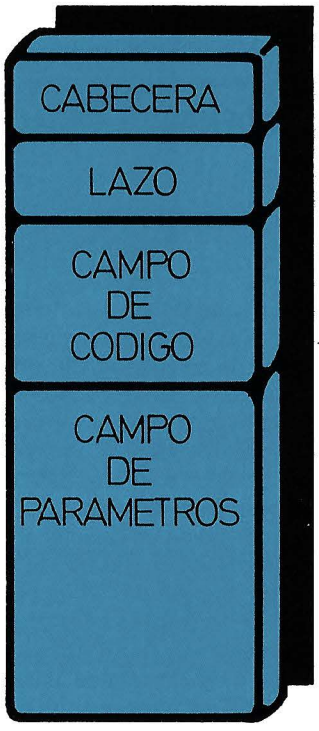
Forth (11)

Cadenas y Arrays

En FORTH existen distintos tipos de palabras. Hasta el momento se han empleado tres tipos: las que se definen mediante la palabra "dos puntos" (:), las constantes que se definen mediante CONSTANT y las variables que se definen por medio de VARIABLE. Existen muchos más tipos de palabras, ya que el usuario puede definir tipos de palabras para su propio uso. Dentro de semejante diversidad caben, por ejemplo, palabras para almacenar cadenas de caracteres y arrays.
Las palabras que componen el diccionario FORTH tienen una serie de puntos comunes:

1. Cada palabra tiene un nombre que sirve para identificarla.
2. Cada palabra está encadenada a las demás. Todas las palabras se basan para su definición en palabras ya existentes.
3. Cada palabra tiene una información (campo de código) que indica cuál ha de ser la acción a realizar cuando se ejecute.
4. En su definición, las palabras incluyen información adicional (el campo de parámetros).

El nombre, el lazo y el campo de código tienen el mismo formato para todas las palabras y se agrupan en la zona de cabecera.



Estructura de una palabra incluida en el diccionario FORTH.

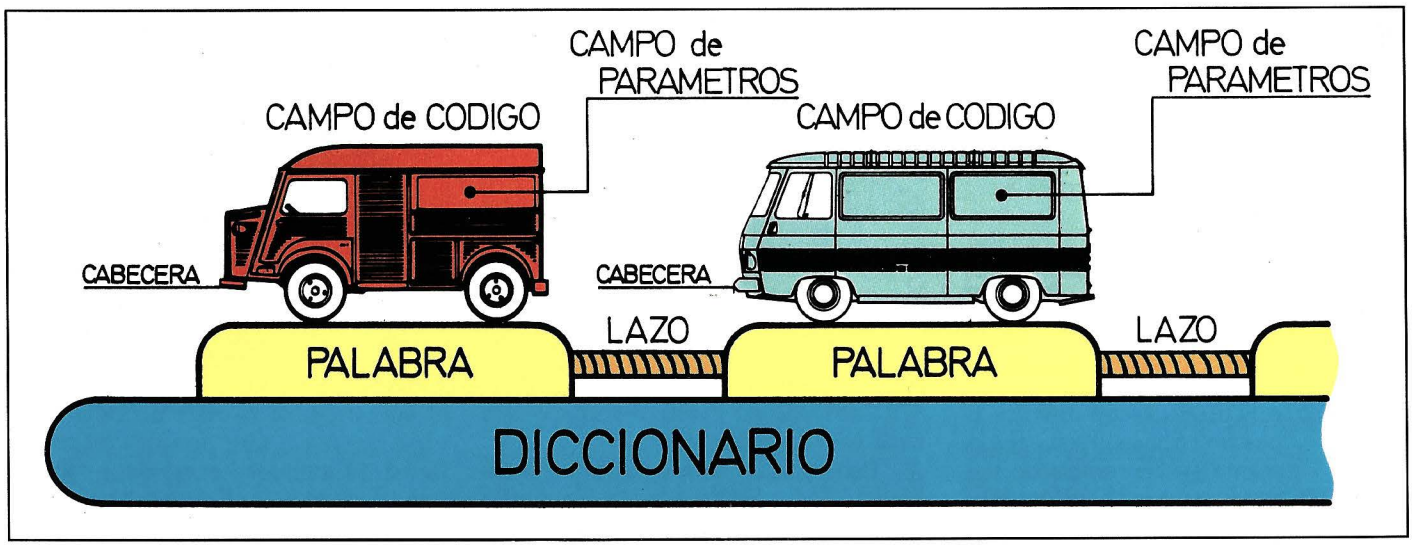
El contenido de la zona de parámetros varía de uno a otro tipo de palabra. La forma de emplear los referidos parámetros depende del tipo de la palabra, el cual se especifica mediante el campo de código.

LOS "ARRAYS" EN FORTH

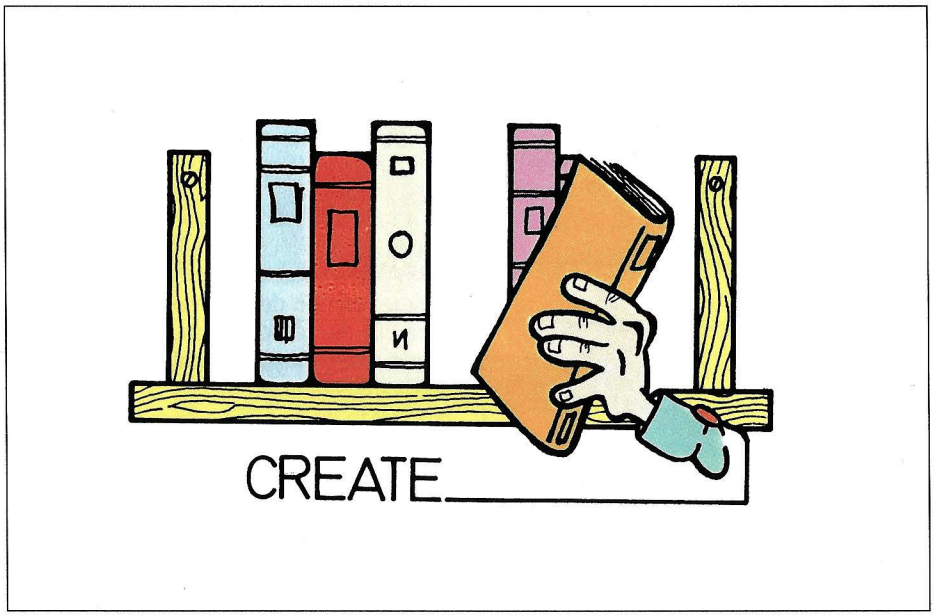
La forma más simple de crear un nuevo tipo de palabra consiste en hacer uso de CREATE. Y, desde luego, es posible construir un campo de parámetros específicos para ella. Su campo de código especificará que cuando esa nueva palabra se ejecute, sea depositada en la pila la dirección de comienzo de su campo de parámetros. De esta forma, el programador puede recoger tal dirección para emplear el campo de parámetros a plena voluntad.
Para construir la cabecera se emplea la palabra CREATE. Suponga que se desea crear un array numérico que almacenará la duración en días de cada uno de los meses del año.

```
CREATE MESES <CR>
```

Para construir el campo de parámetros se dispone de la palabra auxiliar ",", cuyo cometido es extraer un número de la pila y guardarlo en el diccionario, reservando para ello dos bytes adicionales que añade al diccionario.
En esencia se trata de ir situando los elementos que formarán el ARRAY en la zona de parámetros de la palabra.
A continuación se muestra la forma de introducir el ejemplo propuesto en el ordenador:



Cada uno de los componentes de una palabra FORTH tiene una misión específica; su síntesis da cuerpo a la palabra en sí.



Para incorporar nuevas palabras al diccionario FORTH, puede utilizarse la palabra clave CREATE.

```
31 , 28 , 31 , 30 , 31 , 30 , 31 , 31 , 30 , 31 , 30 , 31
<CR>
      31 , 28 , 31 , 30 , 31 , 30 , 31 , 31 , 30 , 31,
      30, 31 OK
```

Para leer un elemento se va a definir la palabra MES. A la hora de emplearla hay que situar el número del elemento que se desea leer en la pila y, posteriormente, invocar a la mencionada palabra:

```
: M ES
1– DUP+ M ESES
+    .
;
```

Suponga que se desea leer el segundo elemento:

```
2 M ES <CR>
```

La respuesta que se obtendrá será la siguiente:

```
2 MES 28 OK
```

CADENAS DE CARACTERES

Las cadenas de caracteres se almacenan en casi todos los lenguajes como secuencias de códigos ASCII; en FORTH se utiliza el campo de parámetros de una palabra para almacenar los códigos que componen la cadena. Como los códigos ASCII ocupan un solo byte cada uno, es necesario emplear la palabra "C," para transferir el código de un carácter al campo de parámetros de la palabra. "C," es la versión para un byte de ",": deposita un número en el diccionario, pero emplea para ello un byte.

Por ejemplo, puede utilizarse la palabra ALLOT para reservar un número de bytes para el campo de parámetros.

Para introducir una cadena en una palabra

es posible emplear la siguiente palabra FORTH:

```
: CADENA
32 WORD DUP 1+
SWAP C  OVER + SWAP
DO
  | C  C,
LOOP
;
```

Veamos con detalle cuál es la actuación de cada uno de los elementos que intervienen en la definición de la nueva palabra CADENA:

: Comienzo de la definición de la palabra.

CADENA Nombre identificador de la palabra.

32 Sitúa en la pila el valor 32.

WORD Interpretará el valor 32 que se encuentra en la pila como el código ASCII del carácter que ha de tomar como separador; en este caso se trata de un espacio en blanco. A continuación, WORD toma el contenido del buffer de entrada y lo coloca en el PAD, sitúa la longitud de la palabra al comienzo del PAD y la dirección del PAD en la pila.

DUP Duplica el número situado en la cima de la pila que corresponde a la dirección de comienzo del PAD.

1+ Suma uno a la cima de la pila.

SWAP Intercambia los dos valores de la cima de la pila, con lo cual el elemento más cercano la cima de la pila es la dirección de comienzo del PAD, mientras que el segundo coincide con esta dirección incrementada en una unidad.

C Lee el primer elemento del PAD (ya que esa es la dirección que se encuentra en la cima de la pila) que coincide con la longitud.

OVER Copia el segundo elemento de la pila y lo deposita en la cima de la misma.

+ Suma los dos primeros elementos de la pila: la longitud de la cadena y la dirección del comienzo del PAD.

SWAP Intercambia los dos elementos más cercanos a la cima de la pila, con lo que quedan preparados los índices para el array (estos son la dirección final de la cadena en el PAD y la dirección de comienzo).

A partir de aquí aparece un bucle que toma como inicio la dirección donde se encuentra el primer elemento del PAD y como final el último elemento del PAD; extrae uno a uno estos elementos y los transfiere al campo de parámetros de la palabra que se esté definiendo.

Para definir una palabra, como ejemplo ASURBANIPAL, basta con introducir la siguiente secuencia de palabras:

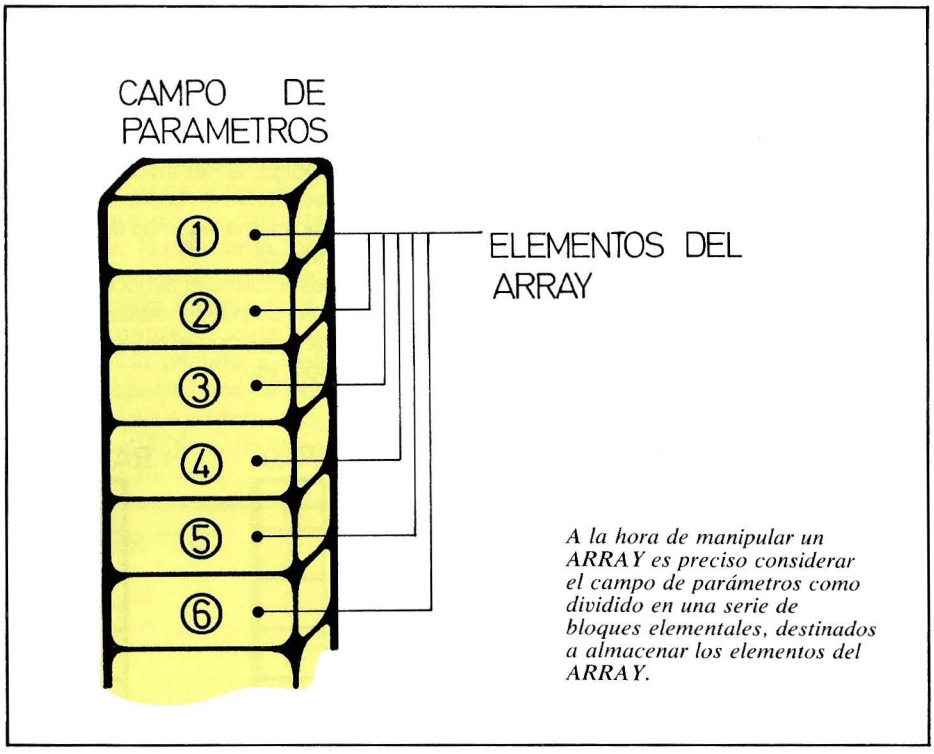
CREATE PALABRA CADENA ASURBANIPAL <CR>

La respuesta del ordenador será la siguiente:

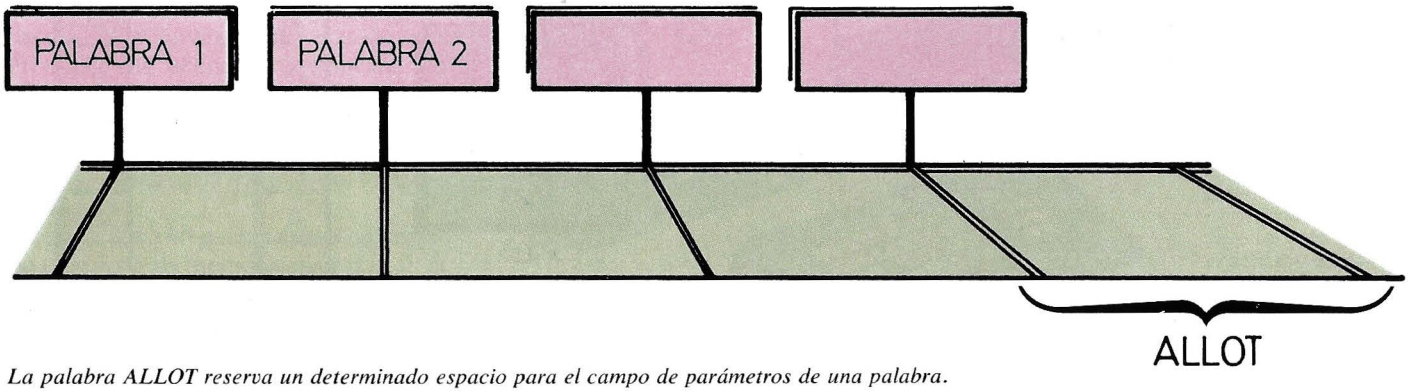
CREATE PALABRA CADENA ASURBANIPAL OK

Para leer esta cadena se puede emplear la siguiente secuencia:

PALABRA 11 TYPE <CR>



A la hora de manipular un ARRAY es preciso considerar el campo de parámetros como dividido en una serie de bloques elementales, destinados a almacenar los elementos del ARRAY.



La palabra ALLOT reserva un determinado espacio para el campo de parámetros de una palabra.

TABLA DE ORDENES FORTH

PALABRA	DESCRIPCION	TIPO
CREATE	Crea una nueva palabra	Manejo del diccionario
,	Transfiere un elemento de la pila al diccionario, reservando para él dos bytes en la zona de parámetros de la palabra	Manejo del diccionario
C,	Transfiere un elemento de la pila al diccionario reservando para él un byte en la zona de parámetros de la palabra	Menejo del diccionario
ALLOT	Reserva un espacio determinado en la zona de parámetros de una palabra	Manejo del diccionario
DEFINER / DOES>	Define un tipo de palabra	Manejo del diccionario

Obteniendo como respuesta:

PALABRA 11 TYPE ASURBANIPAL OK

De la misma forma, también es posible introducir un array de caracteres:

CREATE SEMANA CADENA LUNMARMIEJUEVIESABDOM<CR>

La respuesta será ahora la siguiente:

CREATE SEMANA CADENA LUNMARMIEJUEVIESABDOM OK

Para visualizar un elemento de este array de caracteres, resulta conveniente definir una nueva palabra que facilite el trabajo:

```
: INDICE
1  - 3 * SEMANA +
3  TYPE
;
```

Al ejecutar:

3 INDICE <CR>

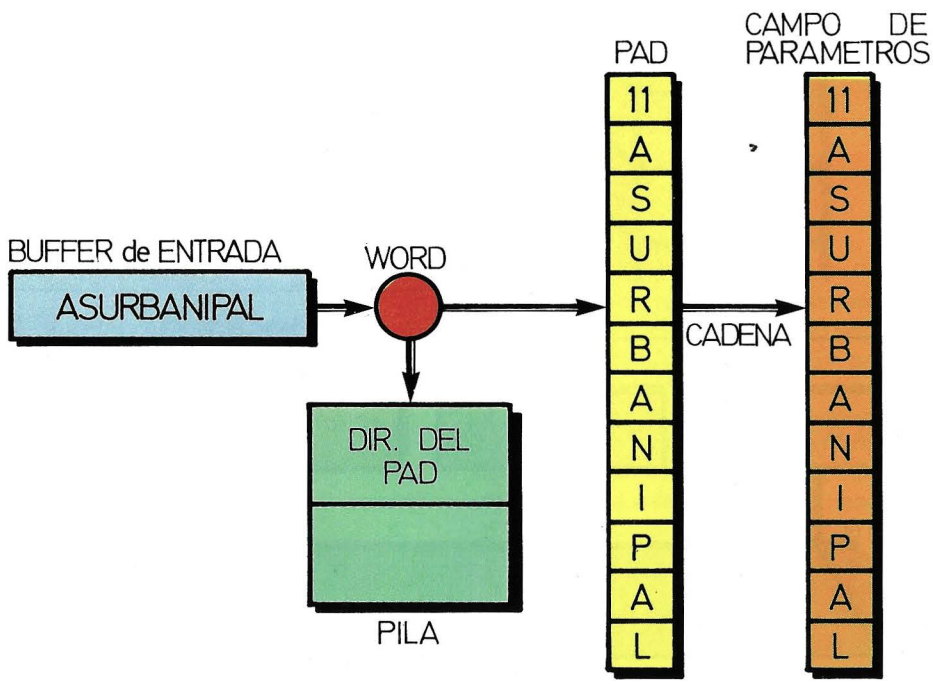
Se obtendría la siguiente respuesta:

3 INDICE MIE OK

Esta forma de crear palabras presentaron ciertos inconvenientes. Al utilizar la palabra tan sólo pasa a la pila la dirección de comienzo de su campo de parámetros, con lo que resulta necesario conocer la longitud de la cadena o la longitud de cada uno de los elementos de la cadena, así como los índices.
Otra palabra que permite definir tipos de palabras es DEFINIR, la cual se emplea además asociada con la palabra DOES>. Mediante esta definición de tipo de palabras, se especifica en primer lugar cómo se creará esa palabra mediante las palabras emplazadas entre la cabecera y la palabra DOES>; e incluso se puede especificar la acción que se ha de realizar cuando una palabra de este tipo sea invocada. La palabra DOES> crea el campo del código.

```
DEFINIR CADENA
 ASCII " WORD DUP 1+ SWAP C    DP C,
 OVER + SWAP
 DO
   | C    C,
 LOOP
 DOES
    DUO 1+ SWAP C
;
```

Por lo que respecta al campo de parámetros, la nueva palabra actúa de la misma forma que la palabra CADENA anteriormente definida; sin embargo, en este caso, por efecto de la definición de un campo de código, al ser invocada la palabra cadena se situará en la pila de la propia longituu de la misma, además de su dirección de comienzo.
A partir de esta forma el almacenamiento de las cadenas se pueden realizar una serie de operaciones con ellas: subdivisión, concatenación, etc.



Mediante la palabra WORD se transfiere el contenido del buffer de entrada al PAD. A su vez, por medio de la palabra CADENA el contenido del PAD pasa al campo de parámetros de una palabra.

Unix (2)

Editores de textos en UNIX

Los usuarios de ordenador que en algún momento desarrollan un nuevo programa o que han de introducir un volumen considerado de información en ficheros de datos, valoran en gran medida la existencia de un editor cómodo y potente.
En este aspecto, el sistema operativo UNIX dispone de un editor que puede ser considerado de los más completos; aunque al ser un editor de línea, el usuario debe enfrentarse al pequeño inconveniente de tener que direccionar cada línea individualmente. No obstante, bajo el ámbito del UNIX se han desarrollado varios editores de pantalla, en los que es posible mover libremente el cursor por todo el campo de la pantalla y realizar directamente la operaciones de edición de textos. Entre estos editores cabe destacar el "VI" desarrollado en la Universidad Californiana de Berkeley.

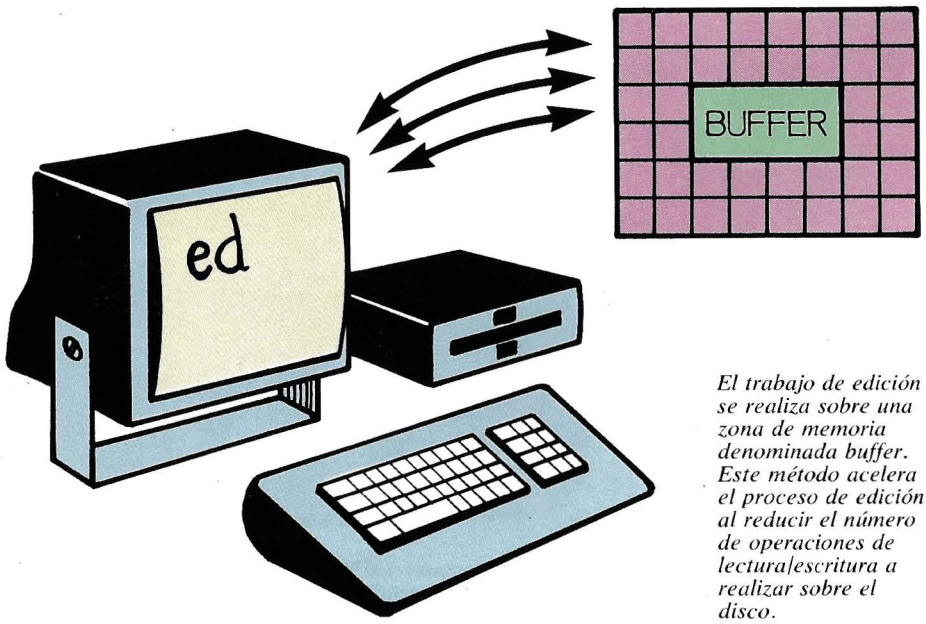
EL EDITOR "ED"

El editor "ed" opera en dos modos: *modo comando* y *modo de inserción* de texto. En el modo comando cualquier entrada de texto se interpreta como un comando de edición, mientras que en el modo de inserción de texto la entrada de texto significa su paso al área de memoria o buffer, sumándose a su contenido.
Cuando se inicia la sesión de edición está activo el modo comando, pasándose al modo de inserción de texto al ejecutarse los comandos de insertar, cambiar y añadir texto. Para volver al modo comando basta con teclear un punto (.) seguido por un retorno de carro.
Los comandos del "ed" tienen una estructura simple y única: de cero a dos números de líneas seguidos por una letra que hace referencia al comando y, opcionalmente, los parámetros que concretan dicho comando; por ejemplo:

1,3 p

En la descripción de los números de línea están permitidos varios caracteres especiales como: el punto (.), utilizado para indicar la línea en curso, y el dólar ($), que hace referencia a la última línea del fichero.
La llamada al editor "ed" desde el caparazón o shell del UNIX se realiza con el siguiente formato:
$ ed nombre-fichero ($ es la petición o prompt del sistema, e indica que estamos en el shell).
La reacción del editor varía según exista o no el fichero especificado, respondiendo con el número de caracteres que contiene el fichero en el primer caso, o bien con el signo de interrogación seguido del nombre del fichero en el segundo. A partir de este instante el editor se encuentra en modo comando.
A continuación se relacionan los diversos comandos de edición, agrupados por funciones, expresándose a continuación de la

Como es fácil suponer, el editor no actúa directamente sobre el fichero contenido en el disco; ello haría al editor más lento debido a los continuos accesos al disco. En la práctica, el contenido del fichero es transferido del disco a una zona de memoria, denominada "buffer", utilizada como área de trabajo durante la sesión de edición para crear o modificar el texto del fichero.
Ha de tenerse en cuenta que el contenido de esta zona de memoria es temporal: su contenido se perderá al terminar la sesión de edición si, previamente, no se ha ordenado que el contenido del buffer se escriba en un fichero.



El trabajo de edición se realiza sobre una zona de memoria denominada buffer. Este método acelera el proceso de edición al reducir el número de operaciones de lectura/escritura a realizar sobre el disco.

letra de comando el resto de la palabra entre paréntesis:

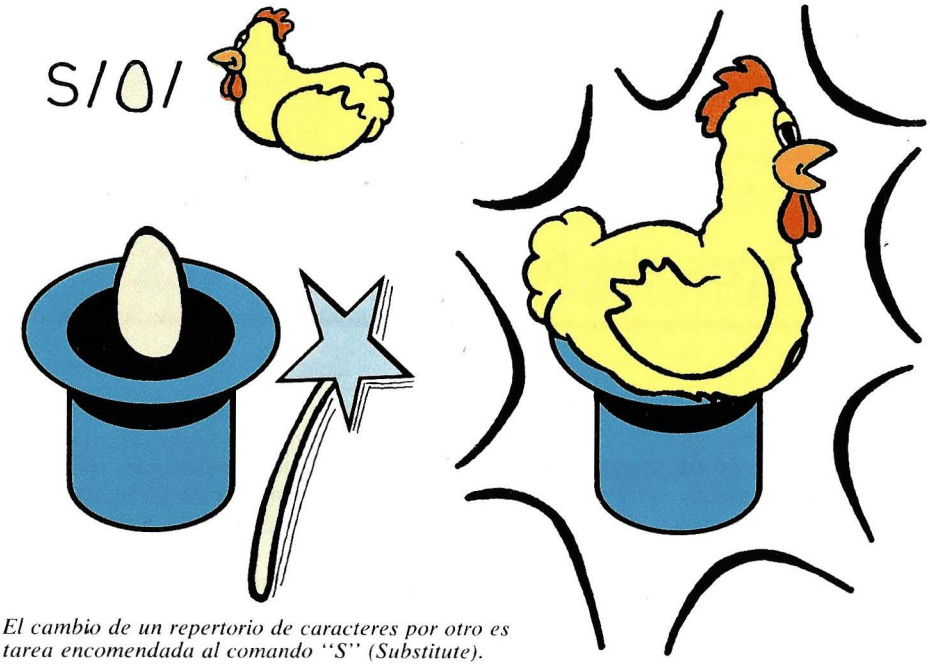
- *Comandos de lectura-escritura de ficheros*

La incorporación de ficheros externos al que se está editando, o la escritura del contenido total o parcial del buffer en otros ficheros, se realiza con los siguientes comandos:

e(dit) nombre-fichero: lee un fichero y lo introduce en el buffer, borrando el contenido previo.

n1,r(ead) nombre-fichero: lee un fichero y lo inserta en el buffer después de la línea n1 sin borrar el contenido.

n1,n2,w (rite) nombre fichero: escribe en el fichero especificando el contenido del buffer desde la línea n1 a n2, o el buffer completo si no se indican estos números de línea.



El cambio de un repertorio de caracteres por otro es tarea encomendada al comando "S" (Substitute).

- *Comando de posicionamiento*

Siempre existe una línea en curso o línea activa de las múltiples contenidas en el buffer sobre la cual actúan los diversos comandos. A veces puede ser neceario cambiar esta línea; para ello existen los siguientes comandos:

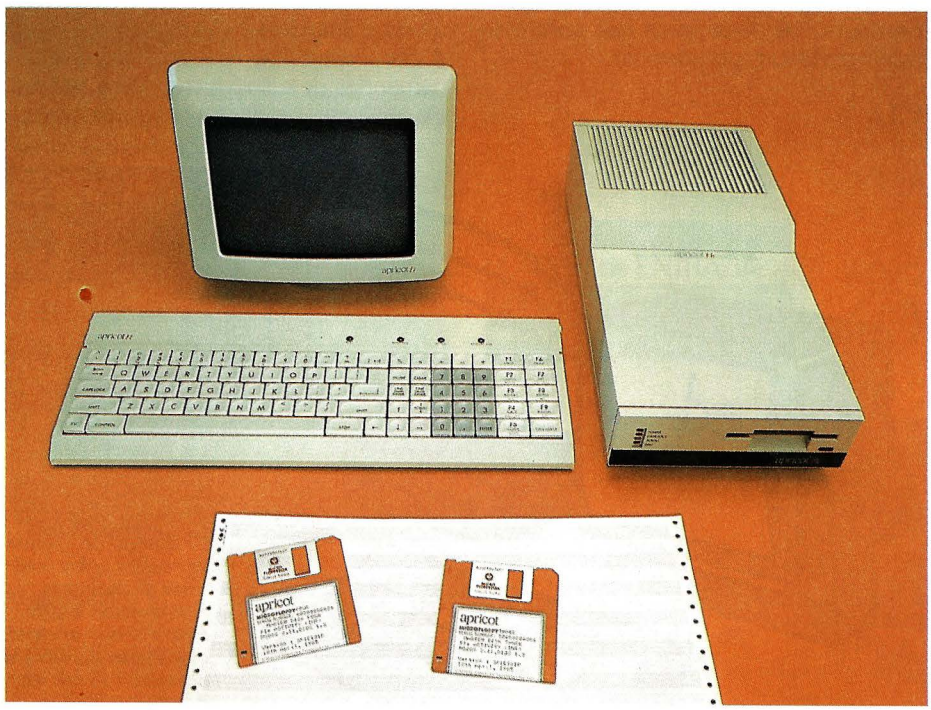
no.línea: hace que la línea en curso sea la indicada.

+n: la nueva línea en curso es la actual más el número n de líneas indicado.

– n: la nueva línea en curso es la actual menos el número n.

.=: muestra el número de la línea en curso.

$=:indica el número total de líneas del buffer.



La tendencia actual es facilitar al máximo la introducción de informaciones en el ordenador; tal es el cometido de los editores, procesadores de texto, periféricos para el reconocimiento de voz...

- *Comando de listado*

La visualización del contenido del buffer se realiza con los comandos:

n1,n2,p(rint): muestra las líneas comprendidas entre n1 y n2.

1 (ist): lista el contenido del buffer incluyendo los caracteres ASCII y los caracteres de control existentes.

- *Comandos de inserción de texto*

La introducción de nuevo texto en el buffer es posible gracias a los comandos:

n1 a (ppend): añade líneas de texto a partir de la línea n1 indicada.

n1 i (nsert): inserta líneas de texto antes de la línea n1 indicada.

Ambos comandos activan el modo de inserción de texto, siendo necesario la introducción de un punto (.) y retorno de carro (<CR>) para desactivar dicho modo y volver al modo comando.

● *Comandos de borrado de texto*

Las líneas de texto no deseadas se eliminan del buffer con el comando:

n1,n2, d (elete): borra las líneas de texto desde n1 a n2 inclusive.

● *Comandos de borrado e inserción de texto*

Dentro de "ed" cabe la posibilidad de borrar las líneas que desee el usuario y luego insertar el texto correspondiente, o bien utilizar el comando "change" que realiza estas dos funciones a la vez:

n1,n2,c (hange): borra las líneas indicadas e introduce el modo de inserción de texto.

● *Comandos de localización de caracteres*

Para encontrar una sucesión de caracteres determinados en el texto se dispone de los siguientes comandos:

/texto/: localiza la primera ocurrencia de texto en el buffer buscando a partir de la línea en curso hasta el final.

//: localiza la siguiente ocurrencia de texto buscando hacia el final del buffer.

?texto?: localiza la primera ocurrencia de texto en el buffer, buscando a partir de la línea en curso hacia el principio.

??: localiza la siguiente ocurrencia de texto buscando hacia el principio del buffer.

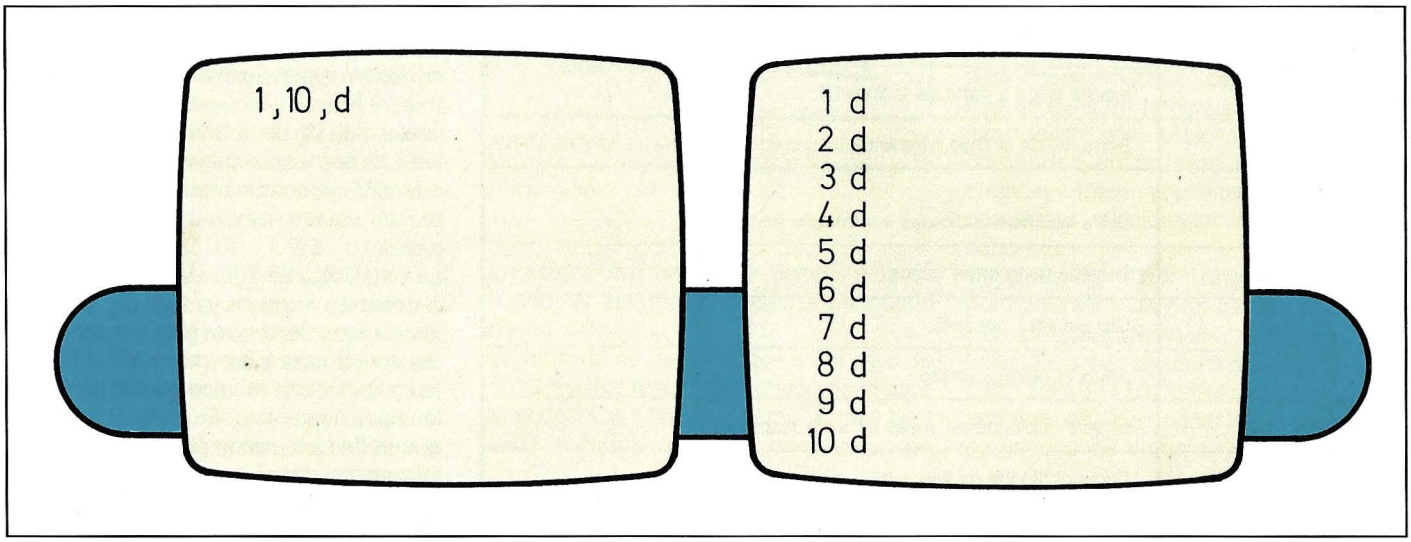
● *Comandos de sustitución de caracteres*

El cambio de un conjunto de caracteres por otro se lleva a cabo por medio del comando "substitute".

