TODO SOBRE SU TIMEX SINCLAIR 1000 Y SU ZX81

editorial mitre

Douglas Hergert

PROGRAMACION DE ORDENADORES EN BASIC
L.R. Carter/E. Huzan

Este es un Curso Elemental para programación de ordenadores en Básic que proporciona las noticias básicas, para que cualquiera que posea o tenga acceso a un ordenador, pueda programar.
También es una introducción general a la programación y un método sencillo y eficaz de introducirse en el complicado mundo del ordenador, tanto personal como empresarial. Es un libro eficacísimo para el pequeño empresario, las tiendas, los profesionales liberales o la persona que desconozca los procesos de programación y quiera introducirse en ellos para establecer, por sí mismo, los programas necesarios sin tener que recurrir a gastos externos.

LOS JOVENES Y EL ORDENADOR
Eugene Galanter Ph.D.

Un manual de ordenadores práctico y fácil de comprender que sirve como introductor a los jóvenes en el campo de los microordenadores, unas máquinas que están transformando el mundo y que se convierten en elementos imprescindibles para el futuro.
El libro le introduce en las técnicas de la programación tanto aplicadas a los estudios, como a los juegos de ordenador, comunicación a través de él, investigación, etc.
A través de este libro, pensado para jóvenes que quieran introducirse en el campo de la informática, tanto los padres como los hijos, pueden aprender y divertirse con las técnicas del futuro.

TODO SOBRE SU TIMEX SINCLAIR 1000 Y SU ZX81

Douglas Hergert

TODO SOBRE SU TIMEX SINCLAIR 1000 Y SU ZX81

Douglas Hergert

editorial mitre

Título original: Your Timex Sinclair 1000 and ZX81
Traducido por: L. Porta

Avda. Sarriá, 137. Barcelona 08017. España
Depósito legal: B. 3.631-1986
ISBN: 84-7652-005-0
Impreso en España
Printed in Spain
Grafic/Cas
C/. Teodoro Llorente, 14. Barcelona

AGRADECIMIENTOS

Las personas que cito a continuación ayudaron a convertir una idea en manuscrito, y, luego, el manuscrito en libro:

James Compton, Joel Kreisman y Rudolph Langer, con su colaboración ténica y de corrección del texto; Bret Rohmer, en la producción; Elaine Foster y Donna Scanlon, en el procesado de palabras; Ingrid Owen, Sharon Leong y Margaret Cusick, en la composición; Valerie Brewster, en el mecanografiado, y Hilda van Genderen y Cheryl Wilcox, en la lectura de las pruebas.

También quisiera enviar un testimonio personal de agradecimiento a todos los empleados de Sybex.

D.H.

INTRODUCCION: AL PENETRAR EN LA ERA DEL ORDENADOR

Usted acaba de llegar a su casa con su nuevo ordenador: el Timex/Sinclair 1.000TM. Y es un momento de gran excitación y entusiasmo; ha entrado por fin, en la era del ordenador. Nunca más sentarse, en la más completa ignorancia, mientras sus amigos hablan de sus ordenadores personales. Ahora va a formar parte del club. Y piensa muy feliz en la escasa cantidad de dinero que ha gastado, comparada con los miles de dólares que otras personas invirtieron en sus ordenadores.

Usted enchufa el ordenador, lo conecta al televisor y observa, con satisfacción que el ordenador responde como debía.

En este punto hay dos perspectivas distintas que pueden describir sus experiencias subsiguientes con el ordenador; la primera, que desemboca en desilusiones y desencantos, resulta completamente insatisfactoria. La segunda, que usted tiene todos los motivos para esperar, es la que le describirá y le hará recorrer el presente libro. Examinemos, pues, las dos perspectivas.