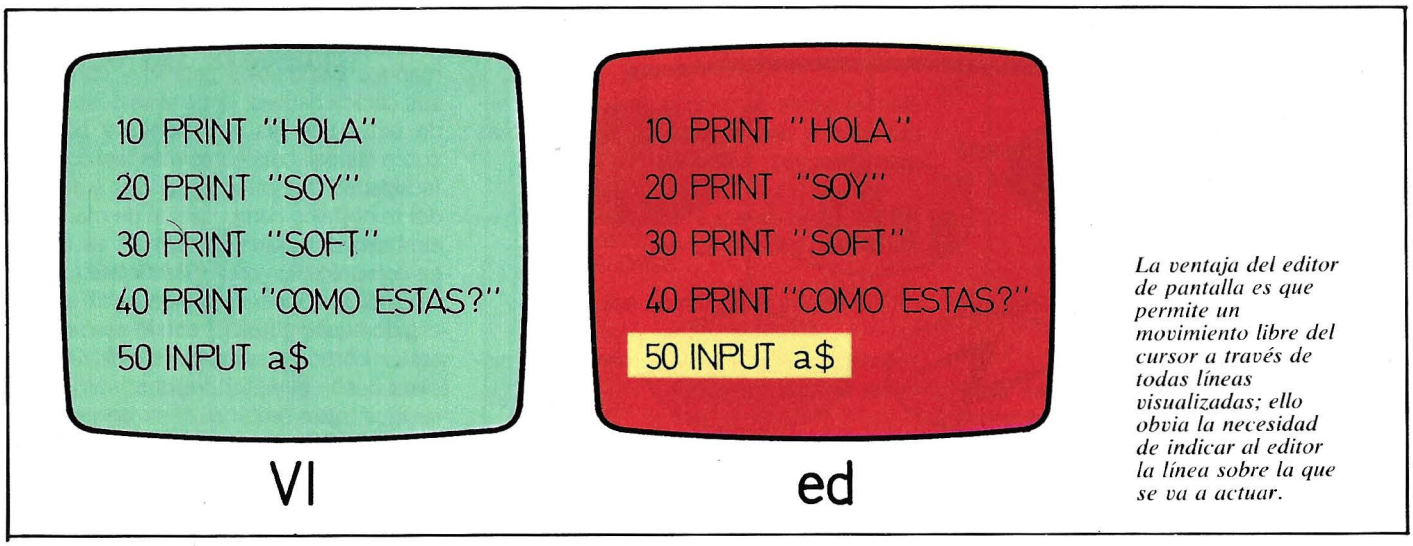
S(ubstituye)/texto-antiguo/texto-nuevo/: cambia el texto antiguo por el nuevo.

Para posibilitar la inserción de caracteres al principio y al final de la línea, existen dos carateres especiales que pueden sustituir dentro del comando al texto antiguo:

^= principio de línea.
$=final de línea.



La disponibilidad de comandos que actúen sobre bloques de líneas simplifica en extremo la corrección de textos.



La ventaja del editor de pantalla es que permite un movimiento libre del cursor a través de todas líneas visualizadas; ello obvia la necesidad de indicar al editor la línea sobre la que se va a actuar.

COMANDOS DEL EDITOR DE LINEAS "ED"

COMANDO	ACCION
e nombre	Lee el fichero nombre y lo aloja en el buffer
n1, r nombre	Lee el fichero nombre y lo inserta en el buffer a partir de la línea n1
n1, n2, w nombre	Escribe en el disco el fichero nombre desde n1 a n2
± n	Se posiciona ± n líneas a partir de la línea actual
. =	Muestra número de línea en curso
= $	Muestra número toltal de líneas
n1, n2, p	Imprime desde la línea n1 a la n2.
l	Lista el contenido del buffer.
n1 a	Añade texto a partir de la línea n1
n1 i	Inserta texto a partir de la línea n1.
n1, n2, d	Borra desde la línea n1 a la n2.
n1, n2, c	Borra las líneas indicadas e introduce texto.
/texto/	Localiza texto entre barras.
s/text1/tex2/	Cambia tex1 por tex2.
u	Anula el último cambio
n1, n2, m n3	Mueve el bloque de líneas n1 a n2 después de n3.
n1, n2, t n3	Copia el bloque de líneas n1 a n2 después de n3.
g	Extiende el valor del comando a todo el texto.
q	Termina la sesión de edición.

si se desea anular el cambio efectuado se puede utilizar el comando "undo":

u(ndo): anula la última sustitución si se está posicionando en la línea afectada por el cambio.

- *Comandos de bloque*

La manipulación de bloques de líneas se consigue con los comandos:

n1,n2,m(ove)n3: mueve las líneas comprendidas entre n1 y n2 situándolas tras la línea n3.

n1,n2,t(ransfer)n3: copia las líneas compredidas entre n1 y n2 detrás de la línea n3.

- *Comando varios*

Restan varios comandos de edición no encuadrables en ninguna de las anteriores clasificaciones; estos son:

g(lobal): hace que la acción del comando se extienda a todo el fichero.

q(uit): finaliza la sesión de edición.

!comando: ejecuta el comando del sistema sin salir del editor.

- *Caracteres especiales*

Existen dos caracteres que sirven para borrar caracteres o líneas; aunque visibles en la pantalla, estos caracteres no se incorporan al buffer:

#: borra un carácter
: borra la línea en curso.

EL EDITOR VI

El editor VI es un editor de pantalla; como tal es en algunos aspectos más potente y práctico que su compañero, el editor de línea "ed.". No obstante, cabe señalar que el editor VI presenta algunos inconvenientes, tales como la imposibilidad de hacer correcciones de texto automáticamente en todo el fichero y mover o copiar bloques de líneas. Joy W. de la Universidad de California resolvió estas pegas al lograr que el editor VI se comunicase con el "ed" para realizar estas tareas hasta entonces imposibles.

La instalación del editor VI depende del tipo de pantalla a utilizar ya que por desgracia los códigos de control para el movimiento del cursor, inserción y borrado de caracteres y líneas, etc., no son iguales para todos los tipos de pantalla. En consecuencia, hay que definir las características de cada pantalla antes de poder utilizar el editor; no obstante esto no es necesario en la mayoría de los casos al disponerse de las definiciones de las pantallas más comunes en el mercado. Esta operación sólo es necesaria cuando se instale el editor.

Una vez instalado el editor ya se está en disposición de ejecutar el abanico de comandos disponibles para VI y comprobar sus características; entre ellas destacan las de actuar en las inserciones y borrados sobre las palabras completas, así como las facilidades que ofrece para fijar el formato del texto tales como fijación de márgenes, centrado de palabras, etc.

La comunicación con el editor "ed" se realiza precediendo a cada comando "ed" con el signo dos puntos (:), con lo que se podrá actuar como si realmente la edición se hiciera desde el editor de línea.

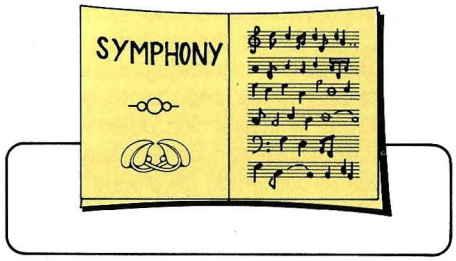
Al igual que "ed", VI tiene acceso a los comandos shell del sistema sin necesidad de terminar la sesión de edición, anteponiendo al comando a ejecutar los signos dos puntos y exclamación (:!).

Symphony (1)

Cinco entornos en un paquete integrado

A finales de 1982, Lotus Development Corporation introdujo su paquete de software integrado LOTUS "1,2,3". Este programa cambió las reglas de competición en el mercado informático, combinando hojas electrónicas, bases de datos y gráficos de gestión en el mismo paquete. Al principio, parecía que LOTUS "1,2,3" era un producto difícil de superar; no obstante, con el paso del tiempo, han surgido varios programas integrados que le han superado notoriamente.

El sucesor de LOTUS "1,2,3" ha sido el paquete integrado SYMPHONY, promovido por la misma compañía propietaria del anterior. Entre otras novedades, SYMPHONY incorpora dos entornos adicionales: tratamiento de textos y comunicaciones. En el presente capítulo da comienzo un estudio sobre el SYMPHONY, en el que se presta una especial atención a su manejo. Dado que las características de sus distintos entornos son similares a las ya descritas para otras aplicaciones, se hará especial énfasis en los comandos de servicio.



SYMPHONY es un paquete integrado creado por la firma americana Lotus Development Corporation.

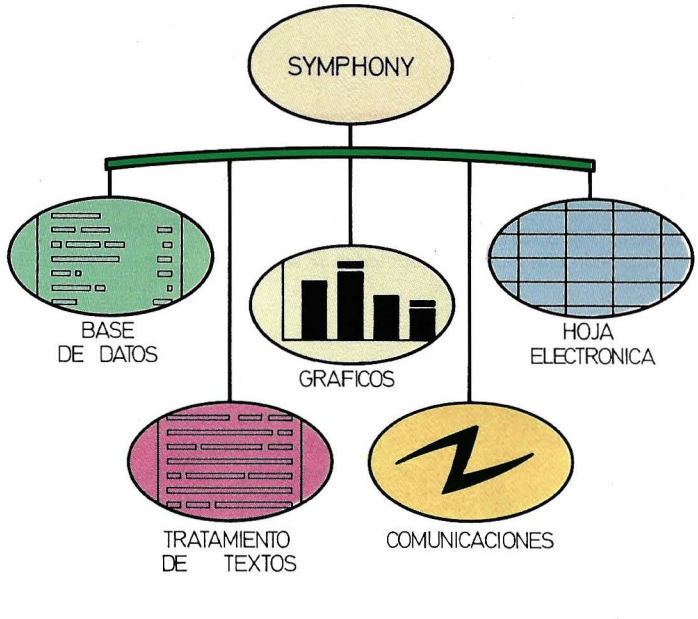
MANEJO DEL SYMPHONY

Como se ha mencionado anteriormente, el paquete SYMPHONY ofrece cinco entornos distintos: hoja electrónica (SHEET), proceso de textos (DOC), gráficos de gestión (GRAPH), base de datos (FORM) y comunicaciones (COMM). El usuario puede pasar de un entorno a otro y "navegar" dentro de cada uno de ellos a través de un sistema de menús. La organización de este sistema es de tipo jerárquico y, por lo tanto, en algunos casos, después de seleccionar una de las opciones de un menú, el programa presenta un nuevo menú.

Es importante destacar que en el caso de SYMPHONY el concepto menú no implica la ocupación total de la pantalla; más bien todo lo contrario: las opciones aportadas por el menú se concentran en dos o tres líneas de la pantalla, quedando el resto a disposición del usuario.

Dentro de los menús aportados por el programa cabe distinguir dos tipos básicos. En primer lugar existe un sistema de menús para comandos de servicio, es decir, para desencadenar la ejecución de órdenes de carácter general. Y en el segundo lugar existe otro sistema de menús para comandos de entorno que, en consecuencia, ofrecen la posibilidad de ejecutar comandos relacionados con un entorno concreto.

Después de haber "arrancado" el programa SYMPHONY, el usuario puede decidir en que sistema de menús desea trabajar. Para ello se limitará a pulsar las teclas F9 y F10 en orden a entrar en el sistema de comandos de servicio o de comandos de entorno, respectivamente.



El paquete SYMPHONY incluye cinco entornos de trabajo: hoja electrónica (SHEET), proceso de textos (DOC), gráficos de gestión (GRAPH), base de datos (FORM) y comunicaciones (COMM).

Una vez pulsada la tecla, aparecerá en la pantalla el menú principal del sistema elegido. A lo largo de la sesión de trabajo se podrá cambiar de sistema de menús sin más que pulsar la tecla correspondiente (F9 o F10).

SISTEMA DE MENUS PARA COMANDOS DE SERVICIO

SYMPHONY dispone de ocho servicios distintos y, por lo tanto, al pulsar la tecla F9 aparecerá en la pantalla un menú en el que se ofrecerán dichos servicios. El cursor aparecerá sobre el primero de ellos y las teclas de desplazamiento hacia la izquierda o derecha (← y →) permitirán al usuario elegir el servicio deseado, ejecutándose éste sin más que presionar la tecla de retorno ← . A continuación se describen las características generales de cada uno de los comandos de servicio:

- **VENTANA** (WINDOW)

Mediante esta opción el usuario podra definir distintas pantallas lógicas que se visualizarán simultáneamente sobre la pantalla física. Cada una de las pantallas definidas podrá tener distinto tamaño e incluso ser de distinto tipo. Así, por ejemplo, en una podríamos estar en el entorno hoja electrónica y en otra en sistema gráfico.

Algunas pantallas pueden ofrecer una información completa sobre un determinado elemento; por ejemplo, una ventana que contenga todos los datos relativos a un registro de la base de datos. En cambio, otras pantallas tan sólo incorporan una información parcial. Un ejemplo de este segundo tipo puede ser una ventana sobre la hoja electrónica en la que sólo se observarán algunas filas y algunas columnas; en este caso el usuario podrá desplazar la ventana en los cuatro sentidos: arriba, abajo, a la izquierda o a la derecha.

Una característica importante de SYMPHONY es la interacción dinámica entre las distintas pantallas lógicas: si el usuario modifica el contenido de alguna de ellas, el efecto puede observarse no sólo en la modificada, sino también en todas las relacionadas con la pantalla alterada.

Dentro del servicio WINDOW pueden distinguirse las siguientes opciones:

1. UTILIZACION (USE).
Permite decidir cuál de las ventanas lógicas se desea utilizar.

2. CREACION (CREATE)
Se puede utilizar para la creación de nuevas ventanas, identificando todas las características asociadas a la ventana lógica creada.

3. BORRADO (DELETE)
Sirve para borrar alguna de las ventanas lógicas definidas previamente.

4. MODIFICACION (LAYOUT)
Mediante esta opción el usuario puede alterar las características de una ventana.

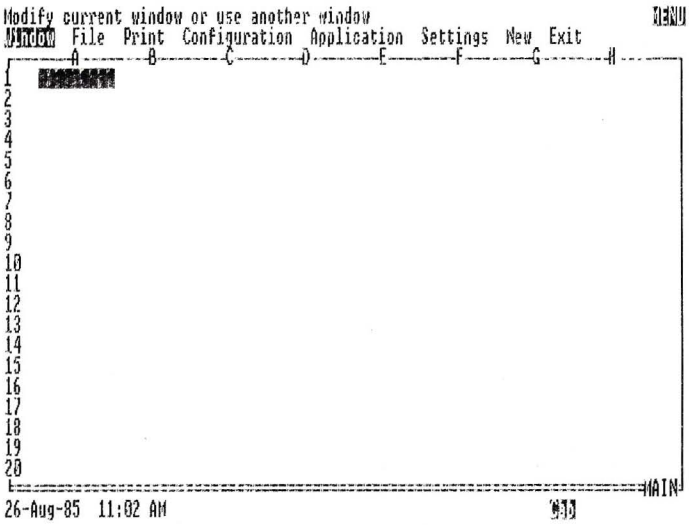
5. OCULTACION (HIDE)
Permite ocultar una pantalla. Es importante no confundir este comando con DELETE, ya visto anteriormente; ambos provocan la desaparición de una ventana, pero HIDE deja abierta la posibilidad de volverla a ver, mientras que DELETE elimina toda posible referencia a la ventana borrada.

6. SEPARACION (ISOLATE)
Se encarga de eliminar todas las pantallas definidas, dejando sólo activa la ventana principal.

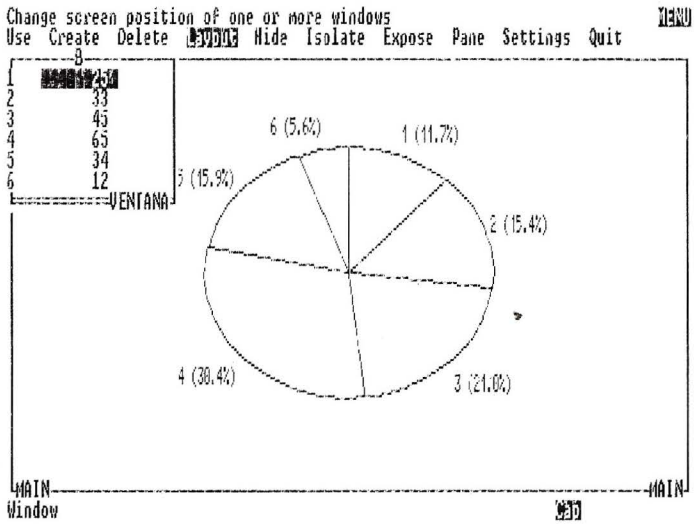
7. EXPOSICION (EXPOSE)
Sirve para que una ventana ocultada mediante el comando HIDE vuelva a visualizarse.

8. PARTICIPACION (PANE)
Se puede utilizar para dividir una ventana en "mitades" o "cuartos".

9. INSTALACION (SETTINGS)
Al ser invocado este comando aparecerá en la pantalla un nuevo menú, ofreciendo



Sin más que pulsar la tecla F9 SYMPHONY presentará en la franja superior de la pantalla una lista con todos los comandos de servicio disponibles.



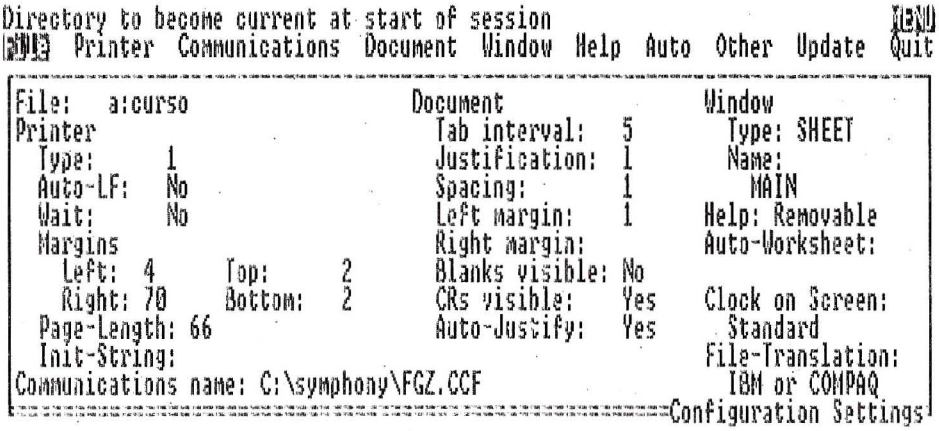
Mediante la definición de ventanas se hace posible utilizar distintas pantallas lógicas. En una de las ventanas de la pantalla reproducida aparecen datos, mientras que en la otra se observa el diagrama asociado a dichos datos.

al usuario distintas posibilidades para configurar la ventana activa.

10. SALIDA (QUIT)
Mediante este comando de servicio se abandona la opción WINDOW.

- **FICHEROS** (FILE)

Los comandos agrupados bajo la denominación FILE tienen como misión transferir información entre la hoja electrónica, cargada en memoria principal, y los soportes externos de almacenamiento. En general, la mayoría de las sesiones de trabajo con SYMPHONY comenzarán con la recuperación de una hoja electrónica cargada en disco. A continuación, el usuario realizará modificaciones sobre ella llegando, en algunos casos, a producir informes, y finalizará almacenando en disco el contenido modificado de la hoja electrónica.
La forma de acceder a este comando de servicio consiste en seleccionar la opción FILE del menú principal y, seguidamente, pulsar la tecla RETURN. De esta manera



El comando CONFIGURATION se emplea para definir las características generales en que se utilizará la aplicación SYMPHONY.

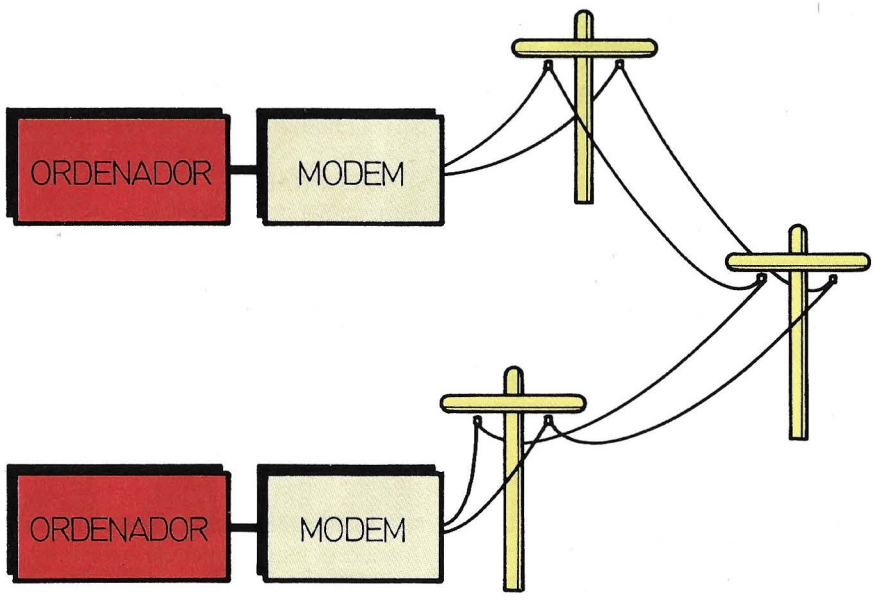
se conseguirá que aparezcan en la pantalla los siguientes comandos, entre los que el usuario podrá elegir:

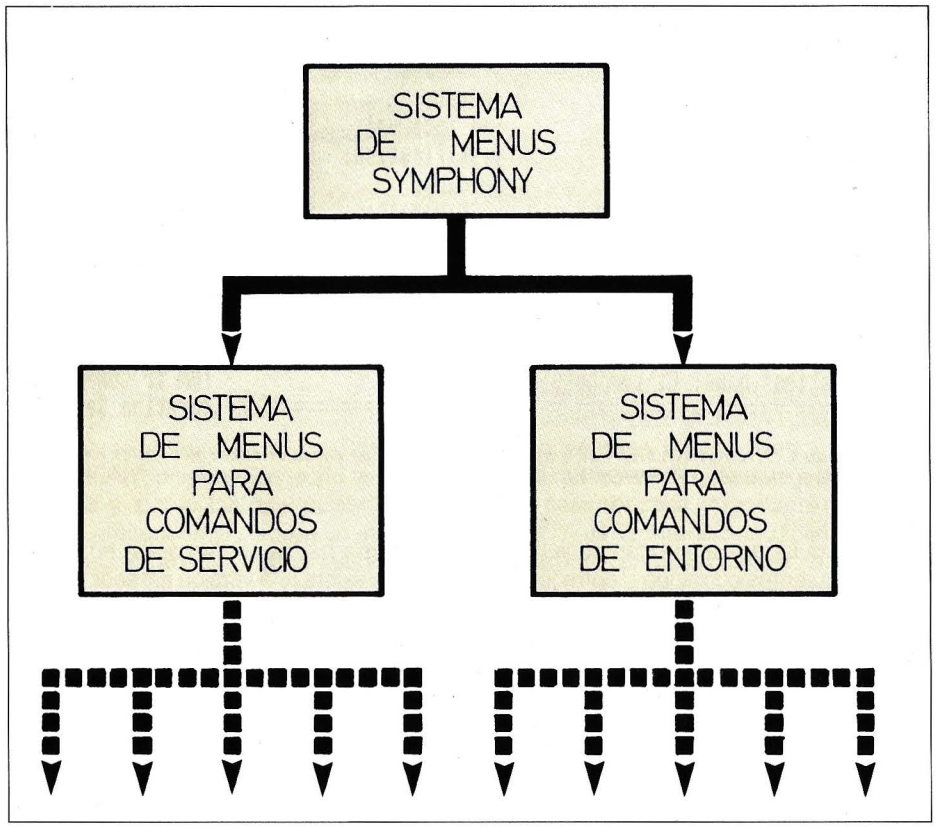
1. GUARDAR (SAVE)
Permite pasar la hoja electrónica desde la memoria principal a un fichero en disco. El programa pedirá confirmación en el caso de que exista una versión anterior almacenada en el disco.

2. RECUPERACION (RETRIEVE)
La misión de este comando de servicio es complementaria respecto a la del anterior:

Líneas telefónicas y ordenadores

Con objeto de aprovechar la infraestructura de líneas telefónicas ya existentes, se han utilizado éstas como elemento de comunicación entre ordenadores. El problema principal para realizar tal operación reside en la diferencia entre las señales manejadas por un ordenador y las que tradicionalmente se utilizan en las comunicaciones telefónicas. Las primeras son digitales, es decir la gama de frecuencias es discreta; mientras que las segundas son analógicas y, por lo tanto, con una gama de frecuencias continua. La solución de este problema consiste en acoplar un modem encargado de realizar la conversión entre ambos tipos de señales.
Otro de los problemas inherentes al uso de las líneas telefónicas como canales de comunicación entre ordenadores, consiste en el ruido producido por los circuitos electrónicos: los átomos que componen los materiales vibran y difunden ondas electromagnéticas de distinta frecuencia. Evidentemente, este ruido se mezcla con la señal enviada y, en algunos casos, puede producir tal distorsión que se pierda completamente el mensaje original. Para evitar este problema resulta necesario colocar repetidores cada cierta distancia en la línea telefónica; de esta forma, cuando la señal llega deteriorada a un repetidor, éste es capaz de recomponerla y volver a lanzarla con la misma calidad original. Para determinar la distancia a la que deben situarse los repetidores basta con hacer una única consideración: debe ser tal que la distorsión producida entre dos de ellos sea lo suficientemente pequeña como para poder regenerar la señal. Salvados estos dos problemas, las líneas telefónicas constituyen un camino aceptable para la comunicación entre ordenadores; de hecho, siguiendo esta técnica se han conectado equipos situados a miles de kilómetros.





SYMPHONY aporta un sistema de menús para facilitar su explotación. Dentro de él se pueden distinguir dos subsistemas: uno para comandos de servicio y otro para comandos de entorno.

permite pasar la información contenida en un fichero a la hoja electrónica situada en la memoria principal.

3. COMBINACION (COMBINE)
El comando COMBINE también se encarga de pasar información del disco a la memoria principal: pero en vez de cargar completamente un fichero, permite "traer" parte de él y combinarlo con la información previamente existente en la hoja electrónica.

4. EXTRACCION (XTRACT)
Su misión consiste en copiar información de parte de la hoja electrónica situada en la memoria principal (no confundir con SAVE que producía una copia completa).

5. BORRADO (ERASE)
Si el usuario selecciona el comando de servicio ERASE, el programa SYMPHONY solicitará a continuación el tipo y nombre de un fichero que será borrado inmediatamente.

6. ESPACIO LIBRE (BYTES)
Se limita a mostrar en pantalla el espacio disponible en la unidad de almacenamiento activa.

7. LISTADO (LIST)
Sirve para producir listados de ficheros; antes de su ejecución el usuario debe indicar el nombre y tipo del fichero a listar.

8. TABLA (TABLE)
Este comando es similar al anterior; la única diferencia estriba en que el listado se producirá en forma de tabla sobre la hoja electrónica activa.

9. ADQUISICION (IMPORT)
Mediante el comando de servicio IMPORT se pueden adquirir datos, bien sea numéricos o textos, desde ficheros ASCII o LICS.

10. DIRECTORIO (DIRECTORY)
Cuando el usuario ejecuta este comando de servicio, en la pantalla aparecerá la identificación del disco activo y una lista con todos los ficheros que en él estén almacenados.

- **IMPRESION** (PRINT)

SYMPHONY permite obtener informes escritos en todos sus entornos. El comando de servicio PRINT sirve precisamente para gestionar dicha impresión. Para ello, al ser seleccionado, ofrece cinco opciones distintas:

1. IMPRIMIR (GO)
Se utiliza para producir la impresión de un documento, bien sea por la impresora o utilizando un fichero de impresión.

2. AVANCE DE LINEA (LINE ADVANCE)
Su única misión consiste en avanzar una línea en la impresora, situando como próxima posición de escritura el primer carácter de la siguiente línea a la actual.

3. AVANCE DE PAGINA (PAGE ADVANCE)
Análogo al comando anterior, pero avanzando una página en vez de una línea.

4. ALINEAMIENTO (ALIGN)
En la ejecución de algunos comandos, el programa SYMPHONY necesita conocer el número de página y el número de línea en que debe escribir; al efecto, el comando ALIGN permite la inicialización de una variable asociada a dichos parámetros (página y línea).

5. OPCIONES (SETTINGS)
Este último comando de servicio de PRINT utiliza una subestructura de menús muy ramificada; en ella permite que el usuario especifique características generales de impresión como: número de líneas por página, tipo de numeración, espaciado, cabeceras, etc.

- **CONFIGURACION** (CONFIGURATION)

Este comando de servicio tiene como misión realizar una configuración de SYMPHONY acorde con el ordenador en que se vaya a explotar. Para ello ofrece una estructura con distintas opciones, mediante la cual se van asignando valores a los parámetros que determinan la configuración. Dado que este comando suele utilizarse sólo para inicializar el sistema, no entraremos en definir con más precisión su funcionamiento.

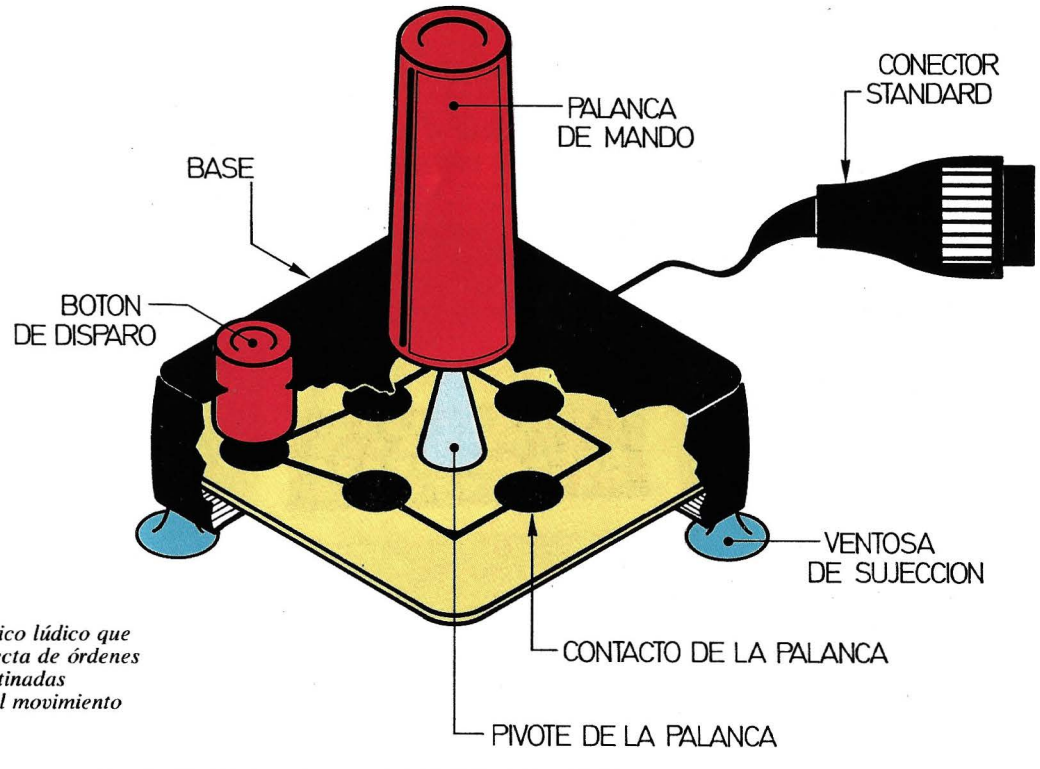
Control de periféricos

Joysticks y paddles en acción

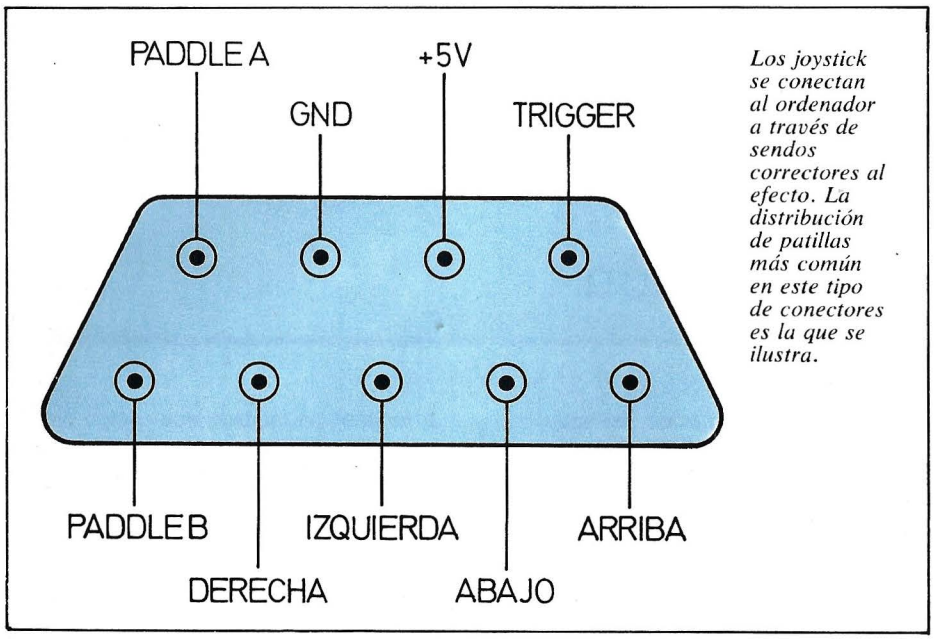
Para lograr una mayor verosimilitud y realismo en la ejecución de programas de acción, el ordenador necesita recibir informes y sensaciones externas que le indiquen qué es lo que está ocurriendo a su alrededor. Estas influencias del medio exterior serán captadas por el ordenador en forma de señales eléctricas, a través de dispositivos adaptadores denominados "interfaces". En resumidas cuentas, los dispositivos de "interface" no son más que fieles traductores de esas señales eléctricas, procedentes de algún periférico, a un formato comprensible por el ordenador.

Para comprender mejor el proceso de captación de información del exterior, se puede pensar en la secuencia que se seguirá para conseguir que una determinada orden llegue a ser "comprendida" por la máquina. En primer lugar, el usuario decidirá qué orden desea transferir a la máquina. Acto seguido emprenderá las acciones necesarias, actuando sobre algún dispositivo periférico externo al ordenador que transforme sus pensamientos en señales eléctricas. A continuación, el periférico pasará esta información a un "interface" que se encargará de convertir las señales eléctricas en señales digitales codificadas en binario (unos y ceros), que son las únicas que puede comprender directamente el ordenador. Una vez que el ordenador ha recibido ese código binario, sólo le queda interpretarlo en la forma adecuada, para poder ejecutar la orden que quería transmitirle el usuario. Pero, ¿quién se encarga de interpretar los códigos binarios recibidos? Como es fácil suponer, será el propio ordenador instruido por el programa que se esté ejecutando en ese momento; programa que, normalmente, estará codificado en BASIC.

Así pues, es obvio que el BASIC dispondrá de diversas funciones y comandos para poder controlar los distintos periféricos que se pueden acoplar a un ordenador; el estudio de estas funciones y comandos constituye, precisamente, el objetivo de este capítulo.



El "joystick" es un periférico lúdico que permite la transmisión directa de órdenes a la máquina; órdenes destinadas normalmente a controlar el movimiento de figuras por la pantalla.



Los joystick se conectan al ordenador a través de sendos correctores al efecto. La distribución de patillas más común en este tipo de conectores es la que se ilustra.

LA PALANCA DE MANDO O "JOYSTICK"

Entre la gran gama de periféricos de "acción" que se pueden acoplar a un ordenador, el más popular es sin duda alguna la palanca de mando o "joystick". Este consta de una base sobre la que se eleva una palanca móvil, similar a la palanca de mando de los aviones de combate. Su utilidad a la hora de controlar la acción asociada a los programas de juegos es indudable... Pero, ¿cómo funciona un joystick?

Su cometido más frecuente es indicar la dirección y sentido hacia el que se quiere desplazar por la pantalla una nave espacial o cualquier otro artefacto similar. Mediante el movimiento de la palanca en cualquiera de las cuatro direcciones cardinales (norte, sur, este y oeste), o sus direcciones intermedias (noreste, noroeste, sureste y suroeste), se consiguen definir ocho posibles orientaciones para el movimiento.

Para detectar la dirección en que ha sido movida la palanca, se dispone de cuatro interruptores situados en la base, uno para cada "punto cardinal". Así, al mover la palanca en uno de estos sentidos se cerrará el correspondiente interruptor, generando una señal eléctrica que es codificada por el "interface"; este circuito adaptador proporciona así un código binario diferente para cada sentido de desplazamiento. El código binario puede ser "leído" desde el BASIC con la ayuda de la función STICK ( ), que retornará un valor diferente para cada una de las ocho posibles direcciones. Como sucede con todas las funciones BASIC, dicho valor debe ser asignado a alguna variable previamente definida para que esté disponible para su uso posterior.

El formato más general de una instrucción encargada de "leer" el estado en cada momento de la palanca de mando, es el que sigue:

<Núm. de línea> X=STICK (<exp.>)

El formato incluye a la variable X (que puede ser otra cualquiera, con otro nombre, siempre que se trate de una variable numérica), en la que se almacena el valor proporcionado por la función STICK; dicha variable será utilizada en el resto del programa.

La expresión <exp.> tiene la misión de diferenciar el "joystick" cuyo estado se quiere examinar, puesto que generalmente será posible conectar con simultaneidad dos palancas de mando en sendos conectores de entrada. Bien es cierto que en algunos casos será posible conectar hasta cuatro palancas de mando o más. En definitiva, el "joystick" conectado a la toma 1 se leerá normalmente con: X=STICK(1); mientras que la palanca conectada al acceso 2 se leerá con: X=STICK(2); y así sucesivamente. La expresión utilizada puede ser cualquier constante, variable o expresión matemática, que se evaluará al ejecutar la instrucción para obtener así un número entero, como se observa en el siguiente ejemplo:

```
X=STICK(Y-2)
```

En él, si el valor actual de Y es 4, se leerá el joystick conectado al acceso o toma 2. Ello significa que se puede transferir el control a otra palanca con sólo variar el valor de Y. Generalmente, el rango de posibles valores del parámetro asociado a la función STICK está acotado entre ciertos límites, generando un mensaje de error (de llamada ilegal a una función) si se in-

El joystick o palanca de control es el periférico lúdico por excelencia; su presencia es muy frecuente junto a los ordenadores domésticos.

tenta una llamada con un valor fuera de rango. Por desgracia, el parámetro <exp.> que se debe aportar a la función STICK ( ), así como los valores entregados por dicha función, varían mucho de unos intérpretes BASIC a otros, por lo que será imprescindible consultar el manual de cada ordenador.

En algunos casos (en los equipos MSX, por ejemplo), es posible leer con la función STICK las teclas de movimiento del cursor, simulando así una palanca de mando que permitirá desplazar objetos por la pantalla. Como es fácil imaginar, se asignará a esta función particular un número de "port" o acceso (<exp.>) exclusivo para este cometido; por ejemplo: X=STICK(0).

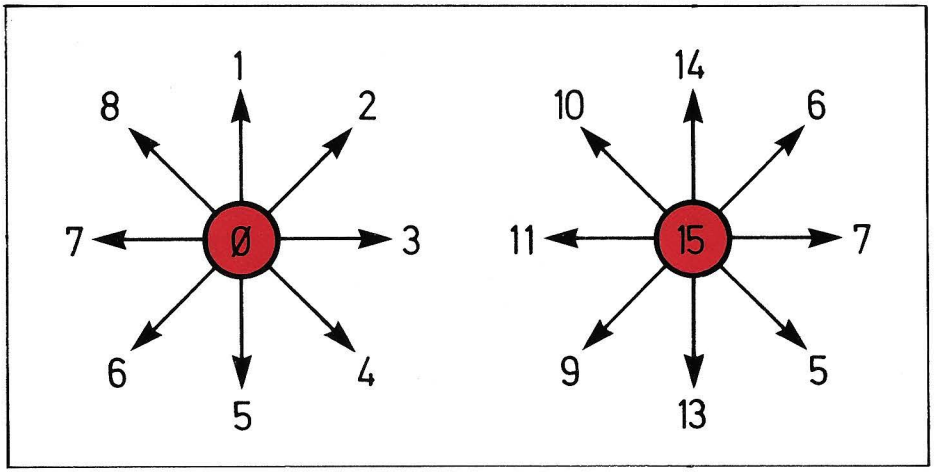
¿Cómo se pueden diferenciar las diversas acciones a tomar según el código entregado por la función STICK ( ) en el contexto de un programa BASIC? Normalmente, esto se hace por medio de instrucciones de decisión del tipo IF/THEN/ELSE: según se cumpla o no una determinada condición, se podrá emprender consecuentemente una acción u otra.

Para poder dar los siguientes pasos habrá que conocer los códigos que genera la función STICK, para lo que será imprescindible consultar el manual de cada dialecto BASIC o bien acudir al siguiente programa de prueba:

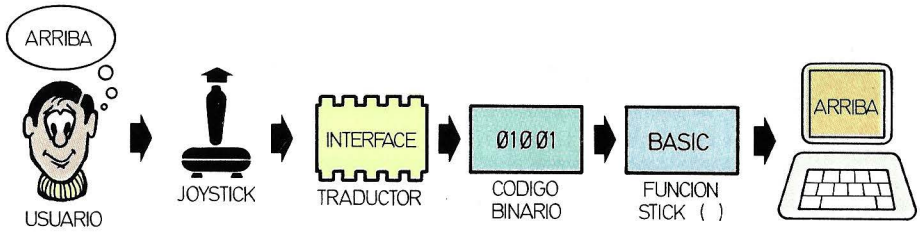
```
10 X=STICK (1)
20 PRINT X
30 GOTO 10
```

Al ejecutar este pequeño programa se imprimirá de forma continua en la pantalla el valor leído en cada instante por la función STICK(1), correspondiente al código de la posición en que se encuentre situado el "joystick" conectado a la toma 1. Para tener un punto de referencia, se supondrá que se dispone de un "joystick" imaginario que entrega los siguientes códigos para cada una de las posibles direcciones de la palanca de mando:

DIRECCION	CODIGO
REPOSO:	0
NORTE:	1
NORESTE:	2
ESTE:	3
SURESTE:	4
SUR:	5
SUROESTE:	6
OESTE:	7
NOROESTE:	8



Los códigos generados por la función STICK ( ) varían mucho de un ordenador a otro. La figura muestra dos de las distribuciones más comunes; los números que aparecen en el extremo de las flechas corresponden a los códigos asociados a cada sentido de desplazamiento de la palanca de mando.



Para que una orden que parte de la mente del usuario llegue a ser «comprendida» por el ordenador, es necesaria la intervención de una serie de dispositivos intermedios.

Bajo este supuesto, se pueden ya codificar las diferentes acciones a tomar. En el siguiente ejemplo cuyo listado (PROGRAMA 1) se reproduce junto al texto, se puede ver la utilización de sentencias IF/THEN junto con la función STICK ( ); el objetivo es, sencillamente, mover el carácter asterisco ("*") por toda la pantalla bajo las órdenes del "joystick".

El funcionamiento de este programa debe

STICK

Lee del "port" o toma de entrada que se especifique, el código que define la posición actual del joystick.

Formato: [<var>=] STICK (<exp.>)

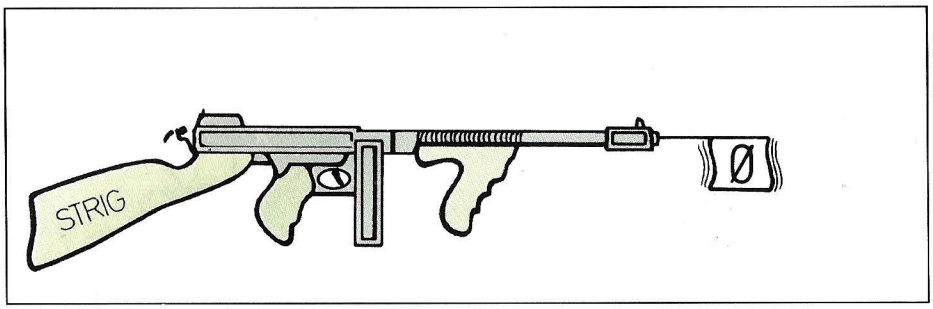
Ejemplo: 30 X=STICK (1)

STRIG

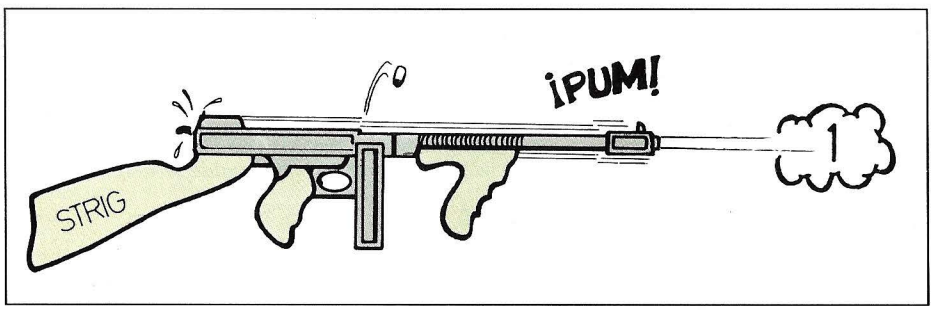
Lectura del botón de disparo del joystick conectado al "port" especificado.

Formato: [<var.>=] STRIG (<exp.>)

Ejemplo: 20 Y=STRIG (2)



La función STRIG ( ) toma el valor "∅" cuando no se está pulsando el botón de disparo.



Al accionar el botón de disparo, la función STRIG ( ) proporciona el valor "1".

```
 10 REM MANEJO DE UN JOYSTICK
 20 CLS:LET X=20:LET Y=10:LET X1=20:LET
    Y1=10
 25 REM BORRAR POSICION ANTERIOR
 30 PRINT AT (X1,Y1);" "
 35 REM IMPRIMIR NUEVA POSICION
 40 PRINT AT (X,Y);"*"
 50 LET X1=X:LET Y1=Y
 55 REM OBTENER CODIGO DE PALANCA
 60 LET A=STICK(1)
 65 REM JOYSTICK EN REPOSO
 70 IF A=0 THEN GOTO 60
 80 REM JOYSTICK EN ACCION
100 IF A=1 THEN LET Y=Y-1
110 IF A=2 THEN LET Y=Y-1;LET X=X+1
120 IF A=3 THEN LET X=X+1
130 IF A=4 THEN LET Y=Y+1:LET X=X+1
140 IF A=5 THEN LET Y=Y+1
150 IF A=6 THEN LET Y=Y+1:LET X=X-1
160 IF A=7 THEN LET X=X-1
170 IF A=8 THEN LET Y=Y-1:LET X=X-1
180 REM LIMITAR POSICION EN PANTALLA
190 IF X<2 OR X>37 THEN X=X1
200 IF Y<2 OR Y>21 THEN Y=Y1
210 GOTO 30
```

PROGRAMA 1: programa ejemplo ilustrativo del comportamiento de la función STICK ( ). El objetivo es desplazar un asterisco a través de la pantalla bajo el control de un joystick.

resultar de muy fácil asimilación a estas alturas del curso, por lo que tan solo basta con precisar que está escrito para un ordenador cuya visualización en pantalla es de 24 filas por 40 columnas en modo de texto, y con el origen de coordenadas para el comando PRINT AT (X,Y) situado en el vértice superior izquierdo de la pantalla. Si su ordenador no satisface este supuesto, será necesario adaptar el programa a las especificaciones de cada caso concreto, lo que no tiene mayor dificultad. Así mismo, habrá que modificar en la mayoría de los casos los valores evaluados por las instrucciones IF/THEN de las líneas 70 a la 170.

JOYSTICK Y BOTON DE DISPARO

Por el momento no se ha mencionado aún otro de los elementos que incorpora toda palanca de mando y que es sumamente útil para muy diversas funciones: el botón de disparo. Se trata de un pulsador que puede estar situado en diferentes puntos del "joystick" —en lo alto de la palanca o en su parte anterior a modo de gatillo, o incluso en la base del joystick—, y al que por norma se le suele asignar el papel de "gatillo" disparador para lanzar mortíferos misiles contra los más extraños alienígenas. En todo caso, y como se verá más adelante, su cometido no se limita únicamente al mencionado.

En su aspecto "hardware", el botón de disparo es totalmente análogo a los contactos de la base de la palanca de mando, pero en este caso sólo existe un contacto, que según esté pulsado o no, permite diferenciar los estados activo e inactivo. Ello se refleja en una señal eléctrica que es tratada normalmente por el mismo "interface" del "joystick". La función BASIC que "lee" el estado del botón de disparo es STRIG ( ), cuyo formato más general es el siguiente:

<Núm. de línea> X=STRIG (<exp.>)

Su funcionamiento, utilización y parámetros son totalmente análogos a los de la función STICK, salvo en lo relativo a que la función STRIG sólo proporciona dos posi-

bles valores, según esté o no pulsado el botón de disparo. Estos dos valores suelen ser: "1" cuando está pulsado el botón, y "0" cuando no está siendo accionado, o viceversa. Al igual que ocurría con la función STICK, estos valores suelen variar bastante de unos intérpretes a otros, por lo que de nuevo se hace necesaria una consulta al manual BASIC de cada equipo concreto. Desde luego, también puede ejecutarse un sencillo programa, similar al realizado en el caso de la función STICK, para conocer el código entregado por STRIG; programa que puede servir también para comprobar el correcto funcionamiento del botón de disparo:

```
10 X=STRIG(1)
20 PRINT X
30 GOTO 10
```

Siguiendo con la plena analogía de esta función con respecto a la comentada anteriormente, se puede ver en el siguiente ejemplo como también el estado del botón de disparo se puede examinar mediante una instrucción IF/THEN, que permitirá emprender la acción deseada:

```
10 CLS
20 PRINT "TODAVIA NO SE HA PULSADO EL
   BOTON"
30 IF STRIG(1)=0 THEN GOTO 30
40 CLS
50 PRINT "YA SE HA PULSADO EL BOTON"
60 END
```

La función STRIG se puede equiparar por tanto a un "conmutador software", que permite realizar una acción específica, saltar a otra zona del programa, o ejecutar una subrutina determinada, cuando sea accionado el "conmutador". Pero estas acciones sólo se pueden realizar cuando la ejecución del programa llegue a una línea BASIC que tome una decisión basada en el examen de la función STRIG.

CONTROL DE INTERRUPCIONES

Si quiere disponer de un "interruptor software", para realizar una determinada acción en el transcurso de la ejecución de un programa, se debe acudir a otra instrucción BASIC cuyo formato es el siguiente:

<Núm. de líneas> ON STRIG GOSUB <núm.> [, <núm.>...]

Esta instrucción genera una interrupción del procesador en el preciso instante en que alguno de los botones de disparo es accionado, sin esperar a llegar a la línea que examina el resultado de la función STRIG. Con esto se consigue una respuesta inmediata, pasando el ordenador a ejecutar directamente la subrutina indicada. Como normalmente existen varias tomas de entrada a las que se pueden conectar los "joysticks", se pueden situar detrás de la palabra GOSUB varios números de línea, separados por comas; se ejecutará la subrutina cuyo número de línea aparezca en el lugar correspondiente del número de orden de la toma a la que se está conectado el botón de disparo accionado para interrumpir el programa. Así, en el siguiente ejemplo:

```
10 ON STRIG GOSUB 1000,2000,3000
```

se ejecutará la subrutina de la línea 1000 si se acciona el botón de disparo conectado a la toma 0; la subrutina de la línea 2000, si se acciona el botón de disparo de la toma 1, y la de la línea 3000 si se actúa sobre el de la toma 2. Cuando se llega a la sentencia RETURN de estas subrutinas, el control retorna al punto del programa en el que se produjo la interrupción, continuando la ejecución normalmente.

Pero este tema no acaba aquí, sino que también se pueden controlar estas interrupciones programadas por medio de la instrucción ON STRIG GOSUB. Después de haber ejecutado esta instrucción, se podrá hacer que sea efectiva la interrupción originada por la pulsación del botón de disparo, se podrán anular las interrupciones, o bien se podrán "congelar" temporalmente, produciéndose la interrupción una vez que sean nuevamente permitidas. Con ello se tendrá un control total del programa, de forma que en ciertos momentos en los que no convenga que se produzca una interrupción, éstas sean anuladas o bien "memorizadas" temporalmente, para "acusarlas" una vez finalizada la ejecución de la zona en la que no se deseaban interrupciones. Este control se logra con la instrucción siguiente:

<Núm. de línea> STRIG (<exp.>) [ON/OFF/STOP]

Con la referida instrucción y mediante el

Moviendo la palanca de mando el usuario puede comunicar al ordenador con celeridad órdenes que, normalmente, afectarán al desplazamiento de un objeto sobre la pantalla.

ON STRIG GOSUB

Bifurca a una subrutina al producirse una interrupción provocada por un botón de disparo del joystick.

Formato: ON STRIG GOSUB <nl.> [,..., <nl.>]

Ejemplos: 10 ON STRIG GOSUB 1000
20 ON STRIG GOSUB 1000, 2000, 3000

STRIG ON/OFF/STOP

Permite, prohíbe o desactiva momentáneamente la interrupción que provoque el botón de disparo especificado. Está relacionada con el comando ON STRIG GOSUB.

Formato: STRIG (<exp.>) [ON/OFF/STOP]

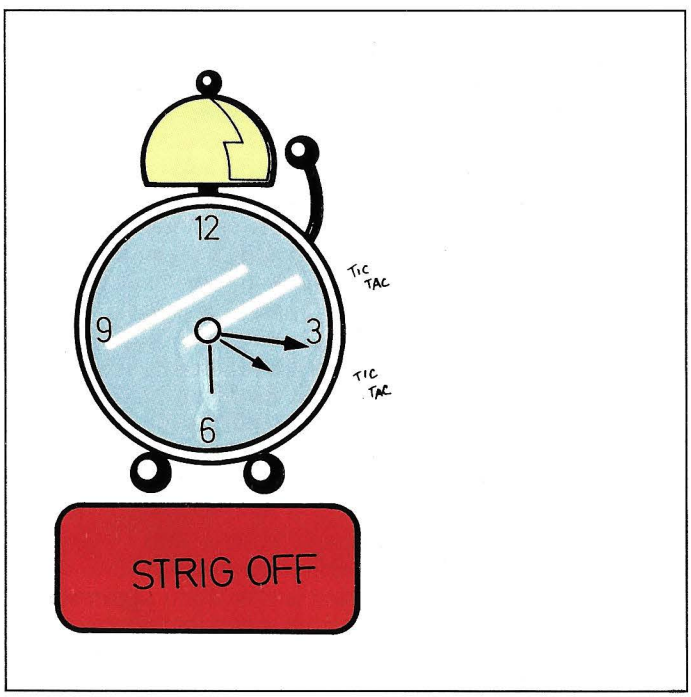
Ejemplos: 10 STRIG ON
20 IF I=ϕ THEN STRIG OFF
50 STRIG STOP

número consignado en la zona <exp.>, se puede elegir el botón de disparo cuya detección con la instrucción ON STRIG GOSUB, se desea activar, desactivar o memorizar. Para permitir la interrupción generada por un determinado botón de disparo se debe utilizar este comando en la forma: STRIG ( ) ON, con lo que al pulsar el botón afectado se pasará a ejecutar la subrutina cuyo número de línea se haya especificado con anterioridad en la instrucción ON STRIG GOSUB.

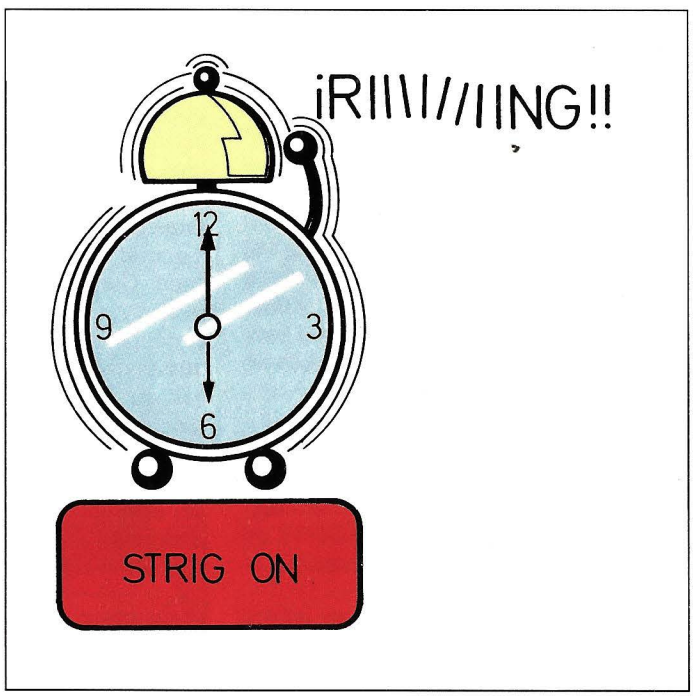
Si lo que se quiere es desactivar esa interrupción, será preciso hacer uso de STRIG ( ) OFF; con ello, el BASIC ignorará las pulsaciones del botón de disparo destinadas a generar una interrupción.
Al ejecutar STRIG ( ) STOP, la interrupción no se activará al pulsar el botón de disparo, si bien este hecho será anotado internamente, y en el mismo momento en el que se vuelva a activar el control mediante una instrucción STRIG ( ) ON, la interrupción tendrá lugar, aunque no se esté pulsando el botón de disparo en ese preciso momento.

EL JUEGO DEL "PING-PONG"

En la ciencia informática el empleo de anglicismos está a la orden del día. Esto viene a cuento del nombre que recibe otro dispositivo periférico, bastante popular, conocido por "PADDLE" ("raqueta de ping-pong"). Despista bastante este nombre al ver el aspecto externo de tal dispositivo, el cual consta de una caja con un mando giratorio que mueve una resisten-



Mediante el comando STRIG OFF se pueden inhibir o desactivar las interrupciones generadas por el botón de disparo a través de la sentencia ON STRIG GOSUB.



El comando STRING ON permite que se produzcan interrupciones mediante la pulsación del botón de disparo del joystick.

TABLA DE CONVERSION

ORDENADOR	STICK	STRIG	ON STRIG GOSUB	STRING ON/OFF/STOP	PADDLE	MOTOR ON/OFF
	STICK(<n>)	STRIG(<n>	ON STRIG GOSUB <nl.>[,...,<nl.>]	STRIG(<n>)ON/OFF/STOP	PADDLE(<n>)	MOTOR ON/OFF
APPLE II (APPLESOFT	—	—	—	—	PDL(<n>)	—
APRICOT (M-BASIC)	—	—	—	—	—	—
ATARI	STICK(<n>)	STRIG(<n>)	—	—	PADDLE(<n>)	—
CBM 64	—	—	—	—	—	—
DRAGON	JOYSTK(<n>)	—	—	—	—	MOTOR ON/OFF
EQUIPOS MSX	STICK(<n>)	STRIG(<n>)	ON STRIG GOSUB<nl.>[,...,nl.>]	STRIG(<n>)ON/OFF/STOP	PDL(<n>)	MOTOR ON/OFF
HP-150	—	—	—	—	—	—
IBN PC	STICK(<n>)	STRIG(<n>)	ON STRIG(<n>) GOSUB<nl.>	STRIG(<n>)ON/OFF/STOP	—	MOTOR<exp.>
MPF	—	—	—	—	—	—
NCR DM-V (MS-BASIC)	—	—	—	—	—	—
NEW BRAIN	—	—	—	—	—	—
ORIC	—	—	—	—	—	—
SHARP MZ-700 (MZ-BASIC)	—	—	—	—	—	—
SINCLAIR QL	—	—	—	—	—	—
SPECTRAVIDEO	STICK(<n>)	STRIG(<n>)	ON STRIG GOSUB<nl.>[,...,<nl.>]	STRIG(<n>)ON/OFF/STOP	—	MOTOR ON/OFF
ZX-SPECTRUM	—	—	—	—	—	—

<n>: número de "port" o toma de entrada. <nl.>: número de línea inical de una subrutina. <exp.>: expresión numérica.

cia variable o "potenciómetro". El origen del nombre se debe a que era empleado por el primer videojuego que se comercializó (el PING-PONG de ATARI), para gobernar las raquetas que impedían que la pelota se colase por el fondo de la pantalla... De ahí que se quedara con el nombre de "raqueta".

El "paddle" es un periférico de entrada de datos, al igual que el "joystick"; pero a diferencia de éste, que sólo puede entregar ocho códigos, el paddle entrega más de doscientos códigos diferentes. Ello no significa que se utilice para elegir entre todas estas opciones, sino que su principal uso es para controlar fenómenos que varíen de forma continua entre dos valores extremos, como por ejemplo los comandos de sonido o la posición horizontal o vertical de una determinada figura re-

PADDLE

Lectura del código generado por un "paddle" o "raqueta" cuyo número de "port" se especifica.

Formato: [<var.>=] PADDLE (<exp.>)

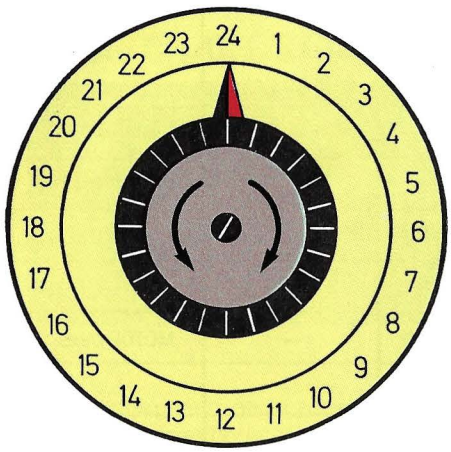
Ejemplos:
```
10 X=PADDLE (1)
20 IF PADDLE (2) > 50 THEN GOTO 20
```

MOTOR ON/OFF

Controla el motor de un casete de audio, activando o desactivando un relé.

Formato: MOTOR [ON/OFF]

Ejemplos: 10 MOTOR ON
20 IF VAR=∅ THEN MOTOR OFF



El PADDLE o potenciómetro continuo, genera un valor comprendido entre dos límites que es "leído" por medio de la función PDLC).

presentada en la pantalla. Su funcionamiento se basa en hacer variar una tensión de forma continua entre dos valores mediante el potenciómetro. Esta señal eléctrica es codificada en forma digital mediante un convertidor analógico-digital, a cuya salida se accede mediante la función "PADDLE" cuyo formato es el siguiente:

<Núm. de línea> X=PADDLE(<exp.>)

De nuevo, se observa que este formato es igual al de las funciones STICK y STRIG, por lo que no es preciso volver a comentar aquí lo dicho anteriormente para esas funciones. La única diferencia es que la función "PADDLE" proporciona como resultado un valor comprendido, normalmente, entre 0 y 255, aunque este rango depende como siempre de cada dialecto BASIC. Este valor se manipulará a través de alguna variable en la que será almacenado, o bien directamente utilizando la función "PADDLE" como argumento de otra función, como puede observarse en el ejemplo que sigue a estas líneas, el cual simula una pantalla de "TELE-SKETCH":

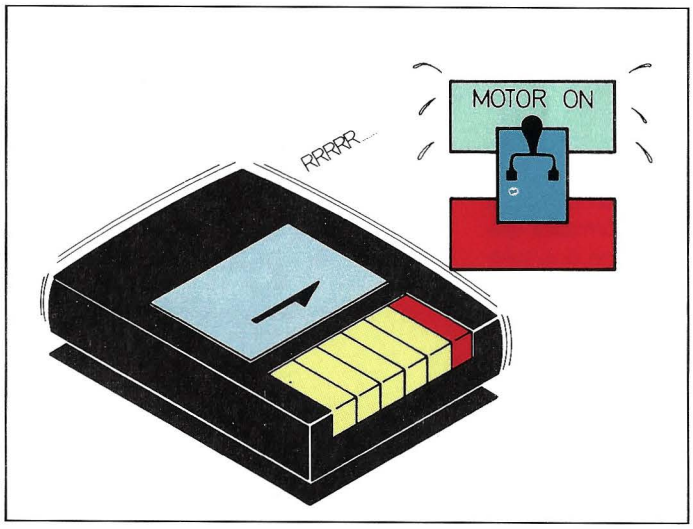
```
10 SCREEN 3:REM MODO GRAFICO
20 PSET(PADDLE(1),PADDLE(2))
30 GOTO 10
```

CONTROL DEL MONITOR

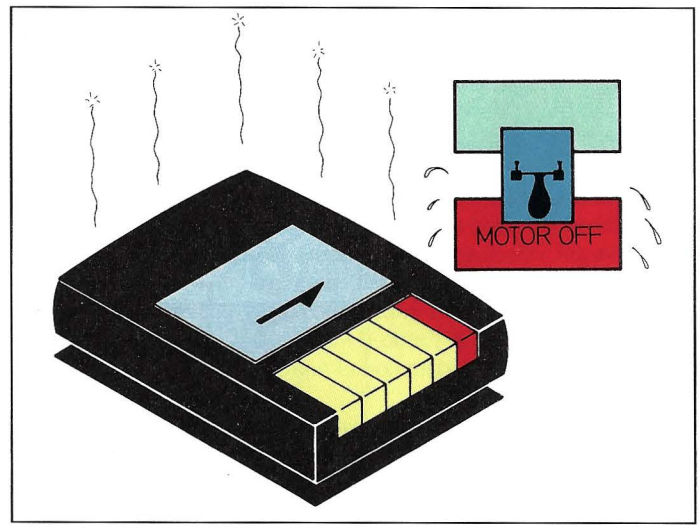
El BASIC no sólo permite controlar periféricos de entrada como los vistos hasta ahora, también hay periféricos de entrada y salida o sólo salida que pueden ser gobernados a través de órdenes en BASIC. Así, por ejemplo, puede ser útil en ciertos casos la posibilidad de conectar y desconectar el motor de un casete de audio, para hacer avanzar automáticamente a la cinta hasta un punto determinado en el que se encuentre situada cierta información. Para esta misión el BASIC dispone de un comando de formato muy simple que permite conectar y desconectar a voluntad el motor del casete:

<Núm. de línea> MOTOR [ON]/[OFF]

Si se elige la opción MOTOR ON, se activa un relé cuyos contactos pueden conmutar una pequeña corriente (de alrededor de 150 mA), capaz de proporcionar energía para el movimiento de un motor de casete, con lo que éste será operativo. Para desconectar el motor habrá que ejecutar la orden opuesta: MOTOR OFF. La utilización de este comando no sólo se limita al gobierno del casete. Puesto que los contactos destinados al control remoto del casete están disponibles normalmente en algún conector externo, se pueden acoplar a él otros dispositivos; pero esto ya entra dentro del campo del bricolage electrónico ajeno al fin de esta obra.



El comando MOTOR ON conecta el dispositivo de arranque del motor de arrastre del magnetófono a casete.



MOTOR OFF apaga o desconecta el motor de arrastre del casete.

Forth (y 12)

Complementos finales

El capítulo de complementos estará dedicado a hablar de algunas palabras no abordadas en los capítulos precedentes; palabras que cumplen funciones muy diversas y que serán agrupadas en base a la función que desempeñan.

MANEJO DE MEMORIA

Las palabras que más directamente manejan la memoria son las siguientes: I y que permiten leer o modificar el valor de dos bytes de memoria, y CI y C @ que permiten realizar esa misma operación con un solo byte. Además de estas, existen otras palabras FORTH catalogables en este apartado funcional:

HERE Proporciona una dirección de memoria que corresponde a la primera posición libre tras el diccionario. Esta dirección de memoria se depositará en la cima de la pila a la disposición del operador.

```
HERE . <CR>

HERE . 15897 OK
```

CALL Ejecuta una rutina en código máquina situada en una dirección de memoria que se extrae de la pila.

Para manejar las operaciones de Entrada/Salida existen dos instrucciones especializadas:

OUT Envía un dato a un acceso para comunicación externa cuya dirección ha de especificarse.
IN Recoge un dato de un acceso cuya dirección se recoge en la pila.

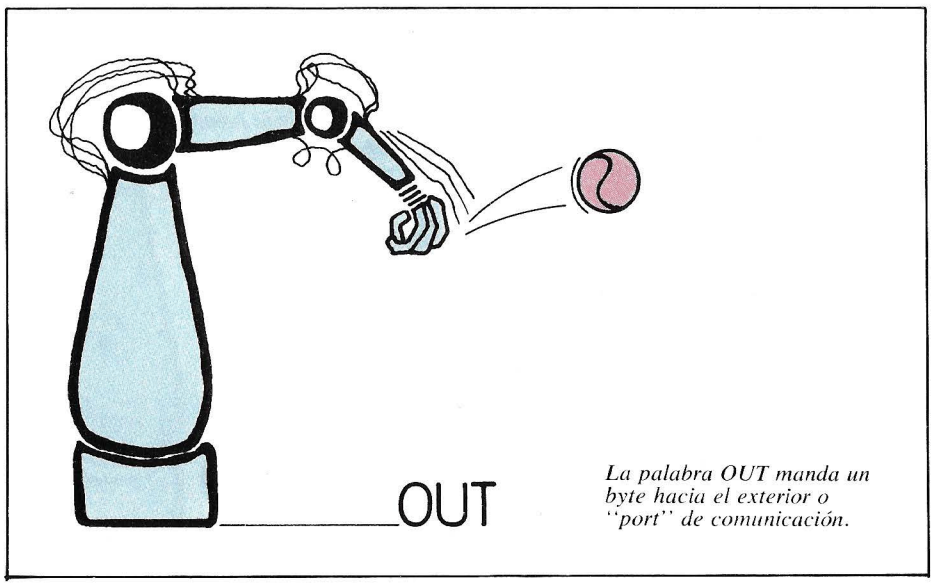
INSERCION DE COMENTARIOS

Al revisar un programa transcurrido algún tiempo a partir del momento en que fue confeccionado, ocurre con frecuencia que el usuario ha olvidado por completo su estructura y contenido. Para evitar tal inconveniente es preciso incluir algunas notas dentro del programa que, más tarde, permitirán recordar ciertos detalles importantes del mismo. Al efecto, la nota u observación a incluir debe aparecer encerrada entre paréntesis; por ejemplo:

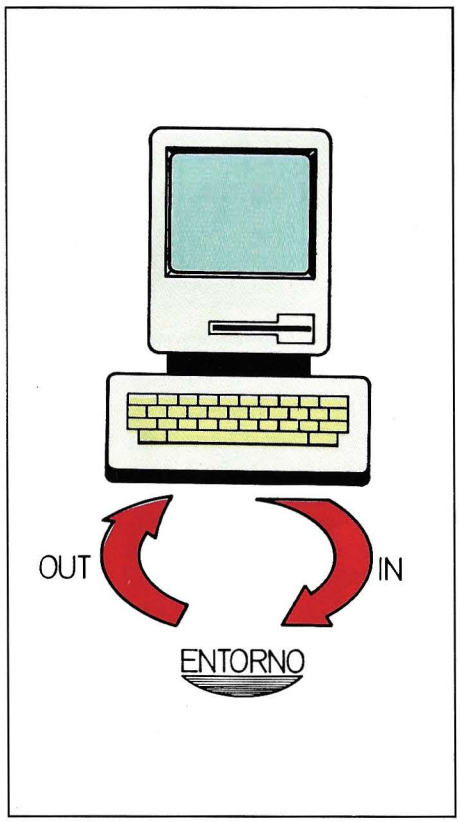
```
: PROGRAMA
2 127 +
(ESTE PROGRAMA SIMPLEMENTE SUMA
2 Y 127 E IMPRIME EL RESULTADO)
.
;
```

DENTRO DE LAS DEFINICIONES DE PALABRAS

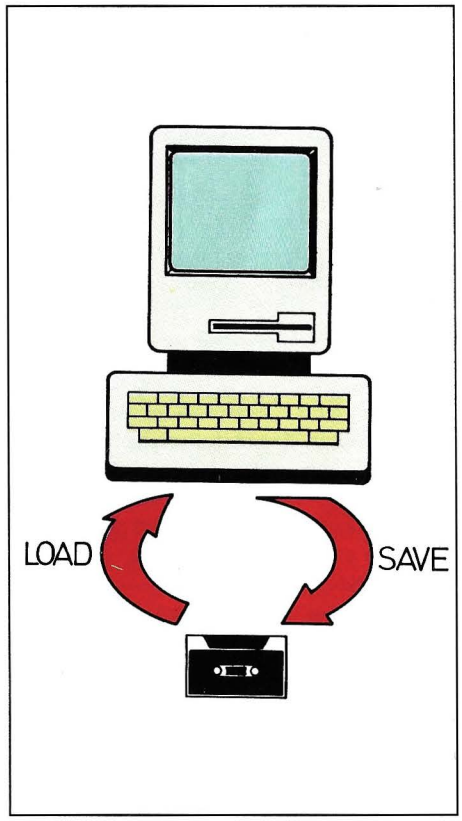
Durante el proceso de definición de una palabra FORTH puede ser deseable pasar



La palabra OUT manda un byte hacia el exterior o "port" de comunicación.



En FORTH, el usuario cuenta con las palabras IN y OUT como herramientas para facilitar la comunicación del ordenador con el exterior.



Las palabras SAVE y LOAD permiten, respectivamente, grabar y cargar diccionarios del casete.

a modo intérprete para efectuar alguna operación. Â la hora de efectuar estas operaciones hay que recurrir a sendas palabras FORTH coincidentes con los símbolos de "corchete". Veamos un ejemplo:

```
: PRUEBA
[ 2 BASE C! ]
111
[ DECIMAL ]
7 + .
;
```

La anterior definición de la palabra PRUEBA equivalente, en definitiva, a la siguiente:

```
: PRUEBA
7 7 + .
;
```

Ya que 111 fue definido a través de un paso intermedio a binario. También es posible definir palabras que actuarán cual si se tratara realmente de una secuencia de instrucciones semejantes a las que se encierran entre las palabras FORTH [ ]. Las referidas palabras se definen según el método habitual y, a continuación, ejecuta la palabra IMMEDIATE:

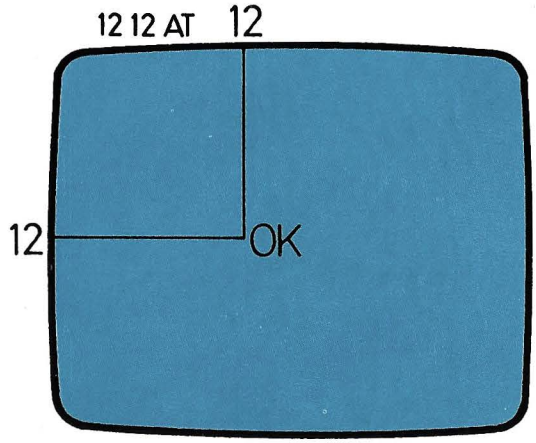
```
: BASE2
  2 BASE C!
;

IMMEDIATE <CR>
```

Otro ejemplo puede ser:

```
: BASE10
DECIMAL
;
IMMEDIATE<CR>
```

A partir de este momento se puede susti-

```
12 12 AT <CR>
```



El cometido de la palabra AT no es otro que posicionar el cursor en un determinado punto de la pantalla.

tuir la definición realizada anteriormente con la colaboración de las palabras [y] por las palabras que se acaban de definir. Veamos lo que sucedería con la anterior definición:

```
: PRUEBA
BASE2 111
BASE10 7
```

En efecto, su funcionamiento coincidirá con el asociado a la primera definición de las mismas.

LITERAL Es una palabra que puede resultar útil especialmente cuando se trabaja con la palabras [y ']. Permiten pasar un elemento de la pila a la definición de una palabra. Veamos un ejemplo:

```
: PRU
[ 2 2 +] LITERAL
12 +
.
;
```

Al ejecutar esta palabra (PRU LCR>) se obtendrá el siguiente resultado:

```
PRU 16 OK
```

Mientras que si ordenamos un listado de la misma (con LIST PRU <CR>) el resultado en la pantalla será:

```
: PRU
4 12 + .
; OK
```

MANEJO DEL CASETE

SAVE Permite grabar un diccionario de palabras FORTH en casete (se graban tan sólo las palabras que el usuario ha ido definiendo). Su formato es:

SAVE <nombre>

LOAD Permite leer un diccionario de palabras almacenado en casete:

LOAD <nombre>

VERIFY Permite comprobar si un diccionario grabado en casete es igual que el que se encuentra en memoria.

VERIFY <nombre>

BLOAD Lee un bloque de bytes almacenados en casete.

<comienzo> <longitud> BLOAD <nombre>

En el caso de desear que la dirección de comienzo y la longitud sean las mismas con las que se grabó, sin necesidad de especificar dichos parámetros, hay que utilizar la formulación siguiente:

0 0 BLOAD <nombre>

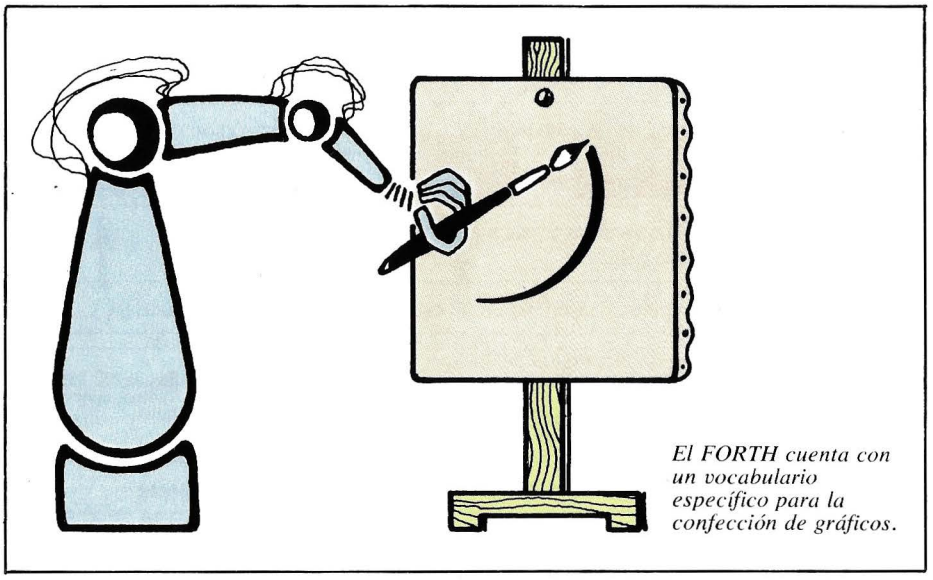
BSAVE Graba un bloque de bytes en casete.

<comienzo> <longitud> BSAVE <nombre>

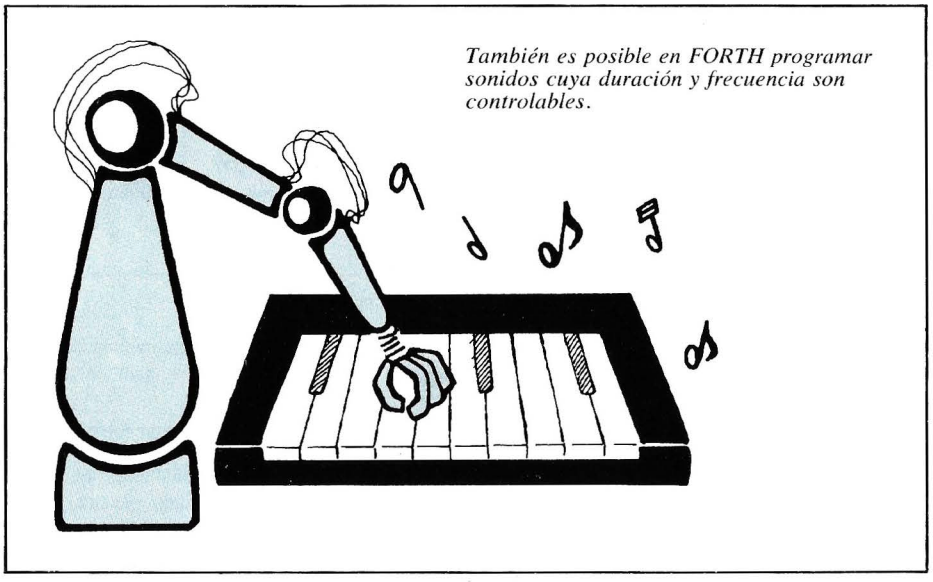
BVERIFY Compara un bloque de bytes almacenados en casete con el mismo bloque residente en memoria.

GRAFICOS EN FORTH

AT Permite posicionar el cursor en cualquier punto de la pantalla. Su formulación habitual es:



El FORTH cuenta con un vocabulario específico para la confección de gráficos.



También es posible en FORTH programar sonidos cuya duración y frecuencia son controlables.

TABLA DE ORDENES FORTH

PALABRA	DESCRIPCION	TIPO
HERE	Da la dirección del último byte libre tras el diccionario	Manejo de memoria
CALL	Ejecuta una rutina en código máquina cuya dirección de comienzo ha de encontrarse previamente en la pila	Manejo de memoria
IN	Recoge un byte de un acceso externo	Entrada/salida
OUT	Envía un byte hacia un acceso de comunicación	Entrada/salida
[	Pasa a modo intérprete	Palabra auxiliar
]	Pasa a modo complilación	Palabra auxiliar
IMMEDIATE	Hace que la última palabra definida se ejecute dentro de cualquier definición como paso a modo intérprete	Palabra auxiliar
(	Comienzo de un comentario	Inserción de comentarios
)	Fin de un comentario	Inserción de comentarios
LITERAL	Pasa a un valor de la pila a la definición de una palabra	Palabra auxiliar
SAVE	Almacena un diccionario en un casete	Manejo de casete
LOAD	Recoge un diccionario del casete	Manejo de casete
VERIFY	Comprueba que la grabación de un diccionario se ha efectuado correctamente	Manejo de casete
BSAVE	Almacena un bloque de bytes en casete	Manejo de casete
BLOAD	Recoge un bloque de bytes del casete	Manejo de casete
BVERIFY	Comprueba que un bloque de bytes ha sido grabado correctamente	Manejo de casete
AT	Posiciona el cursor en una posición determinada de la pantalla	Gráficos
PLOT	Dibuja un pixel en el punto y las coordenadas que se especifiquen	Gráficos
INVIS	Hace que no se envíen a la pantalla los mensajes informativos	Gráficos
VIS	Hace que vuelvan a aparecer los mensajes informativos	Gráficos
BEEP	Emite un sonido de una frecuencia y duración especificada	Sonido

X Y AT

PLOT Imprime un pixel en un punto de la pantalla.

X Y <modo> PLOT

Son cuatro los posibles modos de impresión:

0:Color negro
1:Color blanco
2:No impresión
3:Inversión

Por ejemplo, la ejecución de:

12 12 1 PLOT <CR>

dará la siguiente respuesta en pantalla:

12 12 1 PLOT OK

además de imprimir un pixel en la posición de coordenadas 12, 12. Los mensajes que el ordenador visualiza constituyen un inconveniente que es preciso evitar cuando se desea realizar un gráfico en pantalla. Para obviar la aparición de estos mensajes existe una palabra FORTH al efecto:

INVIS Hace que no aparezca en la pantalla los mensajes habituales. También existe una palabra FORTH adecuada para ordenar que los mensajes vuelvan a aparecer en el momento oportuno; esta es:

VIS Reaparecen en la pantalla los mensajes generados por la máquina. Es importante conocer el juego de caracteres disponibles en el ordenador que se está utilizando. Al efecto, puede resultar muy conveniente ejecutar la siguiente palabra FORTH que permite visualizar todo el repertorio de caracteres:

```
: CARAC
256 0
DO
 I EMIT
LOOP
;
```

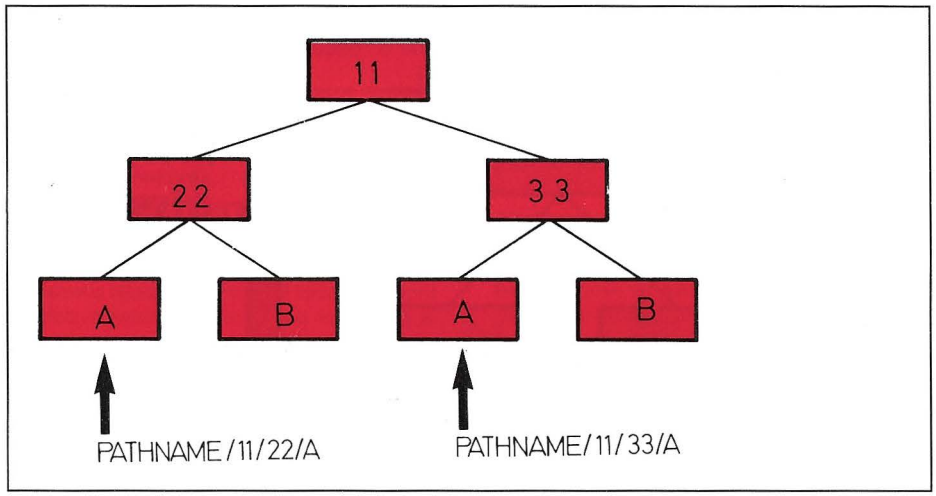
EL SONIDO

El FORTH también brinda la posibilidad de producir sonidos, a tal efecto existe la palabra BEEP.
BEEP produce dos sonidos necesitando para ello que en la pila se encuentren dos valores. Estos son interpretados como la duración en msg (el valor más cercano a la cima de la pila) y el tono deseado (el siguiente elemento localizado en la pila). El sonido que se produce será agudo cuanto más pequeño sea el número especificado para controlar el tono.

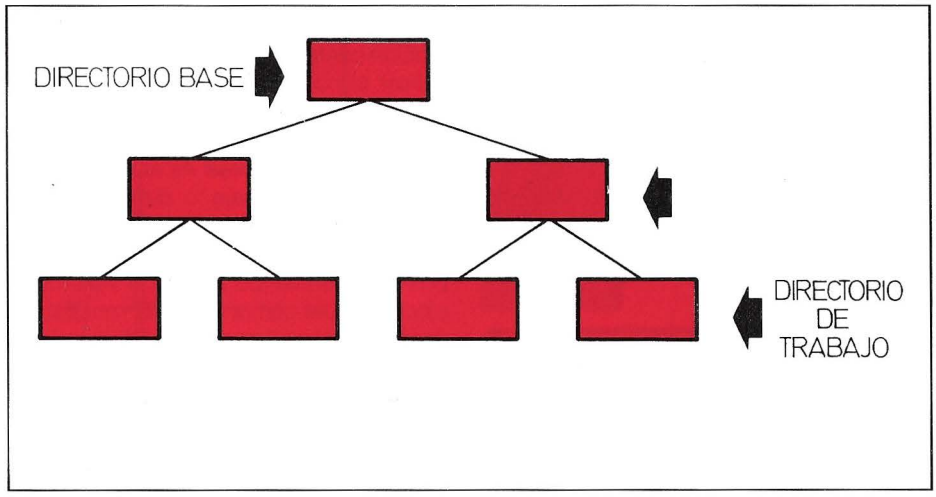
Unix (3)

Estructura interna y gestión de los ficheros

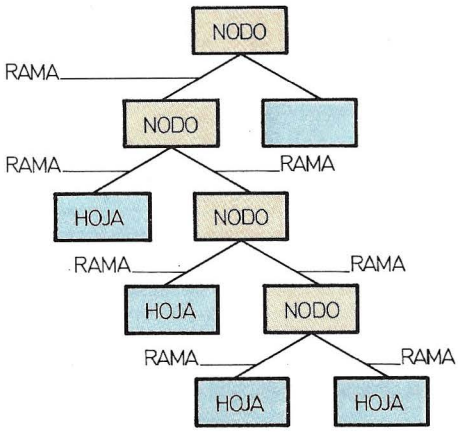
Todo sistema operativo ha de ser capaz de controlar una gran cantidad de ficheros y de presentar al usuario un método claro para la localización de los mismos, de manera que éste pueda acceder fácilmente a la información contenida en ellos. Para cumplir estos objetivos el sistema operativo UNIX trabaja haciendo uso de una estructura en forma de árbol, la cual es muy común en los modernos sistemas operativos, como por ejemplo en el MS-DOS. Una estructura de este tipo comienza, en su raíz, en lo que se denomina un directorio. De este directorio empiezan a bifurcarse otros directorios, denominados subdirectorios, así como también ficheros individualizados. Esta estructura puede prolongarse indefinidamente, si bien está limitada por el espacio físico de almacenamiento disponible. Por la propia naturaleza del conjunto es posible encontrar ficheros con un mismo nombre en distintos directorios. Así, por ejemplo, imagine el caso de una empresa que almacena los datos necesarios para sus operaciones diarias en un ordenador. Dado que la empresa está dividida en varios departamentos,



Aunque coincida la denominación de los ficheros, éstos difieren bajo la perspectiva del sistema operativo, puesto que su "pathname" es distinto.



Directorio base y directorio de trabajo son conceptos distintos, tal como ilustra la figura.

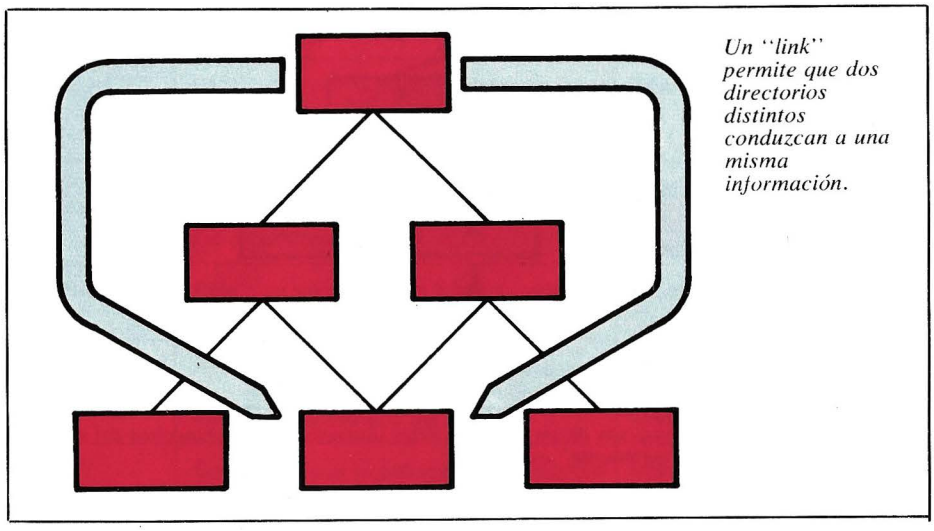


Estructura jerárquica de los archivos en UNIX y nomenclatura genérica.

para racionalizar las tareas se asigna a cada uno de ellos un directorio cuyo nombre coincide con el del departamento: "VENTAS", "PRODUCCION"... En estas condiciones es posible tener varios ficheros cuyo nombre sea "PERSONAL", conteniendo los nombres de los empleados que trabajan en cada uno de los departa-

mentos, siempre y cuando cada uno de estos ficheros cuelguen del directorio correspondiente a su departamento.
Este esquema enlaza con lo que se denomina "pathname" en cualquier sistema operativo que ofrece esta posibilidad de ordenación. A la hora de buscar cualquier fichero, el sistema operativo sigue una trayectoria de arriba abajo a lo largo de las diferentes ramas de la estructura, pasando desde el directorio raíz por todos los diferentes subdirectorios hasta alcanzar el fichero crítico que se está buscando. Esta secuencia de búsqueda es lo que se denomina "pathname" y en tanto en cuanto sea distinta de un fichero a otro, estos serán distintos. Así, en el ejemplo anterior y suponiendo que el directorio raíz se denomina "EMPRESA", los "pathnames" de los ficheros de personal serán:

EMPRESA / VENTAS / PERSONAL
EMPRESA / PRODUCCION / PERSONAL
. . .



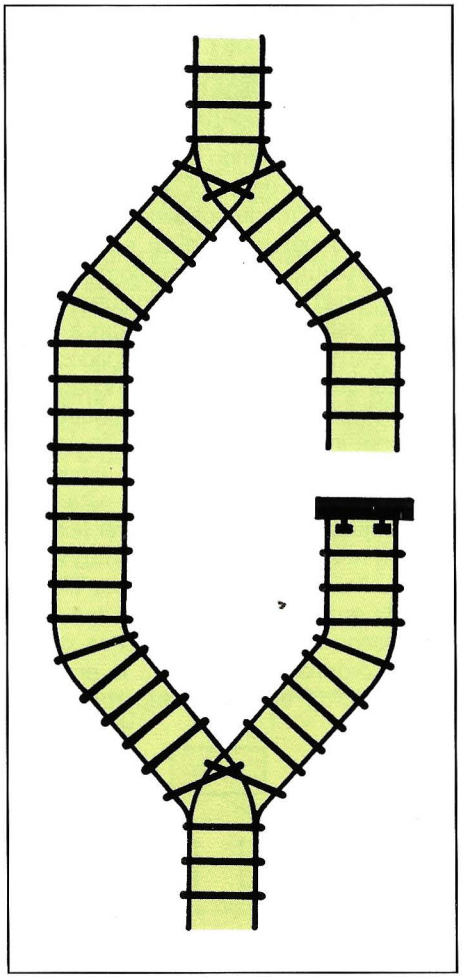
Un "link" permite que dos directorios distintos conduzcan a una misma información.

DIRECTORIOS

Cuando un usuario se conecta al sistema normalmente —y a través de la figura del administrador del sistema que habrá definido previamente unos derechos de acceso así como unas áreas de trabajo permitidas—, se verá posicionado en un determinado directorio. Este directorio es el denominado directorio base, y su nombre se verá presentado en la pantalla haciendo uso del comando PWD. Si tras haber ejecutado esta orden se deseara conocer cuáles son los ficheros contenidos en el mismo, bastaría con invocar al comando IS, el cual presentaría una lista de todos los ficheros del referido directorio.
Sin embargo, existe una nueva figura entre los directorios: el directorio de trabajo. Dentro de la estructura jerárquica hay varios directorios a los que es posible acceder y en los cuales, una vez situado en ellos el usuario, para operar con la información contenida, en un fichero basta con indicar el nombre del mismo, sin necesidad de especificar el "pathname" completo. El directorio en el cual se posiciona el usuario para trabajar es el denominado directorio de trabajo o directorio "current". Si este directorio no coincide con el directorio base en el que el usuario es posicionado cuando accede al sistema, la ejecución del comando PWD mostrará el "pathname" que conduce al directorio en cuestión.
Una vez posicionado en un directorio, el usuario puede crear otro directorio por medio de la orden MKDIR, abreviatura de la frase "make directory". Para ello debe especificar el "pathname" completo del nuevo directorio, el cual será de orden inferior al directorio de trabajo, siendo un subdirectorio del mismo, caso de especificarse así. Regresemos al ejemplo propuesto anteriormente, relativo a la estructura de la información de una empresa; suponiendo que nos encontramos en el subdirectorio "VENTAS", para crear un nuevo subdirectorio dentro del mismo, de nombre "CLIENTES", bastaría con teclear la siguiente orden:

MKDIR /EMPRESA / VENTAS / CLIENTES

Una vez realizada estra operación, la ejecución del comando PWD revelará que no nos hemos movido del directorio de trabajo en el cual nos encontramos. A su vez, la ejecución del comando IS dará como resultado, suponiendo que no se hayan introducido más ficheros que los indicados en el curso de la explicación, lo siguiente:



La eliminación de un "link" no impide el acceso a la información afectada.

PERSONAL CLIENTES

Como puede observarse, el sistema operativo UNIX no hace ningún tipo de dife-

renciación, a la hora de presentar los datos, entre subdirectorios y ficheros. Otra característica de este sistema operativo reside en el hecho de que para recabar información del directorio del cual depende el directorio de trabajo, basta con añadir dos puntos a continuación del comando, en la forma:

IS . .

Si la información a recabar fuera la correspondiente al directorio de trabajo, este sería representado por un solo punto.
Como se ha visto, el sistema operativo UNIX permite el desplazamiento por los distintos directorios, según las necesidades de la gestión informática que haya que llevar a cabo. El comando que permite llevar a cabo tal desplazamiento es CD, sinónimo de "Change Directory"; este comando asocia al usuario con un determinado directorio de trabajo, exigiendo que se especifique el "pathname" de dicho directorio en la forma siguiente:

CD / EMPRESA / VENTAS / CLIENTES

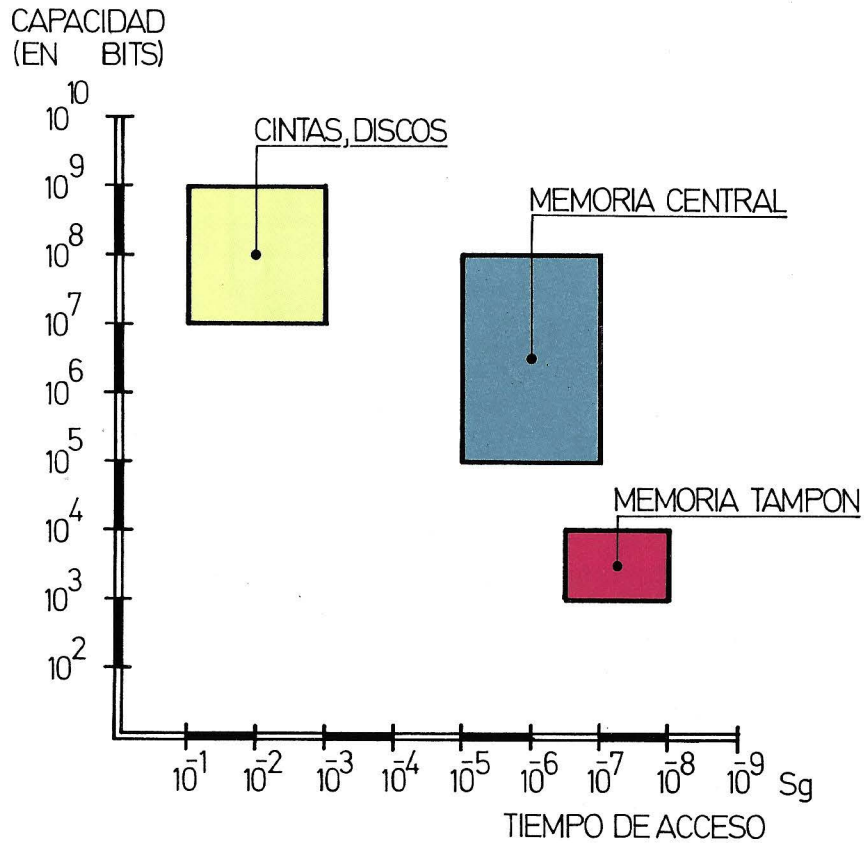
Algo a señalar y que separa en cierta forma el UNIX de otros sistemas operativos, es el hecho de que cuando un usuario se conecta al sistema su directorio base no se convierte automáticamente en su directorio de trabajo sino que ha de ser definido como tal, caso de ser esto necesario de acuerdo con las necesidades del sistema. Al efecto se hace uso del comando CD.

LINKS

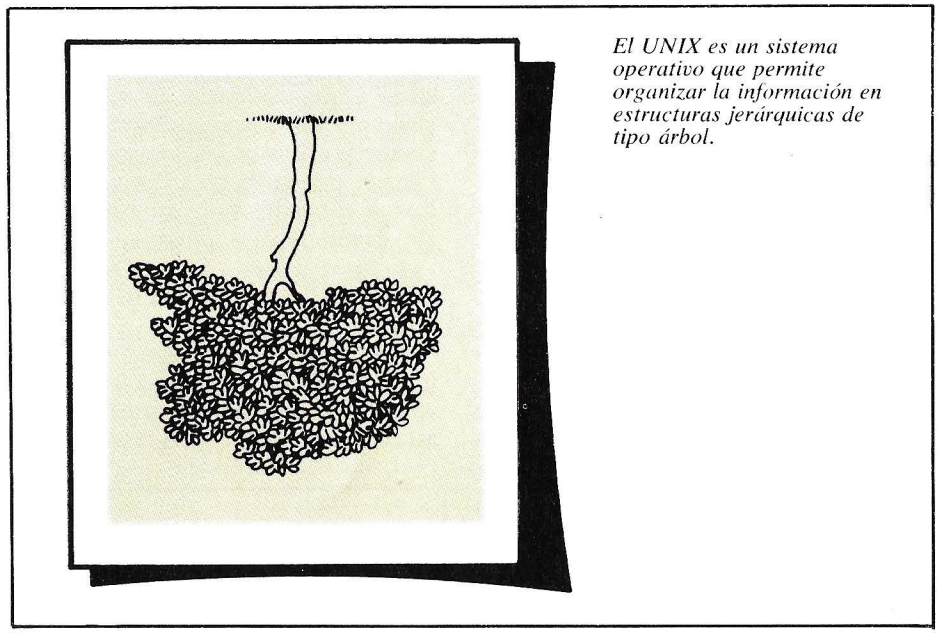
Bajo este nombre se conocen los diferentes punteros electrónicos que permiten

Estructura jerárquica de la memoria del ordenador

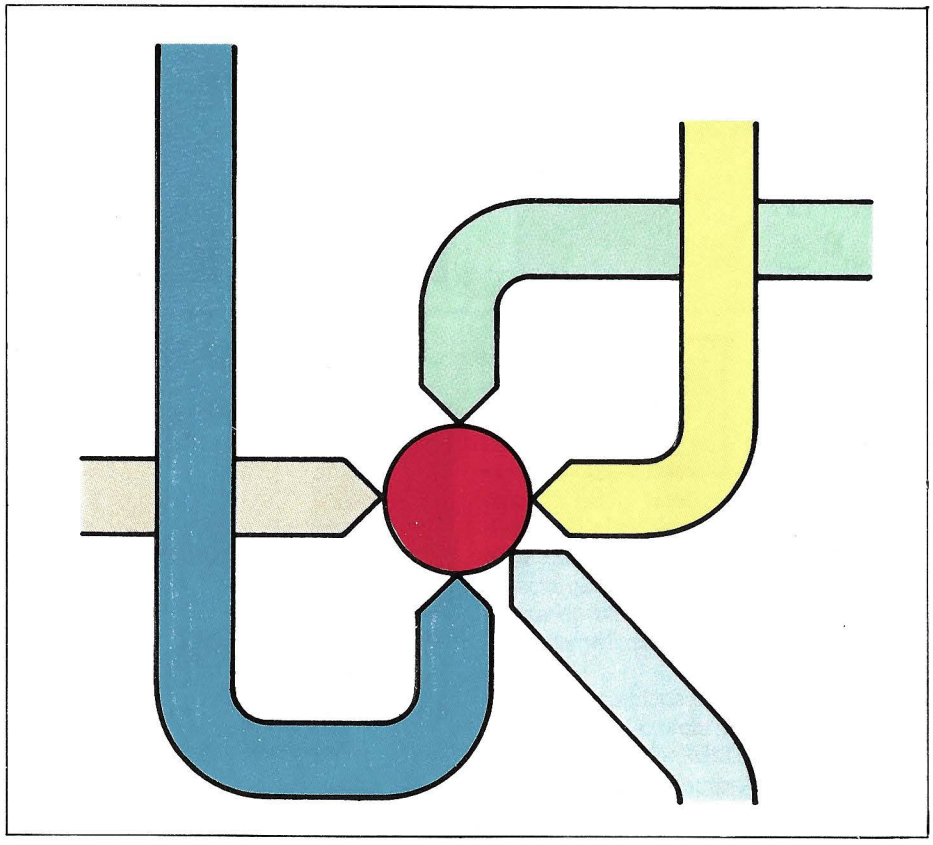
El usuario de un ordenador se verá limitado a la hora de generar sus programas, para el mismo por la capacidad de la memoria disponible en el equipo, toda vez que ésta es el soporte básico de la información (datos, variables, etc.) a procesar así como del propio programa.
Para el usuario, lo ideal sería tener una memoria central de elevada capacidad y con una gran velocidad de acceso; sin embargo, conseguir este objetivo sería sumamente costoso en términos económicos, así como de una complejidad tal en circuitería que anularía las posibles ventajas. Es por ello que dentro de la memoria se ha establecido una jerarquía de dos niveles: por un lado se encuentra la memoria central, de elevada capacidad de almacenamiento y con tiempos de acceso rápidos, y por otro la memoria auxiliar de mucha mayor capacidad pero con tiempos de acceso dilatados. Dentro del primer grupo cabe diferenciar entre dos tipos. El primero coincide propiamente con la memoria central, construida, normalmente a base de dispositivos semiconductores, y cuyos tiempos de acceso oscilan entre los 250 nanosegundos y los dos microsegundos. Por otro lado se encuentran las memorias "tampón" o memorias de anotaciones, con una elevada velocidad de acceso pero escasa capacidad, y que son utilizadas en los canales de comunicaciones o en la propia CPU como memoria local; este tipo presenta tiempos de acceso variables entre los 50 y los 200 nanosegundos.
Dentro del grupo de las memorias auxiliares caben las memorias de almacenamiento masivo o dispositivos con acceso selectivo a la información, de elevada capacidad y con tiempos de acceso relativamente elevados. En la actualidad, el predominio en este tipo de memorias corresponde a las unidades de disco, con velocidades de acceso de aproximadamente 35 milisegundos en algunos casos. El otro apartado dentro de las memorias de este tipo lo ocupan las memorias de fichero, básicamente las unidades de cinta magnética, que por su acceso secuencial a la información se caracterizan por tener tiempos de acceso elevados.



asociar un fichero con una determinada posición en un medio de almacenamiento masivo. En estas condiciones, la especificación del nombre del fichero en un comando permite al sistema operativo acceder al mismo en el medio de almacenamiento empleado.
El sistema operativo UNIX permite, en el caso de trabajar en un entorno multiusuario, definir varios "links" sobre un mismo fichero de manera que varios usuarios pueden compartir la información contenida en el mismo, lo cual es una faceta importante para la obtención de un máximo rendimiento de los medios de almacenamiento masivo.
Cuando un fichero es creado se genera un "link" hacia él. El fichero puede ser a continuación borrado, pero esto no significa que físicamente se eliminen los registros que lo componen, sino que tan solo se elimina el "link" que se dirige a los mismos en el instante de la petición de información; ello se consigue por medio del comando RM.



El UNIX es un sistema operativo que permite organizar la información en estructuras jerárquicas de tipo árbol.



El método de los "links" facilita el acceso a un mismo punto de la estructura a partir de distintos orígenes.

La generación de un nuevo "link" sobre un fichero ya existente se consigue haciendo uso del comando LN. Este comando, tal y como está estructurado, permite que un mismo fichero pueda ser accedido desde diferentes directorios; con lo cual, y si recordamos lo indicado anteriormente respecto a los "pathnames" y a los directorios de trabajo, evita que aquellos sean excesivamente largos al hacer que el fichero perteneciente a otro directorio se comporte de forma idéntica a otro perteneciente al actual directorio. Por medio de este comando también es posible establecer un "link" sobre un fichero del propio directorio desde este mismo, si bien, en este caso, el "link" ha de realizarse con un nombre de fichero distinto al que el fichero posee. Para entender plenamente la actuación del comando LN, hay que tener presente el hecho de que éste crea punteros hacia un fichero, no genera copias del mismo y, en este caso, toda la información sobre el status del fichero como podrían ser derechos de acceso, propietario del fichero o la información referente a la última vez que el fichero fue actualizado, permanece igual y únicamente el "pathname" es diferente, toda vez que la trayectoria por la que se accede al fichero es distinta.
Los diferentes "links" hacia un determinado fichero presentan los mismos derechos, ya que el sistema operativo UNIX es incapaz de hacer una distinción entre uno y otro. De la misma forma es posible eliminar uno de los "links" con un fichero y los demás continuarán operando de manera totalmente normal.

Symphony (2)

Comandos de servicio y comandos de entorno

En el capítulo anterior se inició el estudio de los distintos comandos de servicio del SYMPHONY; el presente concluye dicho análisis, el cual se concreta con los comandos de entorno que permiten al usuario explotar la hoja electrónica, el tratamiento de textos, el sistema gráfico, la base de datos y el entorno de comunicaciones.

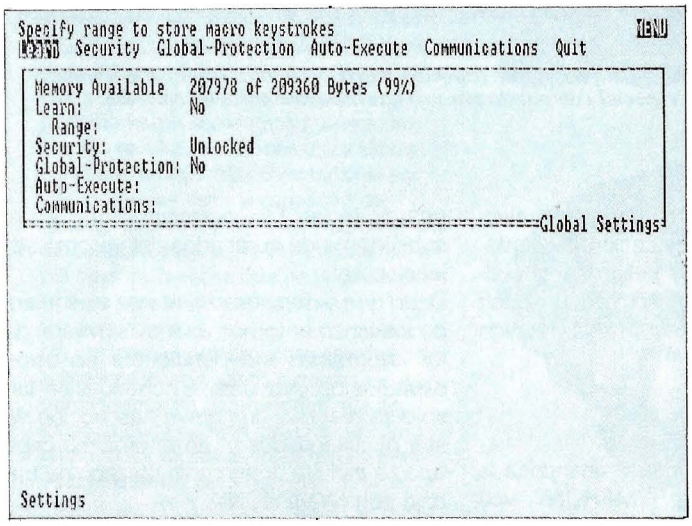
SISTEMA DE MENUS PARA COMANDOS DE SERVICIO (CONTINUACION)

En páginas anteriores se ha visto el funcionamiento del comando WINDOW con el que se podrían definir ventanas, además del comando FILE para la gestión de ficheros, de PRINT destinado a la impresión de informes y del comando CONFIGURATION. A continuación se describe el funcionamiento de los restantes comandos de este sistema de menús.

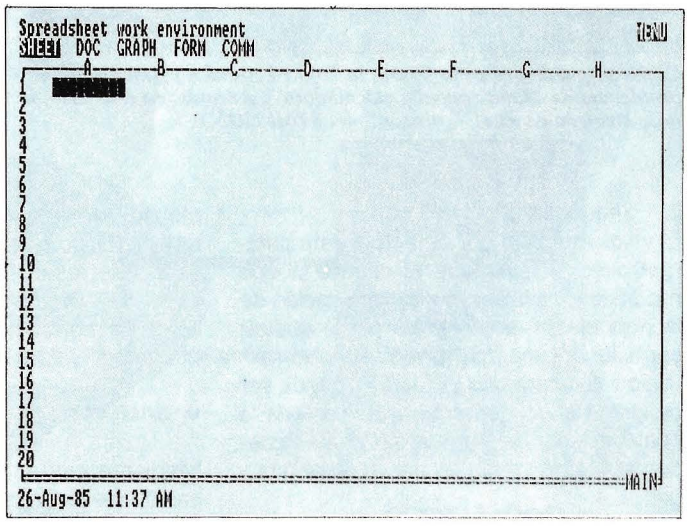
- **APLICACIONES** (APPLICATION)

SYMPHONY permite ejecutar programas desarrollados por el usuario. El comando de servicio APPLICATION posibilita la carga de programas en la memoria principal y su ejecución; analógicamente, también es posible descargar alguno de los programas desde la memoria principal almacenándolo a algún soporte externo.
Cuando el usuario selecciona la opción APPLICATION aparece en la parte superior de la pantalla un nuevo menú que ofrece las cinco opciones que se detallan a continuación:

1. carga (ATTACH)
Sirve para cargar un programa en la memoria principal; para ello, el programa solicitará el nombre del fichero en que se encuentra almacenado.
2. Descarga (DETACH)
Su misión es complementaria a la del comando anterior, es decir: al ser ejecutado descargará el programa desde la memoria principal grabándolo en la unidad de memoria auxiliar activa.
3. Ejecución (INVOKE)
Se puede utilizar para ejecutar un programa previamente cargado en la memoria principal. Tras ejecutar el comando INVOKE, el usuario debe especificar el nombre de los programas a ejecutar, debido a que en la memoria pueden estar cargados varios programas.
4. Borrado (CLEAR)
Mediante esta opción SYMPHONY borra todos los programas cargados en la memoria principal y libera el espacio para otras utilizaciones.
5. Salida (QUIT)
La última opción de este grupo sirve para abandonar el comando APPLICATION.



El comando de servicio SETTING se utiliza como subcomando en muchos entornos. En general, siempre permite visualizar y modificar los parámetros genéricos del entorno en que es invocado.



El menú principal del sistema de comandos de entorno presenta cinco posibilidades distintas que coinciden con los cinco entornos de trabajo del SYMPHONY.

- **CARACTERISTICAS** (SETTINGS)

Permite definir características globales sobre la hoja electrónica, tales como procedimientos de seguridad para evitar el borrado accidental de la información almacenada, activar o desactivar los procesos basados en macros, etc. Al ser activado este comando aparecerá en la pantalla información sobre la memoria principal disponible y sobre el estado de las opciones asociadas a SETTINGS; estas se detallan a continuación:

1. Apoyo (LEARN)
Mediante esta opción, el usuario puede escribir sus propias macros para, posteriormente, ejecutarlas. Al seleccionar LEARN el programa presentrará un nuevo menú de opciones para facilitar la creación de las macros.

SI o NO; en el primer caso permite que el usuario proteja una zona de la hoja electrónica.

4. Autoejecución (AUTO.EXECUTE)
Permite definir una macro que se ejecutará automáticamente al cargar la hoja electrónica.

5. Comunicaciones (COMUNICATIONS)
Sirve para especificar un fichero con la configuración de comunicaciones para que SYNPHONY lo utilice automáticamente.

6. Salida (QUIT)
La última opción de este entorno tiene como única misión abandonar el comando SETTINGS.

- **BORRADO** (NEW)

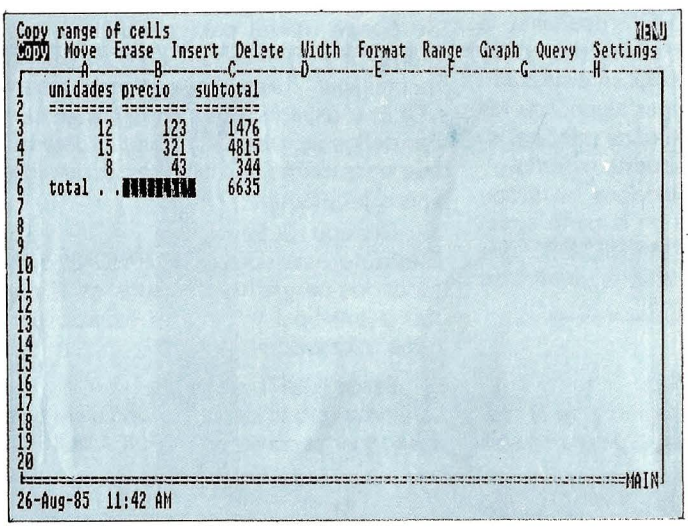
El comando NEW sirve para borrar completamente el contenido de la hoja electrónica. Dado que su ejecución involuntaria puede resultar muy peligrosa, al activarlo SYMPHONY solicita confirmación antes de borrar físicamente el contenido de la hoja electrónica.

- *SALIDA* (EXIT)

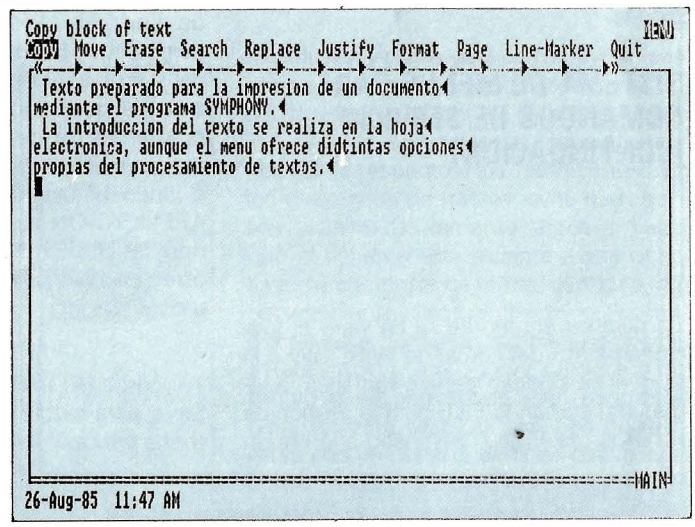
Mediante este comando se abandona la sesión de trabajo con SYMPHONY, volviendo el control del ordenador al sistema operativo. También ahora, antes de ejecutar el comando se solicita confirmación de que realmente el usuario desea ejecutarlo.

SISTEMA DE MENUS PARA COMANDOS EN ENTORNO

El segundo sistema de menús de SYMPHONY contiene en su estructura a todos los comandos utilizables desde cualquiera de sus cinco entornos. Las teclas ALT y F 10 , presionas simultáneamente, permiten visualizar el menú principal del sistema que aporta cinco opciones distintas: SHEET para la hoja electrónica, DOC para proceso de textos, GRAPH para gráficos de gestión, FORM para la base de datos y COMM para comunicaciones. Después de haber elegido alguna de estas opcio-



Como se puede ver en la figura, la hoja electrónica permite resolver problemas de "lápiz, papel y calculadora". Además, en este caso, la hoja electrónica es el "corazón" del SYMPHONY.



Para el proceso de textos SYMPHONY utiliza una hoja electrónica especial compuesta por muchas líneas de una sola columna.

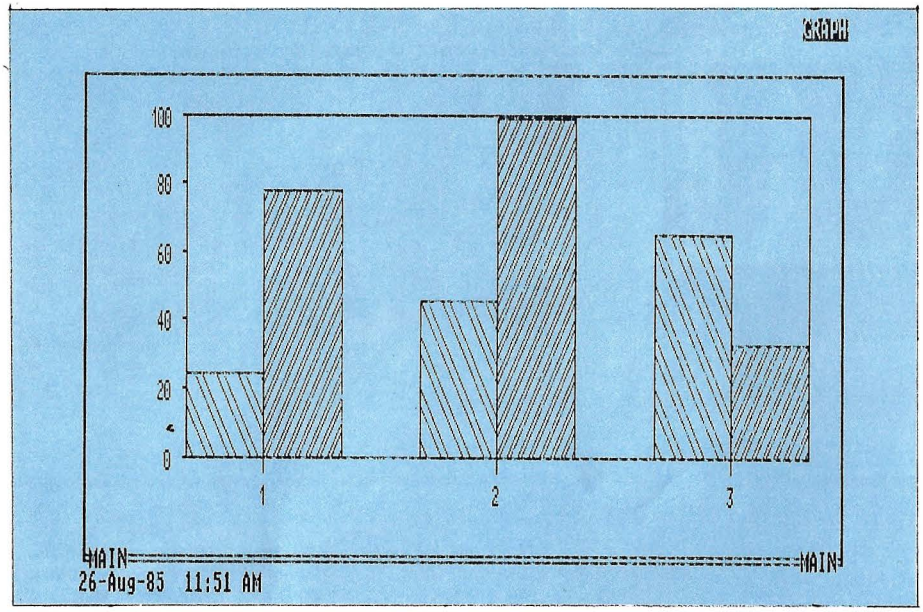
2. Seguridad (SECURITY)
El valor asociado por defecto a este parámetro es NO, aunque el usuario puede modificarlo; en tal caso la visualización de la hoja electrónica solo se podrá realizar conociendo una "palabra clave". La activación del comando SECURITY puede ser peligrosa dado que si el usuario olvida la "palabra secreta", jamás se podrá acceder a la información en ella almacenada.

3. Protección global (GLOBAL-PROTECTION)
Análogamente puede tomar dos valores:

nes, la tecla F9 permitirá entrar en el subsistema de comandos del entorno seleccionado.
Dado que el funcionamiento de cualquiera de los cinco entornos es muy similar al de los programas independientes en otros capítulos de esta obra, a continuación tan solo se realizará una breve descripción de sus peculiaridades y, en el próximo capítulo se incluirá un ejemplo práctico de trabajo con SYMPHONY.

- *Subsistema hoja electrónica* (SHEET)

El entorno principal de SYMPHONY es la



hoja electrónica. Además de permitir realizar las operaciones típicas de este tipo de programas, la hoja electrónica sirve como "centro de coordinación" para el resto de los entornos. La forma de acceso a la hoja electrónica consiste en situar el cursor sobre el comando SHEET del menú principal y pulsar ENTER; inmediatamente, la pantalla se formateará en matriz, asignando números a cada una de las filas y letras a cada una de las columnas. En la parte superior, aparecerán once comandos de entorno para permitir que el usuario gestione la hoja electrónica.
A partir de ese momento el usuario tiene dos posibilidades:

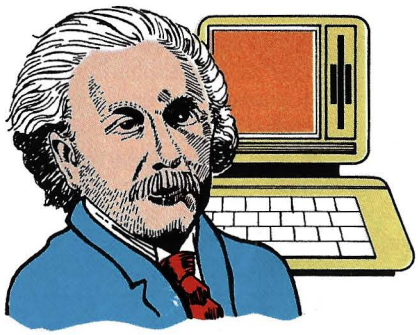
◄

También en el entorno gráfico la hoja electrónica juega un papel importante. En ella se debe introducir una visión parametrizada del diagrama a producir.

Inteligencia artificial

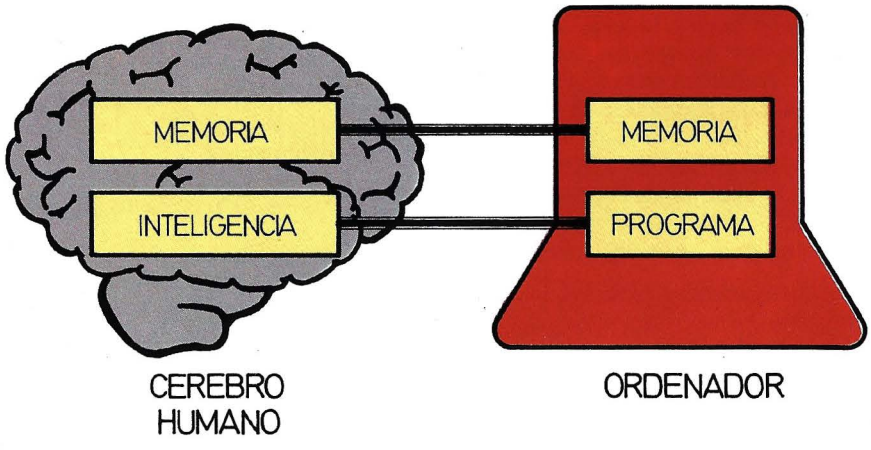
Papert Goldstein afirmó en 1977: "El problema fundamental de comprender que es la inteligencia no reside en identificar unas pocas técnicas muy potentes, sino más bien en cómo representar grandes cantidades de conocimiento de forma que pueda utilizarse con eficacia de un modo automático". En efecto esto es así. La inteligencia artificial es una ciencia a caballo entre la lógica teórica y la informática, que tiene como misión mecanizar los procesos mentales de inferencia mediante programas informáticos. Básicamente existen dos componentes fundamentales dentro del cerebro humano: por un lado la memoria y por otro un mecanismo, no muy bien conocido, que permite razonar a partir de la información contenida en la memoria. Con un ordenador se puede simular perfectamente la memoria humana, en cambio resulta mucho más difícil simular el componente que más caracteriza al ser inteligente, es decir la capacidad de razonar. Esta dificultad surge más de la falta de conocimiento sobre el funcionamiento del cerebro humano que de la dificultad técnica para programar dicho funcionamiento.
No obstante el razonamiento anterior, en la actualidad existen varios sistemas expertos mecanizados en ordenadores, que sin ninguna duda pueden catalogarse como auténticos programas inteligentes. Entre estos sistemas cabe descatar:

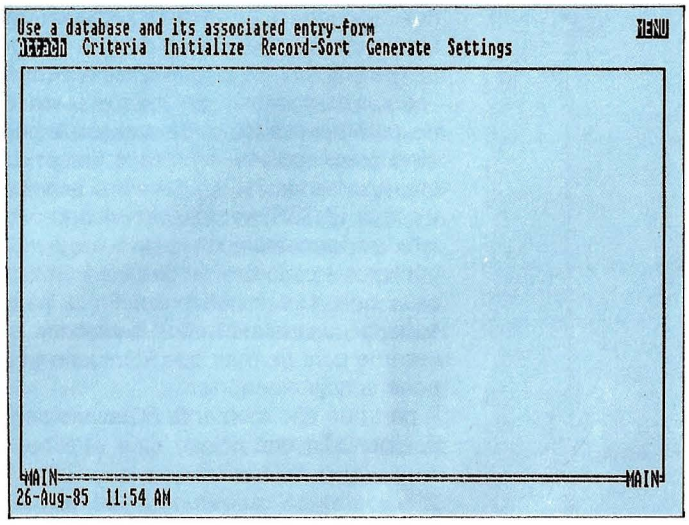
- MYCIN, sistema para el diagnóstico de infecciones producidas por bacterias.
- PROSPECTOR, sistema para la ayuda en la realización de prospecciones geológicas.
- INTERNIST, sistema general sobre medicina interna.



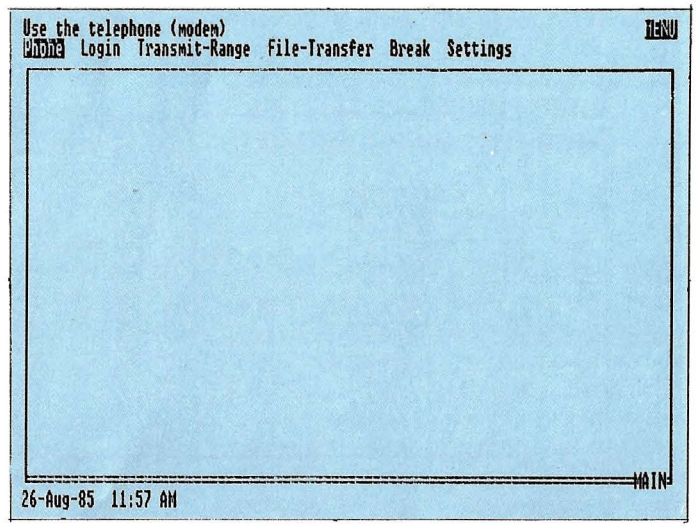
- CASNET, sistema para el diagnóstico de ciertas enfermedades oculares.
- DENDRAL, sistema para deducir la estructura de componentes químicos.
- MOLGEN, sistema para la realización de experimentos genéticos.

Como se puede apreciar por las materias tratadas, estos programas hacen una incursión en técnicas que hasta ahora habían estado reservadas al intelecto humano. No obstante la Inteligencia Artificial es una ciencia en estado poco avanzado y, probablemente, en pocos años se obtendrán resultados apasionantes.





Las posibilidades prácticas de trabajo con SYMPHONY se potencian con el entorno base de datos.



SYMPHONY además de los cuatro entornos de trabajo, dispone de otro adicional para establecer comunicaciones con otros ordenadores.

1. Realizar operaciones elementales sobre la hoja como mover el cursor a la celda deseada, introducir un dato literal, numérico o fórmula, asignar una función (siempre marcadas por el carácter), etc.

2. Ejecutar alguno de los comandos descritos en la parte superior de la pantalla que, de alguna manera, puedan asociarse a operaciones complejas sobre la hoja, como copiar, mover o insertar subconjuntos de celdas, insertar o borrar líneas o filas, asignar formatos fijos al contenido de determinadas celdas e incluso visualizar en forma gráfica el contenido de la hoja electrónica.

- *Subsistema proceso de textos* (DOC)

Al seleccionar el comando DOC, en la pantalla del ordenador aparacerán diez comandos en forma de menú, y el formato de la pantalla tomará el aspecto de hoja electrónica con ciertas modificaciones: Las líneas no aparecerán numeradas y tan solo existirá una única columna. Evidentemente, este aspecto resulta completamente adecuado para la introducción de textos: cada línea de la hoja electrónica será una línea del documento a producir. De nuevo el usuario tiene dos opciones:

1. Introducir directamente las palabras que formarán parte del documento.

2. Desencadenar la ejecución de algunos de los comandos situados en la parte superior de la pantalla que, entre otras cosas, permitirán copiar, mover o borrar porciones del texto, buscar, y en su caso modificar, determinadas frases o palabras, justificar las líneas, etc.

- *Subsistema gráficos de gestión* (GRAPH)

Después de haber seleccionado la opción GRAPH en el menú principal del sistema de comandos de entorno, la pantalla del ordenador adquirirá aspecto gráfico. En la parte superior se visualizarán cuatro comandos de gestión del entorno y en el resto de la pantalla el gráfico deseado, en el caso de que éste previamente haya sido definido.
La manipulación de un gráfico pasa idefectiblenente por la hoja electrónica. En ella se deben introducir los parámetros que lo caracterizan, por ejemplo: en la columna A los valores del eje X, y en la columna B los del eje Y. A continuación, utilizando los comandos GRAPH se pueden asociar los valores de la hoja electrónica al grafico deseado, así como el resto de características del gráfico, para que inmediatamente podamos visualizar el diagrama en la pantalla.

- *Subsistema base de datos* (FORM)

De nuevo hay que repetir algo que ya viene siendo una constante en todos los entornos: la utilización de la base de datos debe venir precedida por su definición mediante la hoja electrónica. De esta forma se podrán marcar las características generales de una nueva base de datos, introducir y revisar registros de datos, clasificar dichos registros, marcar criterios de selección para obtener los registros que verifican determinadas condiciones, etc.
Al seleccionar la opción FORM, en la parte suerior de la pantalla se visualizarán todos los comandos aportados por SYMPHONY en este entorno.

- *Subsistema comunicaciones* (COMM)

El quinto y último entorno de SYMPHONY tiene como misión permitir las comunicaciones entre dos ordenadores personales, o incluso entre un ordenador personal y un gran ordenador. Para entrar en este entorno basta con pulsar la tecla RETURN después de haber situado al cursor sobre el comando COMM del menú principal. Inmediatamente aparecerán en la parte superior de la pantalla los seis comandos ofrecidos al usuario dentro de este entorno:

1. PHONE, para establecer comunicaciones telefóncas vía modem.

2. LOGIN, para establecer comunicaciones con un ordenador remoto.

3. TRANSMIT-RANGE, permite definir el rango de la hoja electrónica implicada en la transmisión.

4. FILE-TRANSFER, sirve para enviar o recibir los datos.

5. BREAK, "Interrumpe" la comunicación.

6. SETTINGS, permite visualizar y modificar las características generales de este entorno.

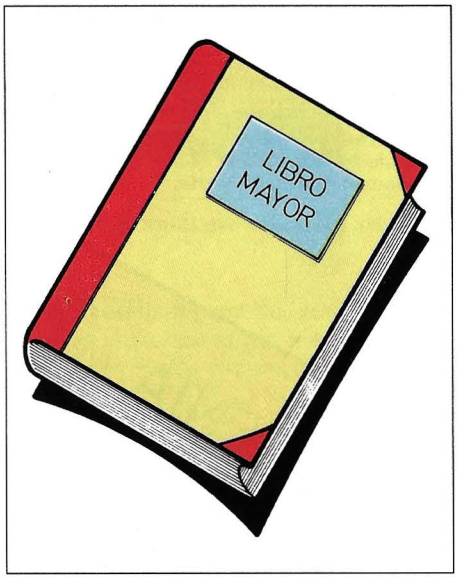
El contable en casa

Un programa de gestión, paso a paso

En el presente capítulo se avanza un peldaño más en el terreno práctico, aplicando los comandos BASIC estudiados hasta el momento. En esta ocasión, el programa a confeccionar es catalogable como de gestión.
Para poder construirlo de forma que tenga una cierta compatibilidad con cualquier tipo de ordenador personal, no se van a utilizar comandos "extraños", sólo disponibles en muy pocos dialectos BASIC. Al efecto, se utilizarán los comandos más corrientes en los intérpretes de BASIC de mayor difusión. La construcción del programa se complementará con explicaciones exhaustivas sobre la misión de cada comando y subrutina utilizadas.

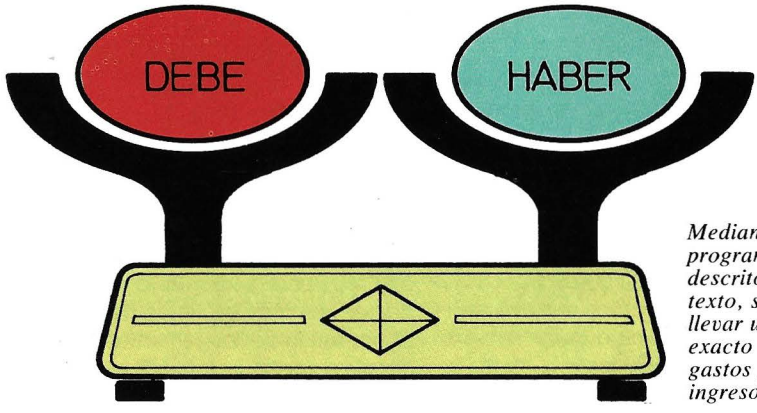
EL CONTABLE EN CASA

El programa que se va a desarrollar a continuación, es un ejemplo de un sencillo banco de datos, destinado a almacenar información relativa a los pequeños o grandes gatos que se realizan en todo entorno familiar, o incluso en el de una pequeña empresa. Se trata de simular un "libro mayor", en el que se irán anotando cuidadosamente todos los gastos y los ingresos que se realicen en la familia o en la empresa, consignándolos en el lugar adecuado (en el "debe" o en el "haber"). El programa permite obtener información relativa al saldo actual en cada momento. Sobre esta base es posible añadir cualquier tipo de subrutina al programa, para calcular o ejecutar la acción o acciones particulares que necesite cada usuario, personalizando así su programa.



El objetivo del presente capítulo es simular mediante un programa en BASIC la contabilidad de un libro Mayor.

Para no hacer pesada la codificación, sólo se va a añadir una utilidad "extra", dejando a la atención del lector la adición de las nuevas utilidades que desee; ello resultará muy sencillo ya que bastará con incluir nuevas subrutinas que se pueden llamar por medio de un menú. Este será precisamente el modo en que estará organizado el programa que se va a codificar a lo largo de los próximos párrafos. La utilidad "extra" que se va a incluir, permitirá llevar en paralelo con el saldo real, el saldo de nuestra cuenta corriente o cartilla de ahorros; como es sabido, los bancos tardan un tiempo en confirmar las operaciones que realizamos, y en consecuencia, hay momentos en los que no se sabe con precisión la cantidad disponible en la cuenta corriente. En cada apunte realizado se podrá indicar si éste ha sido o no confirmado por el banco: así mismo, también se podrá modificar el apunte una vez que el banco nos de la confirmación.
Una vez definido por encima el objetivo, es hora de poner los pies en tierra y comenzar a dar especificaciones prácticas del programa.
El principal elemento del programa será el fichero en el que se van a almacenar todos los datos que se introduzcan. Utilizar un fichero tipo aleatorio ofrece una serie



Mediante el programa descrito en el texto, se puede llevar un balance exacto de los gastos y los ingresos.

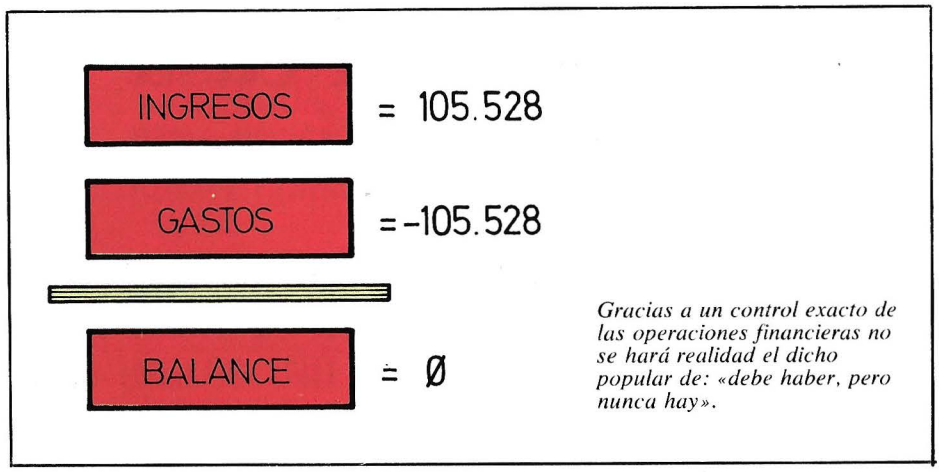
de ventajas que serían muy largas de enumerar aquí, pero, a la vez, su manejo exige un mayor cuidado, hasta el punto de que la codificación de un programa que maneja ficheros aleatorios puede resultar muy complicada. Por esta razón, y para simplificar la labor, se utilizará un fichero secuencial que proporciona las características suficientes para nuestro propósito y que, además, será posible grabarlo en disco flexible o en casete; esto último supone una ventaja para aquellos usuarios que no dispongan aún de una unidad de disco.

El fichero de datos que necesitamos, debe ser fácilmente manejado por el ordenador. La solución más sencilla es utilizar una matriz alfanumérica como fichero interno, ya que el empleo de subíndices facilita su manipulación. Como novedad, no se utilizarán esta vez matrices multidimensionales, sino que se empleará una única matriz alfanumérica unidimensional, en la cual se almacenarán por secciones cada uno de los datos introducidos. Para ello hay que dividir imaginariamente la matriz en sectores y campos, de tal forma que después se pueda localizar fácilmente la información.

Adoptando la terminología contable, se llamará asiento a cada sector de la matriz; a su vez, cada asiento se dividirá en campos de tamaño distinto para cada uno de los datos. Así, por ejemplo, una posible división de los campos podría ser la siguiente:

- Un campo de 1 carácter para la confirmación por el banco.
- Un campo de 6 caracteres para la fecha.
- Un campo de 20 caracteres para la descripción del apunte realizado.
- Un campo de 1 carácter para el signo del cargo (debe/haber).
- Y, finalmente, un campo de 8 caracteres para el importe.

Todos estos campos se han distribuido en su longitud suponiendo que se dispone de una pantalla de 24 filas y 40 columnas. Con ello, cada sector o asiento del fichero tendrá 40 espacios de longitud, lo que permitirá imprimirlo directamente en la pantalla. Para definir el tamaño de la matriz que necesitamos sólo queda por establecer el número de asientos que se quieren utilizar. Si este número lo fijamos en 500, ya se puede dimensionar la matriz y con ello crear las primeras líneas del programa:



Gracias a un control exacto de las operaciones financieras no se hará realidad el dicho popular de: «debe haber, pero nunca hay».

Por medio de una opción incorporada al programa, se puede conocer con celeridad y exactitud el estado actual de nuestra cuenta bancaria.


```
10 DIM DATO$(501*40),M$(40),FECHA(6)
   CON$(20),IMP$(8),CARGO$(1),VALE$(1)
20 DATO$=" ":DATO$(501*40)=" ":DATO$
   (2)=DATO$
30 LET A=1:LET PA=1:LET SR=0:LET
   SC=0:LET P=
140 OPEN #1,4,0,"K"
```

Las líneas 10, 20 y 30 se encargan de dimensionar e inicializar las diferentes matrices variables que serán necesarias a lo largo del programa. Así, DATO$() constituye el ya mencionado fichero interno de datos que se ha dimensionado con el valor 501*40, para poder incluir en su último sector la marca de fin de datos.

El cometido de las restantes variables se detalla a continuación. M$ es una variable de utilidad general adecuada para la obtención de datos del teclado, impresión de mensajes en pantalla, etc. FECHA() se utilizará para almacenar temporalmente la fecha del asiento antes de introducirla definitivamente en el fichero, por si hay algún error y es preciso modificarla. Con el mismo fin se utilizarán las siguientes matrices: CON$(), IMP$(), CARGO$() y VALE$(), para los campos: concepto, importe, cargo (debe="D"/haber="H") y confirmación del banco, respectivamente.

La línea 20 se encarga de inicializar el fichero de datos con caracteres blancos (" "); no obstante, si ésta no funcionara en su ordenador puede sustituirla por un bucle FOR/NEXT como el siguiente:

```
20 FOR I=1 TO 501*40:DATO$(I)=" ":NEXT I
```

Las restantes variables numéricas que son inicializadas en la línea 30 son las siguientes: A para el número de asiento (inicialmente=); PA, para el primer nú-

mero de asiento de una página, ya que al no ser posible visualizar a la vez los 500 asientos del fichero éstos se consultarán por páginas; SR para el saldo real; SC para el saldo de la cuenta bancaria y P como puntero de la línea de la página visualizada, para introducir la confirmación por el banco del asiento señalado.

Por último, la línea 40 se encarga de abrir un canal de entrada para el teclado, por el que se introducirán todos los datos. Respecto a esta línea hay que decir que su formato varía mucho de unas máquinas a otras, por lo que será preciso acudir al manual BASIC de cada máquina. No obstante, si en algunos casos no fuera posible utilizar esta técnica, se podrá recurrir, haciendo las modificaciones pertinentes, al sufrido INPUT, o incluso al comando INKEY$ o equivalentes.

Con estas líneas ya está inicializado el programa, por lo que se pueden comenzar a codificar rutinas que cumplan las diferentes tareas a realizar. A continuación y tras una simple pantalla de presentación con el nombre del programa se puede ya empezar a construir el menú principal que dará entrada a las diferentes subrutinas que componen el programa.

FECHA	CONCEPTO	CARGO	IMPORTE
Ø1Ø285	> SALDO INICIAL	H	30 000
* Ø1Ø285	INGRESO CUENTA	H	16 000
Ø1Ø285	GASTOS VARIOS	D	5 200
Ø2Ø385	COMIDA	D	1 000
	FIN DE DATOS		

PAGINA	1	ASIENTO	1
SALDO TOTAL	39 800		
SALDO CUENTA	23 800		
DIFERENCIA	16 000		

Página de asientos que se obtiene con el programa al consultar un fichero de datos.

1: Consulta del archivo, para ver asientos introducidos con anterioridad.
2: Actualización o entrada de nuevos datos.
3: Cargar del disco o casete un fichero de datos.
4: Guardar el fichero en disco o casete.

Estas son las acciones básicas necesarias

EL MENU DE OPCIONES

El cartel de entrada visualizará simplemente el nombre del programa, previo borrado de la pantalla. Tras un pequeño intervalo de retardo, ésta presentará directamente el menú principal. Naturalmente, el lector puede incluir aquí cualquier cosa que le dicte su espíritu artístico; un simple cartel de entrada podría ser el siguiente:

```
60 CLS:POSITION 11,12:PRINT "GASTOS
   FAMILIARES"
70 FOR I=1 TO 500:NEXT I
```

El comando POSITION seguido del PRINT puede ser sustituido por un PRINT AT u otro comando equivalente. Para modificar el menú principal, será necesario determinar primero qué acciones se desean tomar a partir de él. Así, por ejemplo, éstas podrían ser las siguientes:

El objetivo del presente capítulo es confeccionar un programa BASIC que ayude a llevar al día la contabilidad doméstica.

```
540 CLS:PRINT "    ACTUALIZAR FICHERO"
558 IF A+1<501 THEN 590
560 POSITION 13,12:PRINT "FICHERO LLENO";:
    FOR I=1 TO 1000:NEXT I:RETURN
590 POSITION 1,5:PRINT " ASIENTO.....":
    POSITION 18,5:PRINT A
600 POSITION 1,7:PRINT " FECHA
    (DOMMAA)."
610 POSITION 1,9:PRINT " CONCEPTO....."
620 POSITION 1,11:PRINT " IMPORTE....."
630 POSITION 1,13:PRINT " CARGO (D/H)...."
640 POSITION 1,15:PRINT " CONFIRMA-
    CION..."
650 POSITION 17,7:PRINT ">      ":POSI-
    TION 18,7
660 FOR I=1 TO
670 GET #1,M:M$=CHR$(M)
690 IF M$="E" AND I=1 THEN RETURN
700 IF M$="" THEN 670
710 IF M$<"0" OR M$>"9" THEN 670
720 PRINT M$;:FECHA(I)=VAL(M$):NEXT I
770 POSITION 17,7:PRINT " ":POSITION 17,9:
    PRINT ">":POSITION 18,9
790 CON$="":FOR I=1 TO 20
800 GET #1,M:M$=CHR"(M)
820 IF M$="" THEN 800
830 IF M=155 THEN 870
840 PRINT M$;:CON$(I)=M$:NEXT I
870 POSITION 17,9:PRINT " ":POSITION 17,
    11:PRINT ">":POSITION 18,11
880 IMP$="":FOR I=1 TO 8
900 GET #1,M:M$=CHR$(M)
920 IF M=155 THEN I=9:GOTO 960
930 IF M$<"0" OR M$>"9" THEN 900
940 PRINT M$;:IMP$(I)=M$
960 NEXT I
980 IF VAL(IMP$)=0 THEN 870
990 POSITION 17,11:PRINT " :POSITION 17,
    13: PRNT ">":POSITION 18,13
1000 GET #1,M:M$=CHR$(M)
1020 IF M$<>"D" AND M$<>"H" THEN 1000
1030 PRINT M$;:CARGO$=M$
1050 POSITION 17,13:PRINT " ":POSITION 17,
    15:PRINT ">":POSITION 18,15
1060 GET #1,M:M$=CHR$(M)
1080 IF M$<>"S" AND M$<>"N" THEN 1060
1080 PRINT M$;:VALE$=M$
1110 POSITION 17,15:PRINT " ":POSITION 8,
    20:PRINT "DATOS CORRECTOS (S/N) ?"
1120 GET #1,M:M$=CHR$(M)
1140 IF M$<>"S" AND M$<>"N" THEN 1120
1150 POSITION 0,20:PRINT "                "
1160 IF M$="S" THEN 1180
1170 GOTO 650
```

PROGRAMA 1: subrutina de entrada de datos.

y suficientes para el objetivo que nos hemos marcado; si bien, el lector podría incluir aquí sus propias opciones con plena sencillez.

La rutina que imprime el menú en la pantalla, es la que sigue:

```
90 CLS:POSITION 16,0:PRINT "OPCIONES"
110 POSITION 10,6:PRINT "1:ACTUALIZAR
    FICHERO"
120 POSITION 10,8:PRINT "2:CONSULTAR
    FICHERO"
130 POSITION 10,10:PRINT "3:CARGAR
    FICHERO"
140 POSITION 10,12:PRINT "4:GUARDAR
    FICHERO"
```

Su ejecución dará lugar a la siguiente pantalla:

```
OPCIONES

1:ACTUALIZAR FICHERO
2:CONSULTAR FICHERO
3:CARGAR FICHERO
4:GUARDAR FICHERO
■
```

Ahora sólo queda por realizar la zona adecuada para captar la opción elegida por el usuario. Esto puede hacerse mediante el comando GET#1,M que espera a que se pulse una tecla y guarda su código en la variable M, a través del canal de entrada #1 previamente abierto para el teclado. De esta forma no es necesario pulsar la tecla RETURN como ocurriría con un INPUT. Después habrá que examinar si la opción elegida es válida y, si es así, enviar el flujo del programa hacia la subrutina correspondiente, lo que puede lograrse mediante el comando ON M GOSUB...
El listado de esta zona es la siguiente:

```
150 GET #1,M:M$=CHR$(M)
160 IF M$<"1" OR M$>"4" THEN 150
170 M=VAL(M$):ON M GOSUB 540,190,1380,
    1310
180 GOTO 90
```

Al recurrir a esta técnica es ineludible utilizar el comando CHR$, ya que el código almacenado en M debe transformarse en su carácter correspondiente para poder examinarlo. Así mismo, mediante VAL se transforma ese carácter en su equivalente numérico, el cual es necesario para el correcto funcionamiento de ON M GOSUB. En los números de línea a los que bifurca este último comando mencionado se situarán las correspondientes subrutinas, que no existen todavía, por lo que si se trata de ejecutar el programa se obtendrá un mensaje de error de línea no existente. Como el orden de los factores no altera el producto, empecemos ya a codificar la subrutina a la que se dirigirá el programa al pulsar el usuario la tecla "2"; ésta corresponde a la zona de consulta del fichero. Lo primero que hay que hacer es un borrado de la pantalla con CLS para eliminar el menú de opciones y, a continuación, formatear la pantalla con una cabecera que incluya los diferentes campos que componen cada asiento, para representar estos posteriormente. También hay que representar otras informaciones adicionales, como el número de asiento que señala el cursor, la página, el saldo...

```
190 CLS:PRINT " FECHA        CONCEPTO
    CARGO IMPORTE"
200 POSITION 2,20:PRINT "PAGINA..... "
210 POSITION 17,20:PRINT INT(PA/19)+1
220 POSITION 22,20:PRINT "ASIENTO.... "
230 POSITION 34,20:PRINT PA+P-1
240 POSITION 2,21:PRINT "SALDO TOTAL...
    ":POSITION 17,21:PRINT SR
250 POSITION 2,22:PRINT "SALDO CUENTA..
    ":POSITION 17,22:PRINT SC
260 POSITION 2,23:PRINT "DIFERENCIA....
    ";:POSITION 17,23:PRINT SR-SC;
```

Con el comando POSITION (o con el PRINT AT en su defecto) se sitúan en su lugar correspodiente los diferentes mensajes, en lo que se obtendrá la siguiente pantalla:

```
FECHA      CONCEPTO      CARGO IMPORTE

PAGINA ..... 1      ASIENTO ... 1
SALDO TOTAL ... 0
SALDO CUENTA .. 0
DIFERENCIA ..... 0
```

Al cambiar la página de la forma que se verá más adelante, cada mensaje borra el valor antiguo, el cual se representa a continuación de éste, y después se imprime el nuevo valor. Sólo queda ya imprimir los sectores del fichero que quepan en la página y que, debido a los mensajes anteriores, sólo serán 18 asientos los que se podrán visualizar simultáneamente:

```
270 POSITION 0,1
280 FOR I=PA TO PA+17
290 PRINT MID$(DATO$,I*40-39,I*40-1)
300 NEXT I
310 POSITION 7,P:PRINT ">"
```

La línea 310 imprime el puntero de modificación por confirmación del banco en el primer asiento de esa página; éste podrá ser desplazado por medio de las siguientes líneas del programa:

```
ACTUALIZAR FICHERO

ASIENTO          1
FECHA (DDMMAA)   Ø1 Ø2 85
CONCEPTO         SALDO INICIAL
IMPORTE          30 000
CARGO (D/H)      H
CONFIRMACION     S

DATOS CORRECTOS (S/N)?
```

La entrada de datos se realiza con el formato esbozado en este gráfico.

Un simple ordenador doméstico instruido por el programa cuya confección se acomete en este capítulo, servirá para nuestro objetivo: automatizar la contabilidad doméstica.

```
320 GET #1,M:M$=CHR$(M)
330 IF M<>45 AND M<>61 AND M<>43
AND M<>42 AND M<>27 THEN 320
340 GOTO (M=45)*350+(M=61)*380+
    (M=42)*410+(M=27)
*530+(M=43)*470
350 IF PA=1 THEN 320
360 PA=PA-18:P=1:GOTO 200
380 IF PA+18>A THEN 320
390 PA=PA+18:P=1:GOTO 200
410 POSITION 7,P:PRINT " ":P=P+1:IF P>18
    THEN P=1
430 IF PA+P-1>=A THEN P=1
440 POSITION 7,P:PRINT ">":GOTO 200
470 VAR=(PA+P-1)*40-40:IF DATO$
    (VAR+1)="      THEN 320
480 POSITION 0,P:PRINT " "
490 DATO$(VAR+1)=" "
500 IF DATO$(VAR+30)="D" THEN
M$(1)="-":M$(2)=STR$(VAL(MID$(DATO$,
    VAR+32,VAR+39))):SC=SC+VAL(M$):
    GOTO 200
510 IF DATO$(VAR+30)="H" THEN
M$(1)="+":
M$(2)=STR$(VAL(MID$(DATO$,VAR+32,
    VAR+39))):
SC=SC+VAL(M$)
520 GOTO 200
530 RETURN
```

Aunque el funcionamiento de esta zona del programa puede parecer muy complicada a primera vista, ello no es así. Primero (líneas 320 a 340) se espera a que se pulse una tecla. El código obtenido variará mucho de unas máquinas a otras, por lo que es preciso consultar cada manual específico. Una posible correspondencia de códigos con las teclas pulsadas, puede ser la siguiente:

45: cursor hacia arriba
61: cursor hacia abajo
43: cursor a la izquierda
42: cursor a la derecha
27: tecla ESCAPE

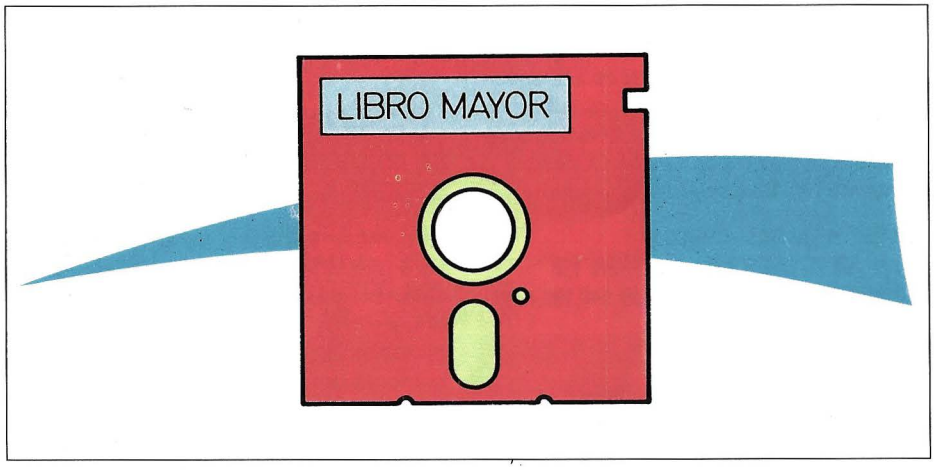
Y esta es precisamente la utilizada en este programa. Así, al pulsar cada tecla se bifurcará con la sentencia de la línea 340 hacia la zona correspondiente, según la acción que se desee realizar. Estas acciones son las siguientes: con la tecla de movimiento del cursor hacia abajo se pasa a la página siguiente del fichero; con la del cursor hacia arriba, a la página anterior; con la del cursor a la derecha se desplaza hacia abajo el puntero de modificación (representado por el signo: ">"), de forma que al llegar el último asiento de esa página vuelve al primero de nuevo; y con la tecla del cursor hacia la izquierda se valida un asiento que no hubiera sido aún confirmado por el banco.

Al introducir los datos, los asientos no confirmados por el banco se identificarán por medio de un carácter asterisco (*) en su primer campo. De esta forma, en las líneas 470 a la 510, se busca si existe este asterisco en el asiento señalado; si es así se procede a su borrado y a calcular el nuevo saldo de la cuenta contabilizando ya el importe de ese asiento. Tras ello se vuelve a la línea 200 que imprime de nuevo estos valores actualizados. Pero, ¿cómo se sale de esta subrutina? Sencillamente, mediante la pulsación de la tecla de ESCAPE, volviendo de inmediato el control al menú de opciones.

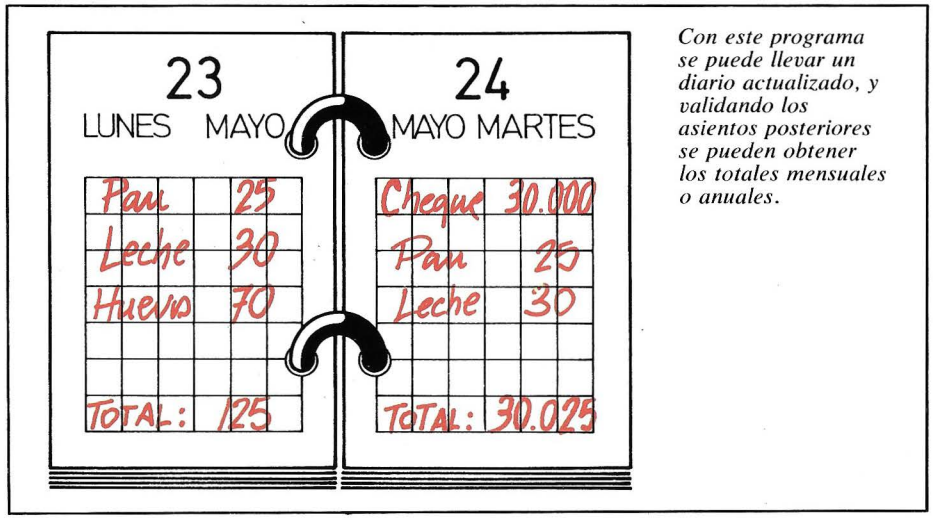
ENTRADA DE DATOS

La siguiente subrutina que se va a modificar es la de entrada de datos (ver listado adjunto). Esta no ofrece ninguna particularidad especial, por lo que sólo se hará un breve comentario de su funcionamiento.

La entrada de datos se hace a través del teclado y de acuerdo al procedimiento explicado anteriormente. Para ello se imprimen unos renglones que indican el concepto a introducir en cada momento, lo cual se indica mediante el símbolo: ">". Así para introducir la fecha, por ejemplo, se recurre a un bucle que se repite 6 veces, una para cada dígito. Los seis dígi-



El libro Mayor "informatizado" se hace más manejable y permite consultas más rápidas y eficaces.



Con este programa se puede llevar un diario actualizado, y validando los asientos posteriores se pueden obtener los totales mensuales o anuales.

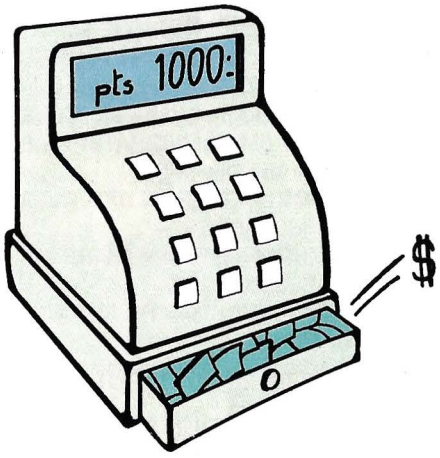
tos habrán de introducirse por tanto en la forma indicada: primero el día, seguido del mes y de las dos últimas cifras del año. Por ejemplo, el día 1 de abril de 1986, se introducirá con las siguientes cifras: 010486, no siendo necesario pulsar la tecla RETURN al final, pues automáticamente se pasa al campo siguiente. La longitud de los campos queda así limitada y al llegar a su final se pasa al siguiente campo.

Una vez introducidos todos los campos se pregunta si los datos son correctos, para en caso de detectar algún error poder corregirlo empezando de nuevo con la introducción de los datos. Estos datos se guardan temporalmente en las matrices respectivas como se explicó al principio.

Si se dan por correctos, los datos serán transferidos al fichero formado por la matriz DATO$, esta función la realiza la siguiente zona del programa:

```
1180 VAR=A*40-40:A=A+1:FOR I=1
TO 40:DATO$(VAR+I)="
":NEXT I
1200 IF VALE$(1)="N" THEN DATO$
    (VAR+1)="*"
1210 FOR I=1 TO 6:DATO$(VAR+I+1)=STA$
    (FECHA(I)):NEXT I
1220 MID$(DATO$,VAR+9,VAR+28)=CON$
1230 LON=LEN(IMP$):MID$(DATO$,
    VAR+40-LON,VAR+39)-IMP$
1250 IF CARGO$(1)="D" THEN
    DATO$(VAR+30)=CARGO$:M$(1)="-"
1260 IF CARGO$(1)="H" THEN
    DATO$(VAR 30)=CARGO$:M$(1)="+"
1270 DATO$((A-1)*40+13)="FIN DE
    DATOS":DATO$(501*40)=" "
1280 MID$(M$,2,2+LON)=IMP$:M=VAL (M$):
    SR=+M
1290 IF VALE$(1)="S" THEN SC=SC+M
1300 GOTO 540
```

Una vez introducidos los datos en la matriz DATO$, se empezarán a pedir datos para un nuevo asiento; éste es el momento, si se desea, de abandonar la subrutina para volver al menú de opciones. Al efecto se pulsará la tecla ESCAPE, cuya misión se detecta en la línea 690 en la que se ha indicado entre comillas el carácter "E"; este corresponderá con el carácter entregado por esa tecla en cada caso concreto, como ya se ha mencionado. Dicho carácter puede ser fácilmente sustituido para que abandone esta subrutina al pulsar cualquier otra tecla, como por ejemplo la "A", sin más que modificar la referida línea.



La cantidad en metálico de que se dispone en casa se puede obtener como la diferencia entre el saldo total menos el de la cuenta.

Al recibir la confirmación de una operación por parte del banco, se puede introducir esta modificación en nuestro fichero de datos para actualizar el saldo de la cuenta.

UN ARCHIVO PERMANENTE

A estas alturas tan sólo restan por codificar las subrutinas de carga y grabación del fichero de asiento. Estas funciones se pueden realizar abriendo sendos canales de entrada o de salida, según se trate de almacenamiento o de carga, respectivamente.

Para ello se utilizará el comando OPEN (nos remitimos a lo expuesto anteriormente al abrir el canal de entrada para el teclado). Las subrutinas de carga y almacenamiento sólo se diferenciarán en que en una se utilizará el comando PRINT# para guardar el fichero en el disco a casete y en la otra se utilizará el comando INPUT# para cargarlo en la memoria del ordenador. Ambos comandos deben trabajar con variables, por lo que se grabarán o se cargarán cada uno de los asientos por separado; ello se hará empleando la matriz M$ de utilidad general. Previamente, y para saber el número de asientos a cargar o grabar, se cargará o grabará la variable A que lleva la cuenta del número de asientos introducidos. Asimismo, se deben grabar los valores contenidos en SR y SC, correspondientes a los saldos que arrojan los asientos existentes.

Las subrutinas de carga y almacenamiento son las siguientes:

```
1310 CLS:PRINT "        GARDAR FICHERO"
1320 GOSUB 1440:OPEN #2,8,0,M$
1330 PRINT #2;A:PRINT #2,SR:PRINT #2:SC
1340 FOR I=0 TO A:M$=MID$
    (DATO$,I*40+1,I*40+40)
1360 PRINT #2;M$:NEXT I: CLOSE #2:RETURN
1380 CLS:PRINT "        CARGAR FICHERO"
1390 GOSUB 1440:OPEN #3,4,0,M$
1400 INPUT #3;A:INPUT #3;SR:INPUT #3;SC
1410 FOR I=0 TO A:INPUT
    #3;M$:MID$(DATO$,I* 0+1 I*  +40)=M$
1430 NEXT I:CLOSE #3:RETURN
1440 POSITION 2,12:PRINT "DISPOSITIVO:
    FICHERO ";:INPUT
    M$:RETURN
```

La subrutina de la línea 1440 es utilizada por las dos anteriores para obtener el nombre del fichero sobre el que se quieren grabar o cargar los datos; probablemente, el usuario tendrá que introducir modificaciones en ella para adaptarla al formato del comando OPEN de que disponga su intérprete BASIC.

```
 10 DIM DATO$(501*40),M$(40),FECHA(6),
    CON$(20), IMP$'3),CARGO$'1),VALE$'1)
 20 DATO$= ":DATO$(501*40)=" ":DATO$
    (2)=DATO$
 30 LET A=1:LET PA=1:LET SR=0:LET
    SC=0:LET P=1
 40 OPEN #1,4,0,"K"
 60 CLS:POSITION 11,12:PRINT "GASTOS
    FAMILIARES"
 70 FOR I=1 TO 500:NEXT I
 90 CLS:POSITION 16,0:PRINT "OPCIONES"
110 POSITION 10,6:PRINT "1:ACTUALIZAR
    FICHERO"
120 POSITION 10,8:PRINT "2:CONSULTAR
    FICHERO"
130 POSITION 10,10:PRINT "3:CARGAR FI-
    CHERO"
140 POSITION 10,12:PRINT "4:GUARDAR FI-
    CHERO"
150 GET #1,M:M$=CHR$(M)
160 IF M$<"1" OR M$> 4" THEN 150
170 M=VAL(M$):ON M GOSUB 540,190,1380,
    1310
180 GOTO 90
190 CLS:PRINT " FECHA CONCEPTO
    CARGO IMPORTE"
200 POSITION 2,20:PRINT "PAGINA..... "
210 POSITION 17,20:PRINT INT(PA/19)+1
220 POSITION 22,20:PRINT "ASIENTO.... "
230 POSITION 34,20:PRINT PA+P-1
240 POSITION 2,21:PRINT "SALDO TOTAL...
    ":POSITION 17,21:PRINT SR
250 POSITION 2,22:PRINT "SALDO CUENTA..
    ".POSITION 17,22:PRINT SC
260 POSITION 2,23:PRINT "DIFERENCIA....
    ;:POSITION 17,23:PRINT SR-SC;
270 POSITION 0,1
280 FOR I=PA TO PA+17
290 PRINT MID$(DATO$,I*40-39,I*40-1)
300 NEXT I
310 POSITION 7,P:PRINT ">"
320 GET #1,M:M$=CHR$(M)
330 IF M<>45 AND M<>61 AND M<>46
    AND M<>42 AND M<>27 THEN 320
340 GOTO (M=45)*350+(M=51)*380+(M=
    42)*410+(M=27)*530+(M=49)*470
350 IF PA=1 THEN 320
360 PA=PA-18:P=1:GOTO 200
380 IF PA+18>A THEN 320
390 PA=PA+18:P=1:GOTO 200
410 POSITION 7,P:PRINT " ":P=P+1:IF P>18
    THEN P=1
430 IF PA+P-1>=A THEN P=1
440 POSITION 7,P:PRINT ">":GOTO 200
470 VAR=(PA+P-1)*40-40:IF
    DATO$(VAR+1)=" "THEN 320
480 POSITION 0,P:PRINT"
490 DATO$(VAR+1)=" "
500 IF DATO$(VAR+30)="D" THEN
    M$(1)="-":M$(2)=STR$(VAL(MID$
    (DATO$,VAR+32,VAR+39))):SC=SC+VAL
    (M$):GOTO 200
510 IF DATO$(VAR+30)="H" THENM$(1)=
    "+":M$(2)=STR$(VAL(MID$(DATO$,VAR
    +32,VAR+33))):SC=SC+VAL(M$)
520 GOTO 200
530 RETURN
540 CLS: PRINT " ACTUALIZAR FICHERO"
550 IF A+1<501 THEN 590
560 POSITION 13,12:PRINT "FICHERO LLE-
    NO";:FOR I=1 TO 1000:NEXT I:RETURN
590 POSITION 1,5:PRINT " ASIENTO.....":
    POSITION 18,5: PRINT A
600 POSITION 1,7:PRINT " FECHA (DDMMAA."
610 POSITION 1,9:PRINT " CONCEPTO....."
620 POSITION 1,11:PRINT " IMPORTE....."
630 POSITION 1,13:PRINT " CARGO (D/H)...."
640 POSITION 1,15:PRINT " CONFIRMACION...
650 POSITION 17,7:PRINT ">
    ": POSITION 18,7
660 FOR I=1 TO 6
670 GET #1,M:M$=CHR$(M)
690 IF M$="E" AND I=1 THEN RETURN
700 IF M$="" THEN 670
710 IF M$<"0" OR M$>"9" THEN 670
720 PRINT M$;:FECHA(I)=VAL(M$):NEXT I
770 POSITION 17,7:PRINT " :POSITION 17,9:
    PRINT ">":POSITION 18,9
790 CON$="":FOR I=1 TO 20
800 GET #1,M:M$=CHR$(M)
820 IF M$="" THEN 800
830 IF M=155 THEN 870
840 PRINT M$;:CON$(I)=M$:NEXT I
870 POSITION 17,9:PRINT " ":POSITION
    17,-11:PRINT ">  ":POSITION 18,11
880 IMP$=" ":FOR I=1 TO 8
900 GET #1,M:M$=CHR$(M)
920 IF M=155 THEN I=9:GOTO 960
930 IF M$<"0" OR M$> "9" THEN 900
940 PRINT M$;:IMP$(I)=M$
960 NEXT I
980 IF VAL(IMP$)=0 THEN 870
990 POSITION 17,11:PRINT " ":POSITION
    17,18:PRINT "> ":POSITION 18,13
1000 GET #1,M:M$=CHR$(M)
1020 IF M$<>"D" AND M$<> "H" THE 1000
1030 PRINT M$;:CARGO$=M$
1050 POSITION 17,13:PRINT " ":POSITION
    17,15:PRINT "> :POSITION 13,15
1060 GET #1,M:M$=CHR$(M)
1080 IF M$<>"S" AND M$<> "N" THEN 1060
1090 PRINT M$;:VALE$=M$
1110 POSITION 17,15:PRINT " ":POSITION 8,20:
    PRINT "DATOS CORRECTOS (S/N)?"
1120 GET #1,M:M$=CHR$(M)
1140 IF M$<>"S" AND M$<>"N" THEN 1120
1150 POSITION 0,20:PRINT "
1160 IF M$="S" THEN 1180
1170 GOTO 650
1180 VAR=A*40-40:A=A+1:FOR I=1 TO 40:
    DATO$(VAR+I)=" :NEXT I
1200 IF VALE$(1)="N" THEN DATO$
    (VAR+1)="*"
1210 FOR I=1 TO 8: DATO$(VAR+I+1)=STR$
    (FECHA(I)):NEXT I
1220 MID$(DATO$,VAR+9,VAR+28)=CON$
1230 LON=LEN (IMP$):MID$(DATO$, VAR+
    40-LON,VAR+39)=IMP$
1250 IF CARGO$(1)="D" THEN DATO$(VAR+
    30)=CARGO$:M$(1)="-"
1260 IF CARGO$(1)="H" THEN DATO$(VAR+
    30)=CARGO$:M$(1)="+"
1270 DATO$((A-1)*40+13)="FIN DE DATOS":
    DATO$(501*40)=" "
1280 MID$(M$,2,2+LON)=IMP$:M=VAL(M$):
    SR=SR+M
1290 IF VALE$(1)="S" THEN SC=SC+M
1300 GOTO 540
1310 CLS:PRINT " GUARDAR FICHERO"
1320 GOSUB 1440:OPEN #2,8,0,M$
1330 PRINT #2;A:PRINT #2;SR:PRINT #2;SC
1340 FOR I=0 TO A:M$=MID$(DATO$,I*40+1,
    I*40+40)
1360 PRINT #2;M$:NEXT I:CLOSE #2: RETURN
1880 CLS:PRINT " CARGAR FICHERO"
1390 GOSUB 1440:OPEN #3,4,0,M$
1400 INPUT #3;A:INPUT #3;SR:INPUT #3;SC
1410 FOR I=0 TO A:INPUT #3;M$:MID$-
    (DATO$,I*40+1,I*40+40)=M$
1430 NEXT I:CLOSE #3:RETURN
1440 POSITION 2,12: PRINT "DISPOSITIVO:FI-
    CHERO ";:INPUT M$:RETURN
```

PROGRAMA 2: programa contable.

El lenguaje C (1)

El aspecto serio de la programación

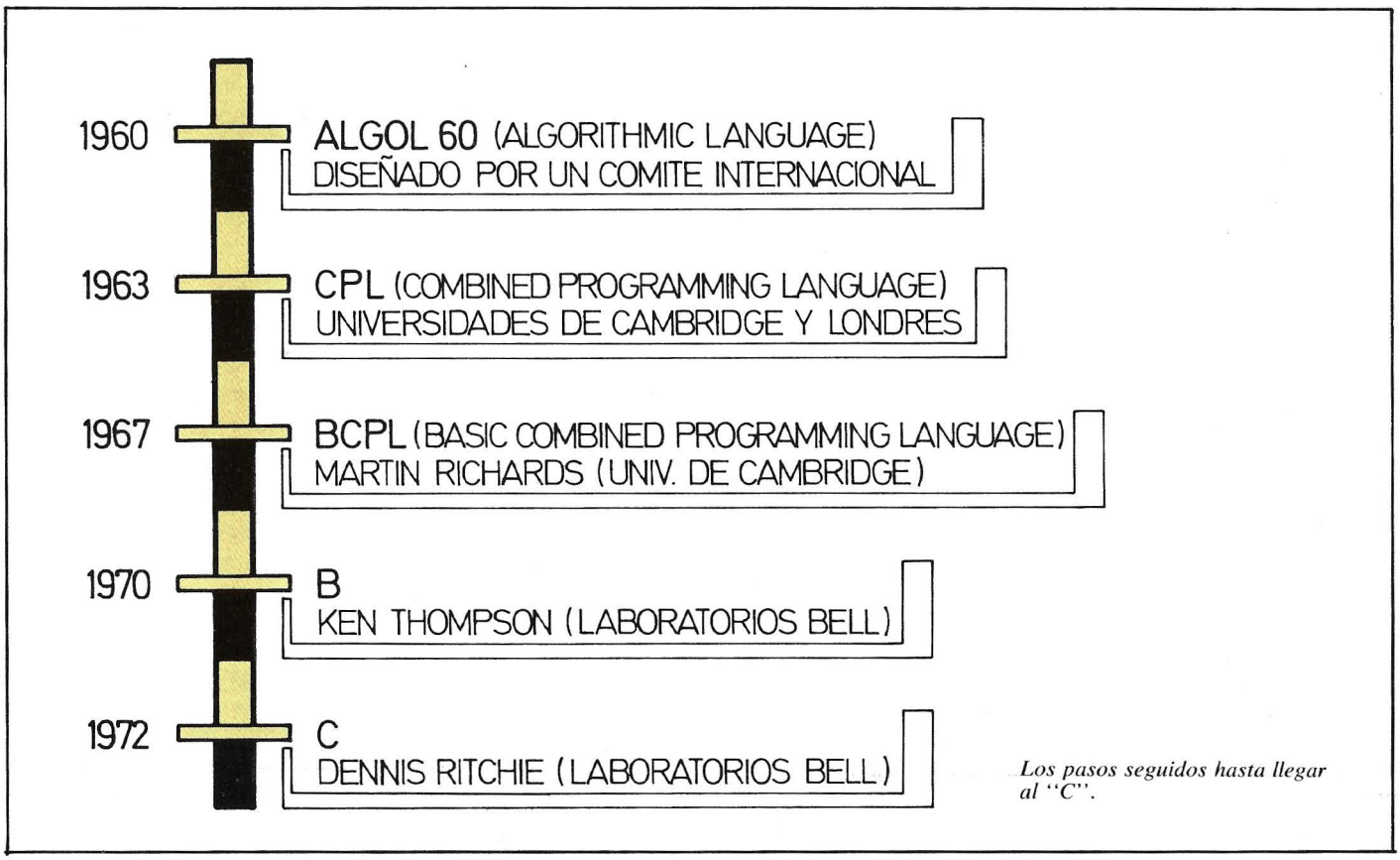
Los orígenes de C se remontan al año 1972, en los laboratorios de la compañía estadounidense Bell. Contrariamente a la tendencia actual, según la cual los grandes proyectos informáticos son encomendados a un equipo de profesionales, C surgió de la mente de un solo hombre, Dennis Ritchie, que por aquel entonces trabajaba en la citada compañía.

No es posible hablar de los orígenes de C sin hacer mención del sistema operativo UNIX. Los años 70 fueron testigos del nacimiento de UNIX, el cual, posteriormente, se convierte en un estándar «de facto» (una norma que, sin tener carácter de ley, es aceptada casi universalmente) para los sistemas operativos multiusuario en el campo de los grandes ordenadores. UNIX proporciona a sus usuarios una serie de herramientas para ayudar a la confección de programas; C era una de esas herramientas, y muy pronto se convirtió en «la» herramienta, hasta el punto de que el propio compilador de C y la mayor parte de UNIX se reescribieron en dicho lenguaje. Veamos ya aquí un aspecto de C sobre el que volveremos más adelante: su capacidad para la creación de software de sistemas.

LA HISTORIA DEL C

Los pasos seguidos por los lenguajes de programación hasta llegar al C son un ejemplo de la sencillez y claridad con la que nos gustaría encontrarnos más a menudo. En la figura adjunta se ilustran estos pasos.

Algol es el segundo "gran lenguaje" de



Los pasos seguidos hasta llegar al "C".

programación desarrollado en la era informática. Fortran fue el primero, y Algol intentó mejorarlo en sus muchos puntos débiles, cuidando especialmente la sintaxis y proporcionando una estructura modular. El resultado fue un lenguaje demasiado abstracto y general, por lo que nunca disfrutó de gran aceptación. CPL intentó, sobre las bases de Algol, traer las cosas más próximas a la realidad; de cualquier manera, Algol todavía estaba demasiado presente, por lo que el nuevo lenguaje seguía siendo difícil de aprender e implementar. BCPL intentó solucionar estos problemas extrayendo las características básicas de CPL.

Siguiendo la misma línea, B es un nuevo refinamiento de BCPL.
El resultado de esta evolución fue que los lenguajes eran cada vez más fáciles de implementar y de aprender, a costa de haberlos restringido a dominios de aplicación cada vez más estrechos. La virtud de C está en dar un paso atrás en la trayectoria seguida, añadiendo algo más de generalidad y manteniendo casi el mismo nivel de dificultad.

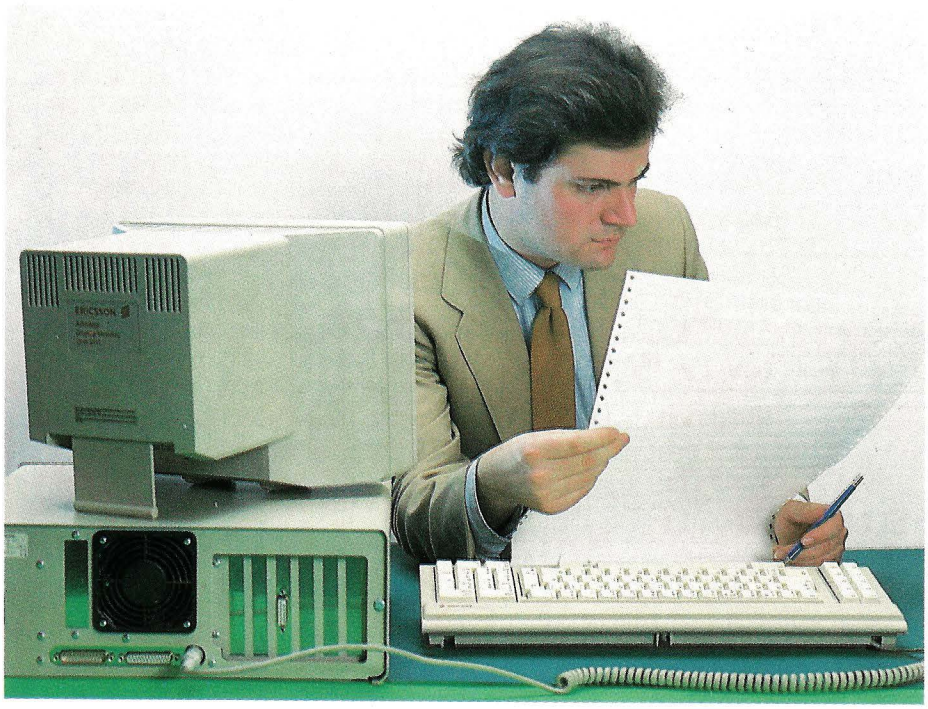
```
MAIN ( )
{
  PRINTF ("PRIMER PROGRAMA \N");
}
```

Ejemplo de programa confeccionado en "C". En la pantalla se observa que el "C" es un lenguaje de los denominados "de formato libre".

SALIDA

\N	SALTO DE LINEA
\T	SALTO AL SIGUIENTE TABULADOR
\R	VUELTA AL PRINCIPIO DE LINEA
\\	PARA ESCRIBIR UNA SOLA BARRA \

Algunas secuencias de escape.



El "C" es un lenguaje de programación especialmente pensado para su uso en el desarrollo de software de sistemas.

¿QUE ES EL C?

Como ya se ha comentado anteriormente, C está especialmente pensado para su uso en el desarrollo de software de sistemas. El software de sistemas enlaza al hardware con el usuario final del equipo, bien sea un ordenador personal o un gran ordenador. El primer ejemplo de software de sistemas es el propio sistema operativo.
Esto no quiere decir que no sea posible hacer un programa de juegos en C (prácticamente se puede hacer casi todo con cualquier lenguaje), sino que, simplemente, sacaremos más rendimiento a nuestro tiempo aplicando C al software de sistemas.
C es un lenguaje que permite estar lo suficientemente próximos a la máquina

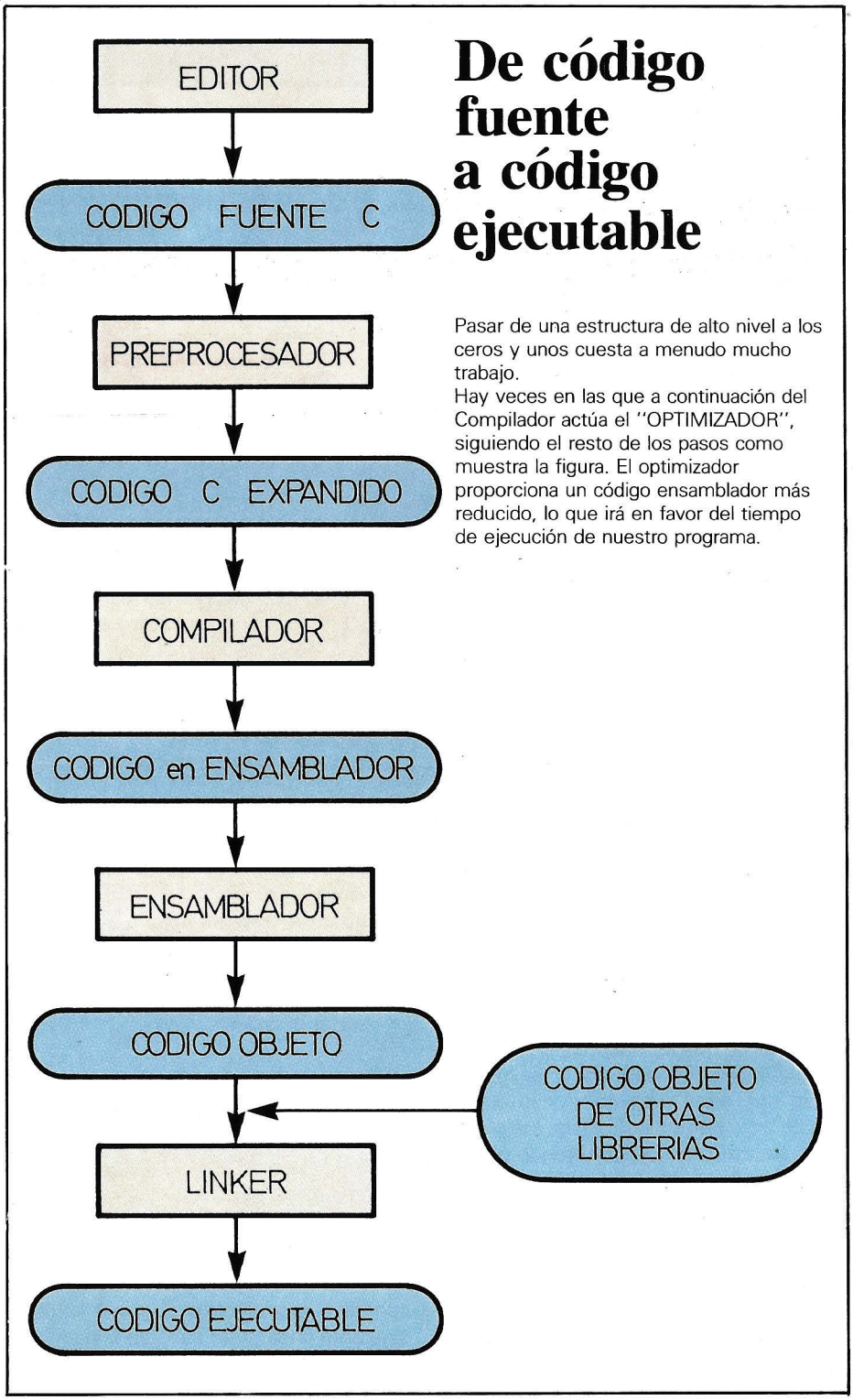
como para manejar directamente posiciones de memoria, bits aislados dentro de estas posiciones, e incluso, de alguna forma, los propios registros de la CPU; a la vez que proporciona las estructuras características de los lenguajes de alto nivel, tan importantes para hacer de la programación una tarea no tan difícil. Por todo esto, cabe afirmar que el C se encuentra a mitad de camino entre un ensamblador y un lenguaje de alto nivel de tipo estructurado como el PASCAL.

EL CAMINO HASTA EJECUTAR UN PROGRAMA EN C

En la actualidad no existen intérpretes de C: C siempre es compilado. En la figura se observan los pasos que transcurren desde la creación del programa por medio del editor, hasta conseguir un fichero con el programa listo para ser ejecutado.
Uno de los objetivos de Dennis Ritchie al concebir el lenguaje C era la simplicidad y sobre todo la portabilidad. Por tal se entiende el que un programa en C pueda ser ejecutado en máquinas totalmente diferentes, con distinta arquitectura y configuraciones de entrada/salida. Como ya se ha comentado, el propio sistema operativo UNIX está escrito mayoritariamente en C, y en la actualidad, fabricantes tan diversos como Digital, IBM o Honeywell incorporan como sistema operativo a UNIX.
C no posee instrucciones específicas de entrada/salida. Una vez que el ensamblador ha generado el código objeto quedarán lo que técnicamente se denominan "referencias sin resolver", que son básicamente llamadas a subprogramas que realizan las operaciones de entrada/salida. El "linker" tomará dicho código objeto y accederá a los ficheros donde se encuentran las referencias, incluyendo en el nuevo código generado los segmentos de programa necesarios para "resolver" tales referencias.

APARENCIA DE UN PROGRAMA EN C

La estructura fundamental a partir de la cual se forman programas en C es la función. Todo programa debe tener al menos una de ellas y siempre habrá una que se llame "main" —principal en inglés—. Pueden estar repartidas en cualquier orden a lo largo del programa y la ejecución empezará siempr por "main"; aunque no hay ninguna regla sintáctica que nos obligue a ello, se suele colocar a "main" al principio del listado.
C es un lenguaje de los llamados "de for-



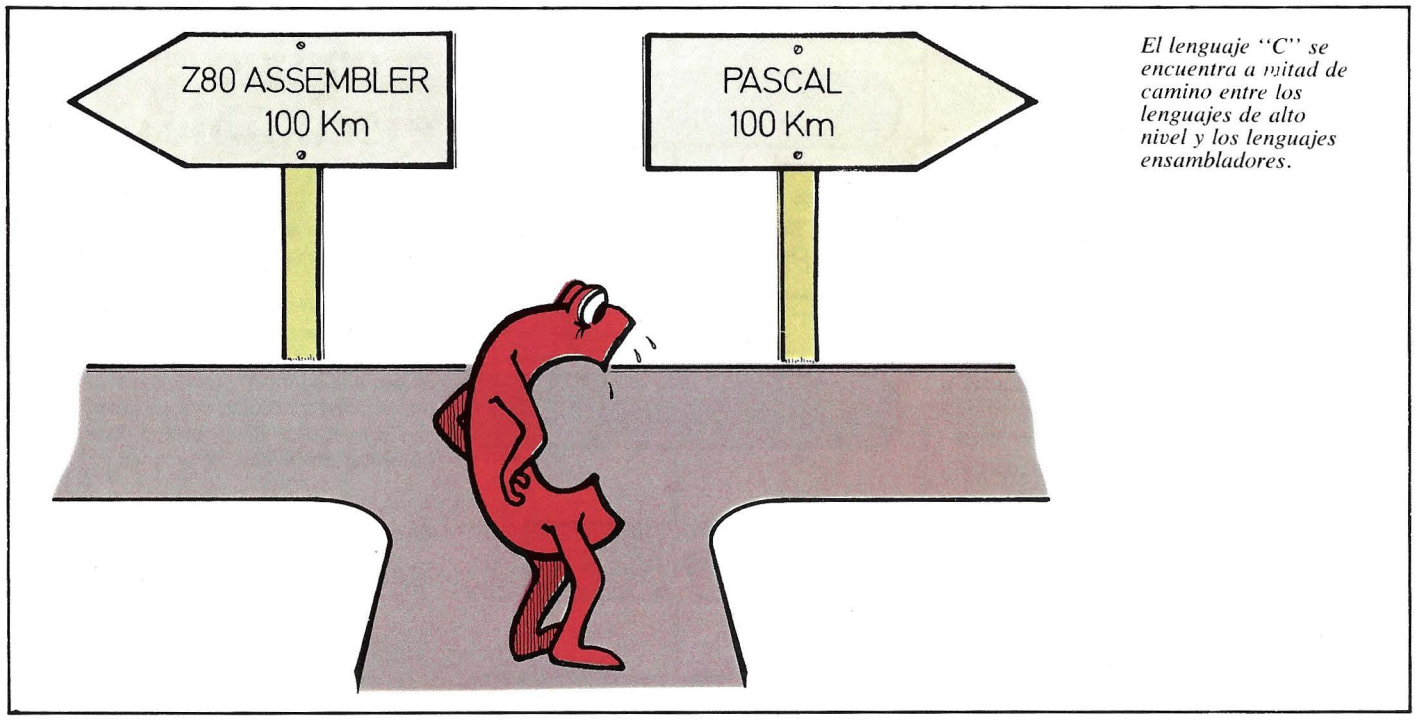
De código fuente a código ejecutable

Pasar de una estructura de alto nivel a los ceros y unos cuesta a menudo mucho trabajo.
Hay veces en las que a continuación del Compilador actúa el "OPTIMIZADOR", siguiendo el resto de los pasos como muestra la figura. El optimizador proporciona un código ensamblador más reducido, lo que irá en favor del tiempo de ejecución de nuestro programa.

El siempre presente ";"

Los conocedores del Lenguaje Pascal están acostumbrados a "dejar caer" un ";" al final de casi todas las sentencias.

En C también hay que utilizar el ";" muy a menudo, aunque su significado respecto a Pascal varía sutilmente.
En Pascal, un ";" se utiliza como separador de sentencias, mientras que en C se usa como terminador de sentencias. Por esta razón, nunca va un ";" antes de un ELSE en Pascal ya que entonces el compilador tomaría a ELSE como principio de una nueva sentencia, y esto no está de acuerdo con la sintaxis del Pascal. En C, sin embargo, si aparecerán ";" antes de los "else" ya que, por su definición sintáctica, siempre ha de seguir una sentencia a un "if", y una sentencia debe ser terminada por un ";", por lo que si a continuación hay un "else", éste se verá precedido del dichoso símbolo.



El lenguaje "C" se encuentra a mitad de camino entre los lenguajes de alto nivel y los lenguajes ensambladores.

mato libre"; esto quiere decir que, al igual que en PASCAL, los espacios en blanco no tienen significado, y son utilizados para resaltar la estructura del programa. Los comentarios van entre los grupos / * y * /.
En la figura adjunta aparece un ejemplo de programa en C. Como vemos, está presente "main" seguido de un par de paréntesis. Toda función tendrá unos argumentos sobre los que se elaborará el resultado de la función. En este caso, la función "main" no tiene ninguno, por lo que la lista de argumentos está vacía; los paréntesis, sin embargo, siguen siendo obligatorios. Las llaves ({ }) delimitan el cuerpo de la función y son equivalentes a los BEGIN-END de Pascal.
En el cuerpo de "main" nos encontramos una sentencia de salida, la cual escribe por pantalla el ar um nto que tiene entre comillas dobles. Por pertenecer esta sentencia al grupo de entrada/salida, NO es parte de C; podría haberse llamado de cualquier otra forma que le hubiera gustado al implementador de C de nuestro ordenador, aunque el nombre de "printf" está universalmente aceptado.
La salida del programa es:

```
primer programa
—
```

con el cursor situado al principio de la siguiente línea. Este salto de línea lo produce el grupo "\n" (en algunas máquinas que no tienen el símbolo "\" se utiliza una Ñ) y se denomina "secuencia de escape"; como vemos, las secuencias de escape no son escritas en pantalla, sólo sirven —fundamentalmente— para controlar el cursor. Hay varios tipos según ilustra la correspondiente figura. De no haber finalizado el argumento de "printf" con la secuencia "\n", la salida habría sido la misma pero con el cursor situado al final de la línea recién escrita.
Las secuencias de escape pueden ir en cualquier lugar del argumento; cuando se detecta una barra invertida se analiza el siguiente carácter para ver dónde se sitúa el cursor y se sigue escribiendo el resto del argumento. Así, por ejemplo:

printf ("primera frase\r segunda frase");
da como resultado:
segunda frase—

ya que \r vuelve el cursor a principio de línea y al seguir escribiendo el resto de los caracteres quedan borrados los anteriores.

Unix (4)

La herramienta Shell

Uno de los puntos básicos de un sistema operativo se concreta en su comunicación con el usuario del ordenador. La interacción debe ser sencilla y cómoda para el usuario, con las consiguientes ventajas a la hora de desarrollar programas o, simplemente, de recabar información de las entrañas del equipo. Algunos elementos del sistema operativo que dan paso a esta eficaz interrelación ordenador/usuario son:

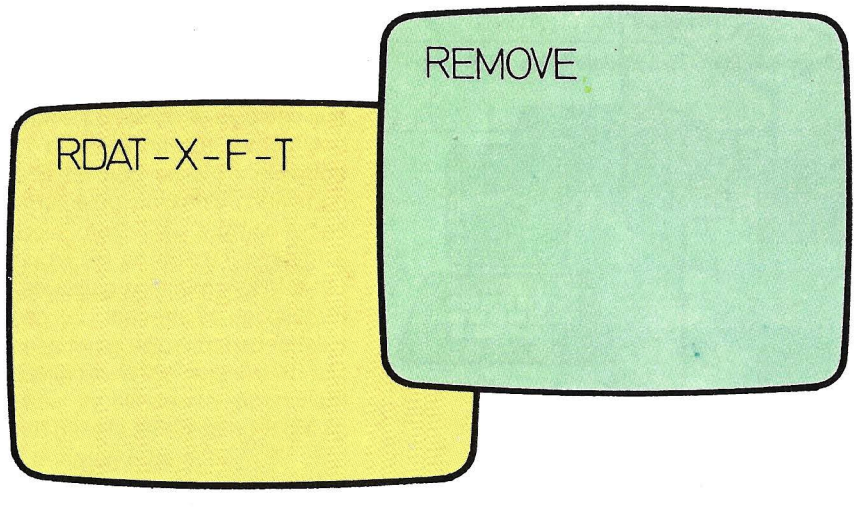
— Comandos con nombres claros y constituidos por una sola palabra.
— Ausencia o número limitado de parámetros en el comando.
— Posibilidad de emplear abreviaturas en los nombres de los comandos.
— Mensajes de error claros y cubriendo el mayor número posible de errores distintos.

Un sistema operativo con estas características podría ser definido como "amigable" para el usuario, toda vez que le proporciona una gran ayuda en su trabajo, descargándole de la necesidad de memorizar largas secuencias de comandos y parámetros.

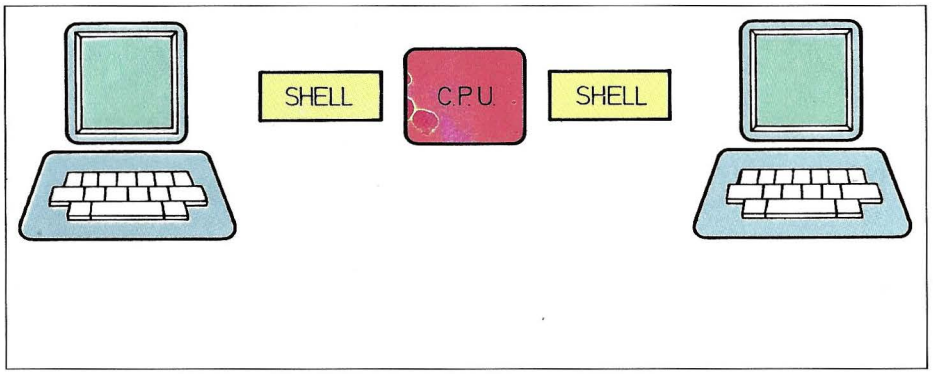
En el extremo inferior de esta escala de bondad se encuentran sistemas operativos como el CP/M y el MS-DOS; los cuales presentan un repertorio de comandos más o menos fanagoso y con múltiples parámetros. En el extremo opuesto aparece el sistema operativo de los ordenadores LISA y MACINTOSH de Apple, en los cuales toda la gestión se realiza a través de menús de tipo persiana y símbolos gráficos (iconos) por pantalla; no se hace empleo de comandos tecleados, en lo que cabe catalogar como un ejemplo máximo de facilidad en la interface ordenador/usuario.

El sistema operativo UNIX se encuentra a mitad de camino entre ambos extremos. Por su propia concepción, el UNIX dispone de una herramienta denominada Shell a través de la cual se gestionan los diferentes comandos que se dan al ordenador. Con esta utilidad el sistema puede acondicionarse a las necesidades de los distintos usuarios, estableciendo, en caso de necesidad, una estructura en forma de menús de manera que a través de éstos sea posible el acceso a distintos comandos o programas de aplicación. Por esta causa, el UNIX es un sistema operativo cómodo para el usuario; sin llegar a la facilidad de manejo propia del sistema operativo que gobierna a los ordenadores LISA y MACINTOSH, presenta unas características sumamente prácticas para el usuario. Un extremo asociado a esta constatación, también importante, lo constituye el hecho de que las diferentes interfaces de los usuarios con el ordenador pueden ser distintas entre sí, con lo cual el sistema operativo puede parecer distinto a cada usuario, no estando obligado por tanto a una interface estándar que podría no resultar totalmente adecuada a sus necesidades.

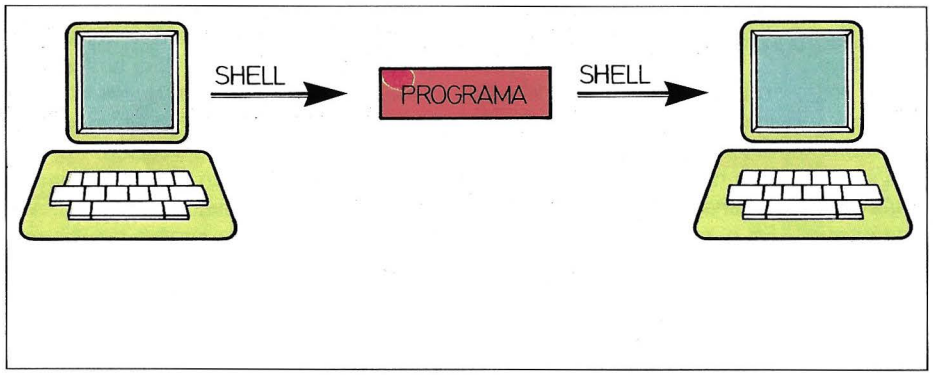
En la actualidad existen dos tipos de Shell que cabe considerar como los más comu-



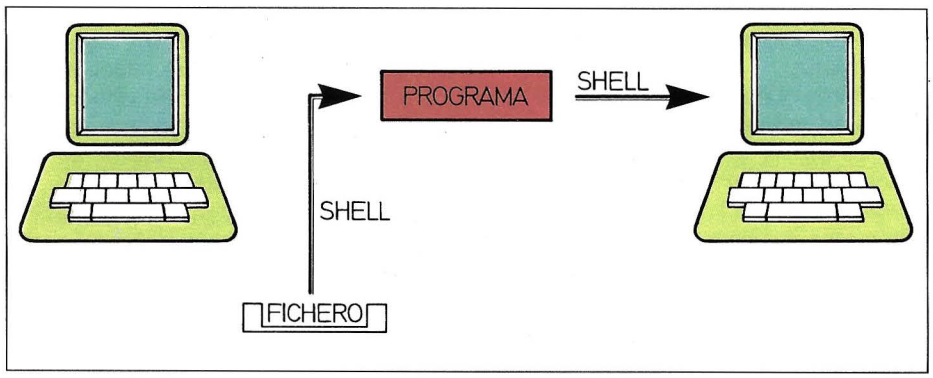
Un sistema operativo ha de buscar una interrelación simple y cómoda con sus usuarios.



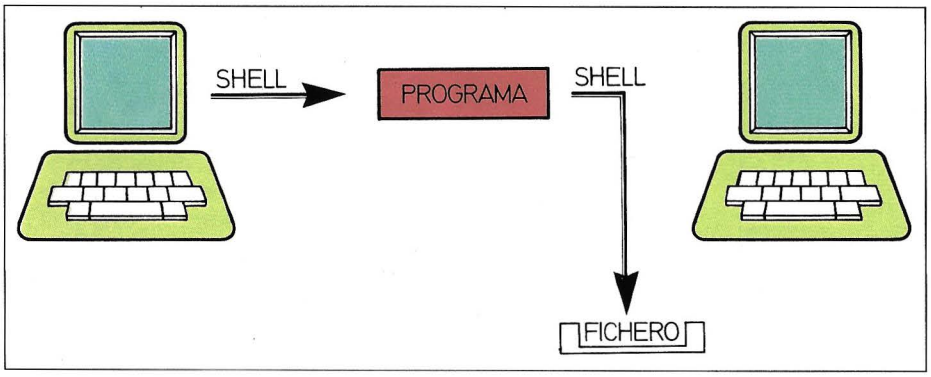
El Shell es el elemento de conexión entre los deseos del usuario y la máquina.



Proceso normal de entrada y salida estándar.



Proceso de redireccionamiento de la entrada estándar (teclado) dirigiéndola hacia un fichero residente en memoria externa.



Redireccionamiento de la salida estándar (pantalla) hacia un fichero.

nes: el Shell Bourne, correspondiente al sistema operativo UNIX estándar, y el Shell C, correspondiente al UNIX desarrollado en Berkeley. Esta utilidad, el Shell, no es sólo una herramienta para la gestión de los distintos comandos; además es un lenguaje de programación de alto nivel, a través del cual se pueden gestionar las llamadas a distintos programas, empleando e implementando una estructura "pipeline" en la cual la salida de un programa es utilizada como entrada de otro.

LA LINEA DE COMANDOS

La ejecución de cualquier programa se lleva a cabo indicando al sistema operativo un comando. La línea que contiene el comando, incluyendo los distintos argumentos, se denomina línea de comandos. Su sintaxis es la que determina la ortografía, puntuación y distribución de los distintos elementos que intervienen en un comando.
La estructura general de una línea de comando toma el siguiente aspecto:

comando (arg 1) (arg 2) ..... (arg w)

Los diferentes argumentos de un comando no son estrictamente necesarios; su necesidad o no depende exclusivamente del comando en cuestión y de la función que éste lleve a cabo. El argumento como tal puede estar compuesto por un texto, número, nombre de un fichero o, en general, por cualquier dato que indique al comando sobre qué tiene que actuar. Como ejemplo más simple podríamos indicar que en el caso de un comando de edición, un argumento podría ser el nombre del fichero sobre el cual va a trabajar. Dentro de los argumentos están los denominados opciones, los cuales modifican los efectos del comando, no existiendo limitación en el número de opciones a incluir en el conjunto de un comando. Su estructura va siempre precedida por un guión en el caso de los programas de utilidad pero no en el Shell en sí.
El tratamiento de cualquier comando se realiza a través de un buffer intermedio en

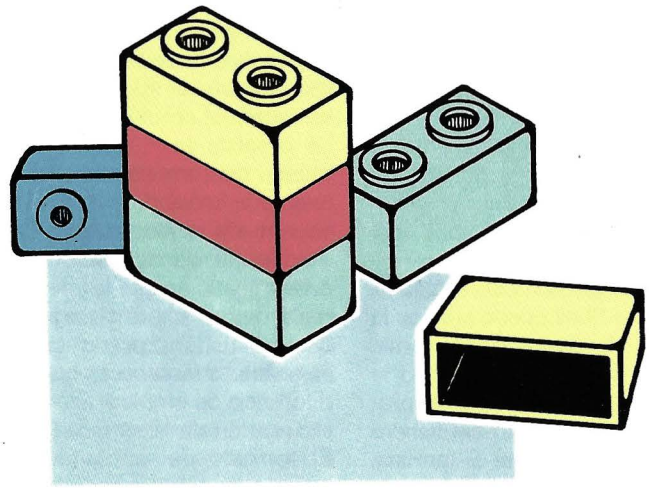
el cual se van almacenando los distintos caracteres que constituyen la línea de comandos. Los caracteres van siendo analizados individualmente, en espera de encontrar algún carácter de control, como pueda ser una indicación de eliminar caracteres o incluso una línea completa de comandos. Caso de no ser así, éstos son almacenados en el buffer; una pulsación de la tecla RETURN hará que el contenido del buffer, coincidente con el conjunto de la línea de comandos, sea enviado al Shell para que éste lleve a cabo el tratamiento del conjunto.
El Shell lleva a cabo un proceso hasta cierto punto inverso al anterior. Mientras que inicialmente se efectuaba una carga en el buffer, ahora se realiza una descomposición de la línea en sus distintos componentes. El comando es identificado buscando el grupo de caracteres localizado hasta el primer espacio. Caso de que este nombre no coincida con ninguno de los conocidos, se emitirá un mensaje de error. Una particularidad del Shell es el hecho de que no tiene por sí mismo capacidad de discurrir si una opción particular o cualquier otro argumento es válido para un programa determinado. Los mensajes de error correspondientes a estas situaciones han de ser considerados por el propio programa, dándose el caso de que muchos programas de utilidad UNIX ignoran las opciones erróneas.
La ejecución del comando se inicia una vez que el Shell ha encontrado un programa con el mismo nombre. En este momento pasa las opciones y argumentos al programa y éste comienza su ejecución. Mientras ésta se produce, el Shell entra en un estado "durmiente" hasta que recibe una señal indicando que el programa ha finalizado su ejecución, instante en el que pasa de nuevo a un estado activo en el cual queda en disposición de gestionar nuevos comandos.

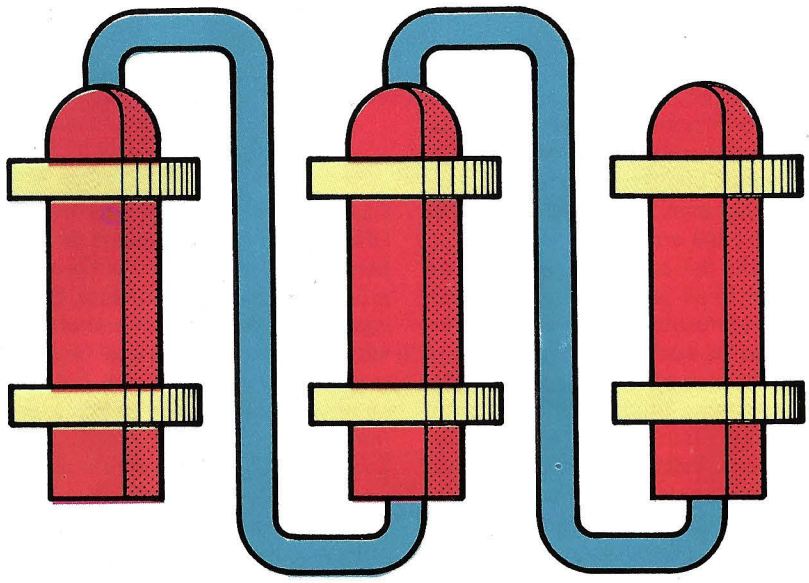
ENTRADA Y SALIDA ESTANDAR

Cualquier comando del sistema operativo UNIX lleva a cabo la gestión de los distintos elementos de información contenidos en la memoria del ordenador. En algunos casos la respuesta a un comando es una lista de información, la cual ha de llegar al usuario de una u otra forma. El lugar al que se envía la información es lo que se denomina salida estándar; normalmente, el programa nunca sabrá dónde se envía esta información, si es enviada a un terminal o a un fichero. De igual manera encontramos que todo comando ha de recibir sus órdenes de algún sitio, bien sea de un

Visión artificial

La vida real ha presentado a la tecnología multitud de retos en el deseo de esta última de repetir o imitar fenómenos de la Naturaleza. Uno de los fenómenos naturales básicos y que sin embargo presenta a la tecnología un problema de enorme magnitud es el fenómeno de la visión.
La respuesta tecnológica a este fenómeno ha dado muchos resultados diferentes, pero hasta ahora los sistemas puestos en operación y capaces de "ver" son sistemas que trabajan en contextos muy específicos y limitados, normalmente en la inspección de circuitos integrados o en la manipulación de piezas dentro de una cadena de montaje. Este tipo de sistemas operan normalmente sobre imágenes binarias (en blanco y negro), no pudiendo aceptar la escala de grises que produciría el empleo de una cámara normal de televisión.
El problema que en la actualidad se intenta resolver, y que es algo fundamental en la vida real, es el de la selección de objetos en un recipiente con distintos elementos. Los sistemas de manipulación anteriormente mencionados trabajan sobre objetos situados separadamente sobre una placa horizontal, mientras que ahora se trata de escoger entre los distintas piezas del recipiente la más adecuada a la operación que se lleva en curso. Los ordenadores que controlan este proceso han de ser sumamente rápidos, para poder dar respuesta al usuario en un intervalo de tiempo razonable.
En la actualidad, partiendo de estudios médicos sobre el fenómeno de la visión humana, se están desarrollando ordenadores para este tipo de actividades. Internamente están organizados para gestionar los procesos lógicos asociados con la visión, según una arquitectura de procesadores en paralelo, en la que distintos procesadores tratan simultáneamente diferentes parcelas de la información.
Pese a que ya en los primeros años 60 Larry Roberts, en el MIT, puso los cimientos al problema de la comprensión de imágenes estableciendo una serie de métodos de análisis, y pese al gran avance tecnológico producido desde entonces, la idea de que una máquina pueda comprender imágenes de la vida real como una persona parece, aún lejana.





En la estructura "pipeline", la salida de un proceso constituye la entrada de otro.

fichero o a través de un terminal. Este conjunto es lo que se conoce por entrada estándar.

Hemos visto la estructura de los directorios y ficheros en el sistema operativo UNIX. Sin embargo, ha de señalarse la presencia de un tipo de fichero denominado "fichero de periférico". Este tipo de fichero es residente en la estructura del sistema operativo UNIX y representa a un dispositivo periférico como puede ser un disco, un terminal o una impresora. Este fichero puede ser empleado como entrada y salida estándar de los distintos comandos y programas. Así, en el caso de un terminal, cuando un usuario se conecta al sistema, el Shell dirige la salida estándar al fichero de periférico que representa a su terminal, de forma que todas las informaciones de salida aparecen en el terminal. Igualmente, el Shell dirige la entrada estándar para que provenga del mismo fichero, de forma que cualquier cosa tecleada en el terminal es un "input" del programa.

Este tipo de operación sería muy rígida si no fuera porque el Shell puede redirigir la entrada y salida de cualquier programa. Esta redirección se realiza asociando la salida y entrada estándar de cualquier programa con un fichero distinto del fichero de periférico correspondiente al terminal que se esté empleando. La redirección de la salida se realiza por medio del símbolo de redirección de salida (>), el cual va seguido por el nombre del fichero sobre el que va a producirse la salida.

La forma global de esta redirección sería la siguiente:

Nombre de programa (argumentos) > nombre de fichero salida

La redirección de la salida ha de realizarse con cuidado, toda vez que el Shell escribe sobre el fichero al que se redirige, borrando su contenido el cual se pierde. Para obviar esta posibilidad, o bien para llevar un control de las diferentes salidas sucesivas, es posible redirigir la salida de manera que la misma se añada al final del fichero de salida estándar correspondiente, no alterando la información previamente grabada. El símbolo que se coloca en lugar del empleado para la redirección de la salida es , manteniéndose un comando con una estructura totalmente similar a la señalada anteriormente.

La entrada estándar también es posible modificarla, de manera que un programa determinado acepte la información de entrada de un fichero en lugar de obtenerla del terminal. El proceso a seguir es similar al señalado anteriormente, con la única diferencia de emplear ahora el símbolo de redirección de la entrada (<).

El formato del comando queda ahora como sigue:

Nombre de programa (argumentos) < nombre del fichero de entrada.

ESTRUCTURA DEL PIPE

El sistema operativo UNIX, aprovechando la facilidad que el Shell proporciona al usuario para la redirección de la información, permite el crear una estructura de pipe o pipe-line en la cual es posible el empleo de la salida estándar de un programa como entrada estándar de otro; de esta forma la información generada por un proceso puede ser empleada por otro a continuación. El formato que toma esta estructura es el que sigue:

Nombre de programa 1 (argumentos) | nombre del programa 2 (argumentos)

Como se observa, la separación de los nombres se lleva a cabo por medio de una línea vertical.

Una posibilidad que da la estructura "pipe" es la de emplear lo que se denominan filtros. Un filtro es un programa que utiliza un vector de datos de entrada para producir un vector de datos de salida. Una línea de comandos que emplee un filtro utilizará la salida estándar de un programa como entrada del filtro y, posteriormente, la salida del filtro como entrada de otro programa.

Symphony (y 3)

Sesión práctica

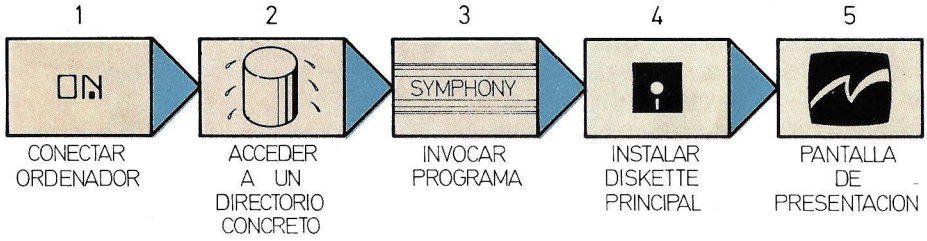
Sin duda el mayor motivo de crítica de los paquetes integrados en general, y de SYMPHONY en particular, consiste en su complejidad de uso. En efecto, al incorporar en un único programa las capacidades de cuatro o cinco aplicaciones, el número de comandos disponibles se multiplica también por cuatro o cinco, y además la estructura de menús se expande notoriamente. No obstante puede afirmarse que SYMPHONY puede ser explotado convenientemente por un usuario no experto en informática, sin más que dedicar un tiempo razonable a formación y entrenamiento.

El presente capítulo está dedicado exclusivamente a describir una sesión de trabajo con SYMPHONY, haciendo especial énfasis en la posibilidad total de transportar información entre los distintos entornos.

CONEXION Y PLANEAMIENTO DE LA SESION

Dado el tamaño del programa, supondremos por comodidad que la instalación donde se realiza nuestro trabajo dispone de una unidad de disco rígido; en ella se encuentran cargados tanto los distintos programas integrantes de SYMPHONY, como los ficheros con datos de usuario. En tal caso, para "arrancar" el programa son necesarios los siguientes pasos:

1. Conectar el ordenador.
2. Acceder al directorio de almacenamiento donde esté almacenado el programa.



En la figura se resumen los cinco pasos necesarios para arrancar el programa SYMPHONY.

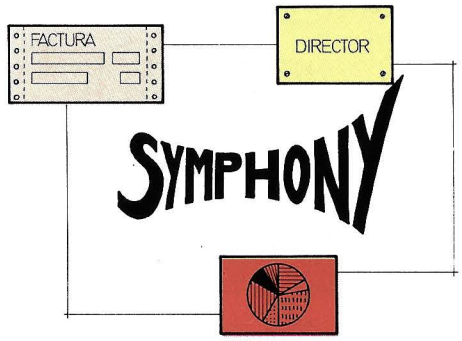
3. Invocar al programa, tecleando la palabra SYMPHONY y pulsando RETURN.
4. Instalar el disquete principal del programa en la unidad de disco flexible.
5. Visualizar en la pantalla el menú de arranque, donde se incluirá el número de copia del programa utilizado.

En principio puede parecer una contradicción necesitar uno de los disquetes teniendo todos os programas residentes en disco rígido; esto se debe a la autoprotección realizada por SYMPHONY de forma que no se pueden utilizar copias "piratas".

Una vez activada la aplicación y antes de comenzar a trabajar, se debe realizar un planteamiento de la sesión, ya que según los procesos que se desee realizar será necesario utilizar uno u otros entornos. En nuestra sesión los objetivos serán los siguientes:

1. Producir la facturación a partir de los datos introducidos por el usuario (número de unidades vendidas) y de la información de la base de datos (precio de los productos).
2. Obtener un informe gráfico en el que quede patente la importancia relativa tanto de los productos vendidos como de los clientes.
3. Preparar un informe para el director de la empresa en el que se describa la marcha general del departamento de ventas en el mes en curso.

En sesiones prácticas con otros programas hemos visto ejemplos muy similares; la novedad en este caso consiste en la integración de tres objetivos distintos en un único programa: SYMPHONY.



El objetivo de nuestra sesión de trabajo con SYMPHONY se centra en tres cuestiones: producir facturas, obtener resúmenes gráficos y producir un informe destinado a la dirección de la empresa.

ENTORNOS NECESARIOS

Para resolver los problemas planteados deben entrar en acción todos los entornos

de SYMPHONY, a excepción del de comunicaciones. Las tareas encomendadas en cada caso serán las siguientes:

- Base de datos.

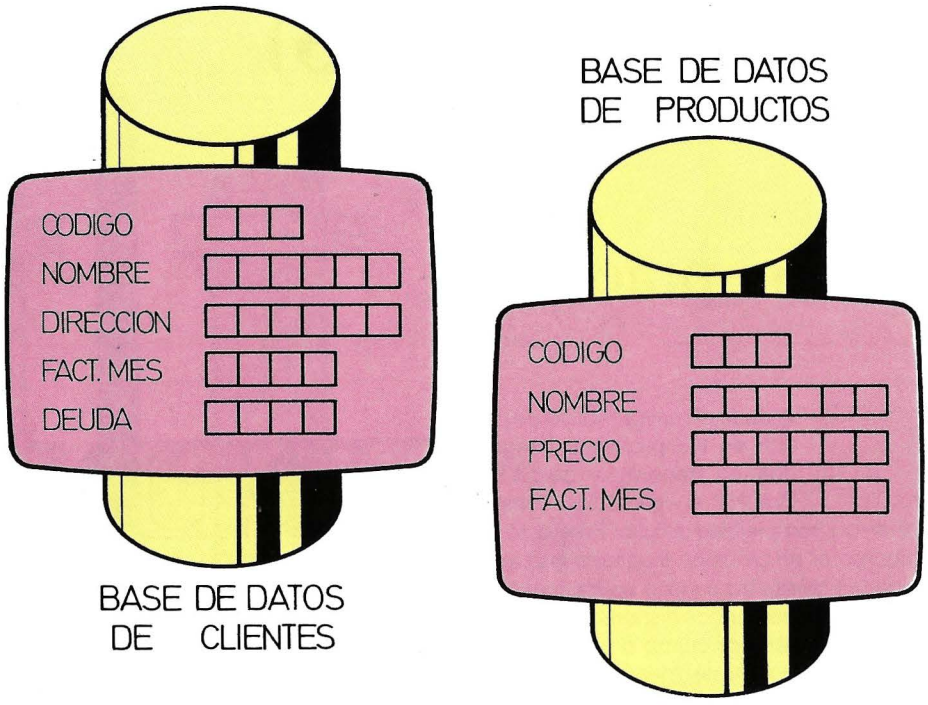
Serán necesarias dos estructuras de información: una con la información relativa a los clientes de la empresa y otra con los datos asociados a los productos vendidos.

- Hoja electrónica.

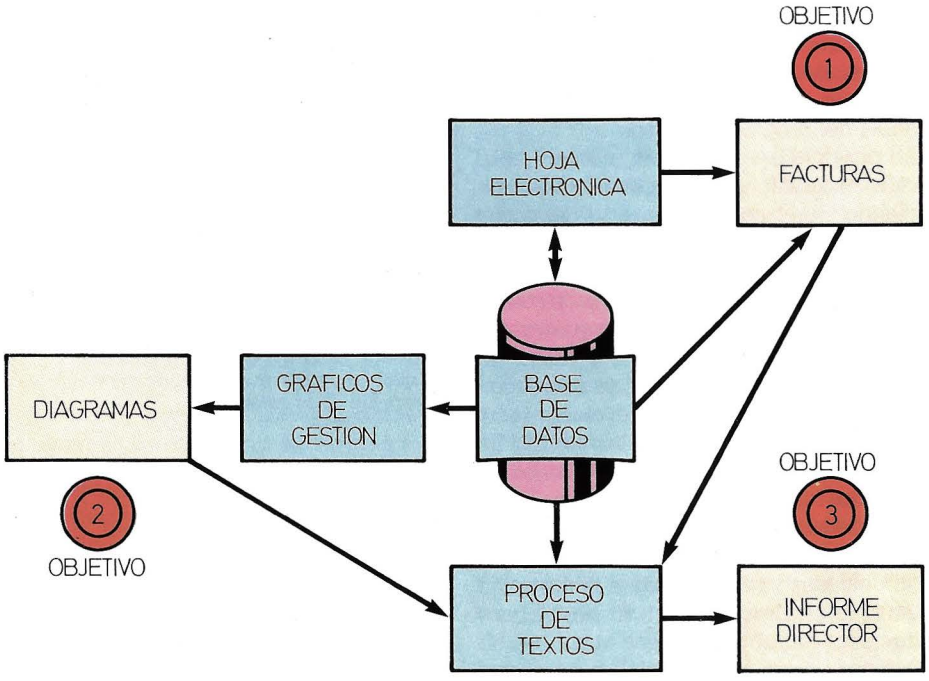
Se utilizará para producir las facturas. Su misión estará estrechamente relacionada con la base de datos, ya que para producir una factura se pretende que sólo sea necesario introducir el código del cliente y las cantidades vendidas de cada producto. A continuación, para completar la información de la factura, será necesario localizar el nombre del cliente y los nombres e importes unitarios de cada producto. Además de leer de la base de datos, al producir las facturas se debe escribir en ella, de forma que queden almacenados los datos necesarios para el resto de los informes.

- Gráficos de gestión.

Dentro de las posibilidades ofrecidas por SYMPHONY en este entorno se elegirán dos. Por un lado un diagrama de barras en el que se visualizará la facturación realizada en el mes a cada uno de los clientes y, por otro lado, mediante un diagrama de tarta se representarán los porcentajes de ventas para cada uno de los productos.



La información necesaria se almacenará mediante el entorno base de datos y, posteriormente, se recuperará para conseguir los objetivos marcados.



La imbricación entre todos los entornos en la sesión de trabajo queda patente en la presente figura.

- Tratamiento de textos.

Por último, mediante el entorno para el tratamiento de textos, se obtendrá el informe del estado de la empresa. En él se deben reflejar textualmente no sólo los datos cuantitativos, que ya quedaron patentes en los gráficos, sino más bien las opiniones personales del informador a partir de dichos datos.

Queda claro en este caso que el centro más importante de información lo constituye la base de datos; no obstante, esta característica no impide que el centro técnico esté en la hoja electrónica, ya que desde ella se controlará la propia base de datos.

LA SESION DE TRABAJO

Después de activar el paquete SYMPHONY, el primer objetivo consistirá en producir las facturas; para ello caben dos alternativas:

1. Mediante macros.

En el caso de haber programado previamente una macro encargada de producir las facturas, nos limitaremos a ejecutar dicho programa e introduciremos la infor-

mación que nos solicite. De esta forma será la macro la encargada de "traducir" los códigos de cliente y producto, y realizará automáticamente las operaciones necesarias para calcular el importe final de la factura.

2. Mediante la hoja electrónica.
La primera operación a realizar consistirá en cargar el modelo de hoja que previamente se habrá diseñado; inmediatamente introducirán los valores variables y SYMPHONY calculará el importe y "traducirá" los códigos.
Mediante cualquiera de los dos métodos el resutado final consistirá en un número variable de facturas y en la actualización de las bases de datos de clientes y productos, de acuerdo a los importes facturados.
Tras haber repasado las facturas y comprobado la ausencia de errores estaremos en disposición de obtener los informes gráficos. Para ello se cargará en la hoja electrónica la información correspondiente a los importes facturados a cada cliente. A continuación se asignarán los parámetros necesarios para la obtención de gráficos, indicando títulos generales, rangos a considerar y tipo de gráfico, que para el diagrama de facturación/clientes será de barras. Una vez realizadas estas operaciones se visualizará en la pantalla el gráfico correspondiente y, en el caso de así desearlo, se producirán las oportunas copias por impresora. Los pasos necesa-

Redes de ordenadores

Se llama red de ordenadores a un conjunto de equipos interconectados entre sí para permitir la utilización conjunta de ciertos recursos, como programas, ficheros, etc.
Básicamente se puede hablar de tres tipos de redes de ordenadores según su esquema de funcionamiento.

1. *Redes centralizadas*

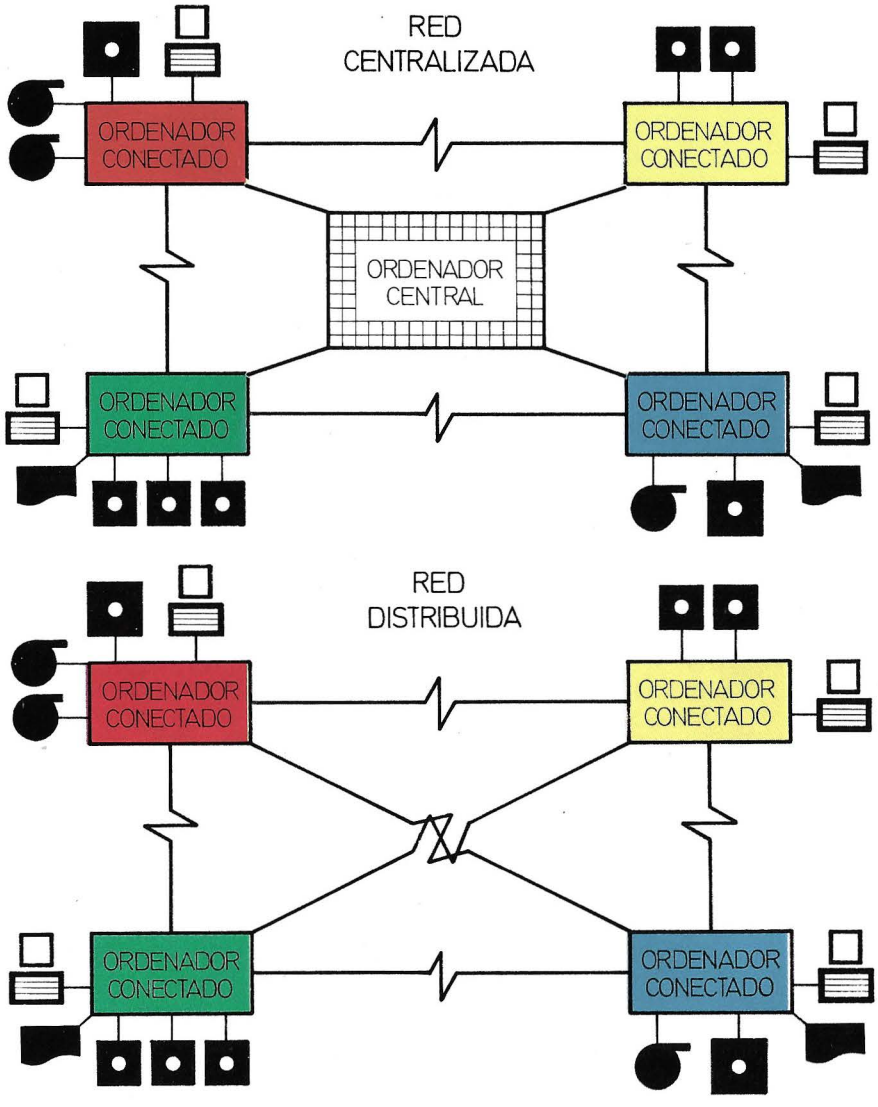
Los distintos ordenadores conectados en la red se comunican a través de un ordenador central, al que se le puede considerar como el gestor de toda la red. Dado que su misión suele ser relativamente compleja, este ordenador es el más potente de toda red. La principal virtud de este tipo de red estriba en que la utilización del ordenador central libera de ciertas tareas al resto de los ordenadores de la red; y su principal defecto consiste en que una avería en el ordenador central provocará la inmediata "caída" de toda la red.

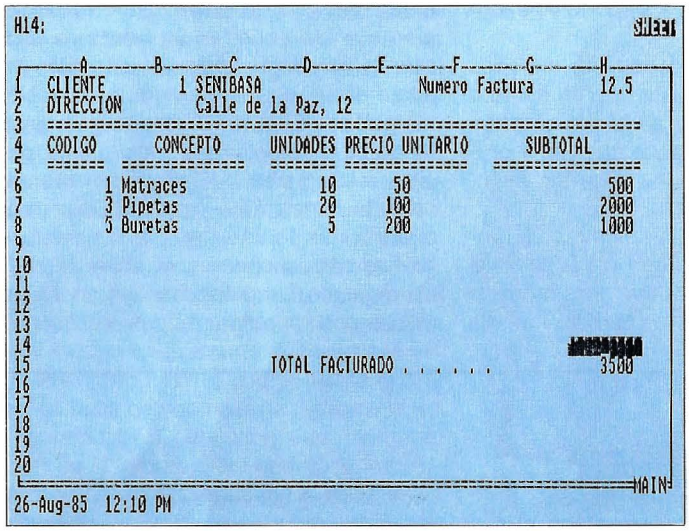
2. *Redes distribuidas*

En este caso no existe ningún ordenador distinguido, es decir todos los ordenadores están conectados entre sí y se encargarán de establecer comunicaciones particulares cuando lo necesiten. Lo normal es que cada equipo esté conectado al menos con otros dos. La principal desventaja de este modelo consiste en que todos los ordenadores tienen que dedicar parte de su "tiempo" a labores de comunicación, y su principal ventaja es que la avería de uno de ellos no debe afectar al funcionamiento global de la red.

3. *Redes mixtas*

En el fondo este tipo de redes son una variante de las redes distribuidas. Se limitan a proporcionar acceso a terminales conectados a la red; terminales que no necesitan para resolver las misiones que tengan encomendadas toda la potencia de un ordenador.





Ejemplo de factura producida con SYMPHONY, tal como se visualizaría en la pantalla del ordenador.

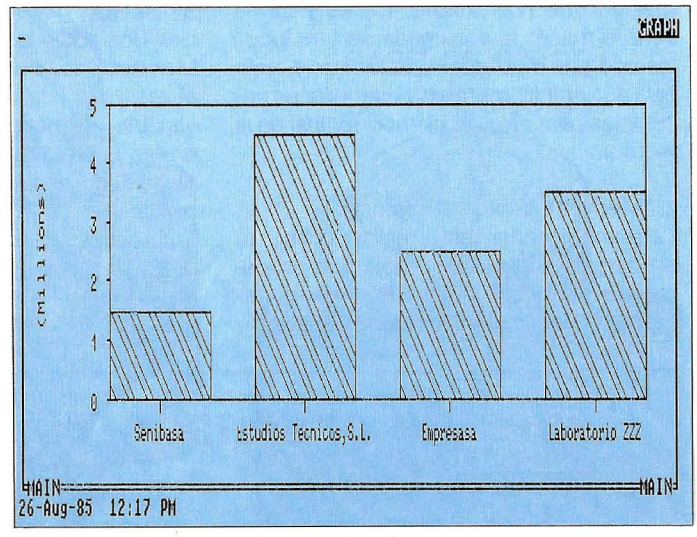
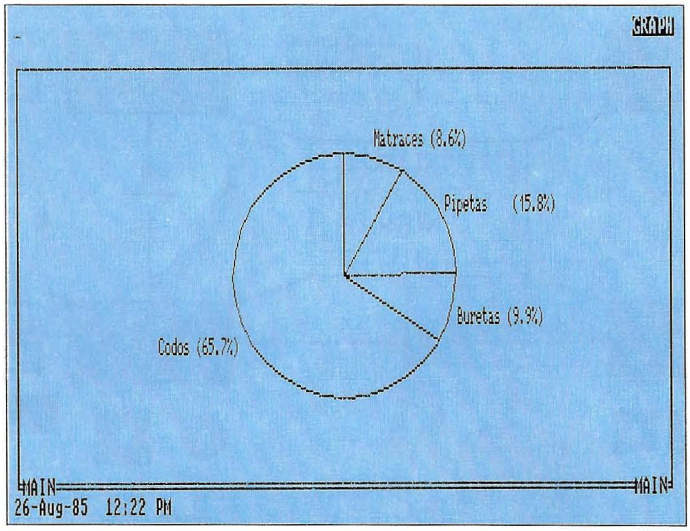


Diagrama de barras con información sobre la facturación a clientes.



En el diagrama de tarta se aprecia el porcentaje correspondiente a la facturación mensual por producto.

rios para producir el diagrama de facturación/producto serán exactamente los mismos, exceptuando que se marcará "tarta" (PIE) como tipo de gráfico; así, los resultados producidos nos darán información sobre los porcentajes de facturación que representa cada producto.

El último paso de la sesión de trabajo consistirá en preparar el informe para dirección. Este proceso es el menos mecanizable dentro de nuestros objetivos; de hecho resulta imprescindible que una persona analice los resultados anteriores para que, en función de ellos y de sus conocimientos, decida qué comentarios debe incluir en el informe. De nuevo, en este caso, se plantean dos alternativas para la producción del informe:

1. A partir de un modelo.

En el caso de que la estructura del informe (capítulos, subcapítulos, etc.) sea igual todos los meses, será necesario entrar en el entorno de tratamiento de textos, recuperar el informe del mes anterior y modificar sobre éste los comentarios del mes actual.

2. Sin modelo fijo.

Si, por el contrario, en función de las incidencias del mes, la estructura del informe es distinta, después de haber entrado en el entorno de tratamiento de textos, no se recuperará el informe del mes anterior sino que se producirá un nuevo informe.

En cualquiera de los dos casos, una vez finalizado el proceso de redacción, se podrán sacar varias copias impresas y almacenar en memoria auxiliar el resultado final.

COMENTARIOS FINALES

Aunque todos los procesos de un mes han sido descritos en la misma sesión de trabajo, su producción se podría haber realizado en diversas sesiones; de hecho, en un caso real, esto sería lo más frecuente.

Tanto en el caso de los gráficos como en cualquier otro, es posible variar las características de los informes de un mes a otro. Así, si se desea, el número de diagramas podría aumentar e incluso se podría obtener la misma información en distintos tipos de gráficos.

Por último es importante destacar que para la producción de algunos informes puede ser muy interesante definir distintas pantallas lógicas; con ello, por ejemplo, en el momento de redactar el informe para dirección, se podrían visualizar simultáneamente los resultados obtenidos en otras fases.