En la primera, usted empieza dando golpecitos al teclado y mirando qué pasa en la pantalla; y al principio le maravilla el poder que parece tener sobre el funcionamiento del ordenador. Pero, poquito a poco, empieza a darse cuenta de que, en realidad, no *entiende* lo que pasa. Cierto, el ordenador hace algo cada vez que usted pulsa el teclado, pero ¿qué hace? Quizá la cosa le cueste más trabajo de lo que se había figurado. No importa —piensa usted—, el manual lo

explicará todo. De manera que abre el libro con la espiral de alambre por lomo que le dieron con el ordenador y se pone a leer. Después de un par de párrafos, empieza a sentirse algo incómodo. Y al cabo de unas cuantas páginas se siente genuinamente perdido. En cierto modo, el libro parece dirigido a personas bastante más enteradas que usted. Se da cuenta de que, en verdad, el libro debe de estar repleto de informaciones útiles para una persona con un poco de experiencia —y lo está, ciertamente— pero, ¿cómo empieza usted? Vuelve al ordenador, pulsa esperanzado unas cuantas teclas más, pensando que en un determinado momento se le abrirá una luz en la mente. Pero ese momento no llega. Su entusiasmo inicial se ha disipado bastante, substituido por un sentimiento de confusión y desencanto. Desenchufa el ordenador, lo guarda en el armario, y sale a segar el césped... Dos o tres meses después, advierte la presencia del ordenador, todavía en el armario con una fina capa de polvo sobre el teclado. Usted ha llegado a contarle a la gente que *es dueño* de un ordenador; pero, ¡ay!, cuando sus amigos empiezan a hacerle preguntas detalladas tiene que arreglárselas para cambiar de tema...

En la perspectiva número dos, usted se enfrenta con el ordenador de una manera más sistemática. Advierte en seguida que tiene que aprender un montón de cosas sobre su nuevo ordenador, y cuantas más aprende, más crece su entusiasmo. Empieza manejando bien el teclado del Timex Sinclair. Aprende a introducir órdenes, y descubre qué significa cada una de ellas para el ordenador. Usted escribe un programa completo en BASIC, el lenguaje de ordenador incorporado al T/S 1.000 y queda profundamente impresionado por los sorprendentes resultados. La experiencia se vuelve más fascinante cada vez que se sienta usted ante el ordenador; y aprende cosas sobre gráficos, cálculos, manejo de series... y antes de que se entere usted mismo se ha convertido ya en un buen programador en BASIC, impresionándose no solamente a usted, sino también a sus familiares y amigos, con todo lo que es capaz de hacer en su ordena-

dor nuevo. Al cabo de unas semanas, se da cuenta de que es algo más que dueño de un ordenador, es —cosa muchísimo más importante— *usuario* de un ordenador.

Este libro, *Su Timex Sinclair 1.000 y ZX81,* fue escrito para garantizar que usted viva y se mueva en la segunda de estas perspectivas. Si usted es una persona completamente ajena al mundo de los ordenadores y se siente confundida por el nuevo vocabulario que todo el mundo parece apropiarse automáticamente, éste es el libro que le conviene. Porque empieza la historia por su mismísimo comienzo y le guía cuidadosamente en todos los pasos que tiene que dar para convertirse en un usuario de ordenador perfectamente satisfecho.

Capítulo 1 —*Reparto de Personajes*— describe la *hardware,* o sea, el equipo material que necesitará usted para sacar el máximo provecho de su ordenador. Además, le da a usted instrucciones cuidadosas, paso a paso, para conectar equipo a su ordenador. Finalmente, el presente capítulo examina el papel que usted tiene que representar para que el ordenador funcione según sus deseos, y las habilidades que adquirirá a medida que utilice el ordenador.

Capítulo 2 —*Acto Primero: Introduzca su programa*— es una introducción a la *software,* las instrucciones y *programas* que uno prepara a fin de guiar al ordenador a través de unas tareas concretas. Empezará aprendiendo, exactamente, a utilizar el teclado para comunicarse con su T/S 1000. Usted introducirá un programa entero, lo pasará y, durante este proceso, adquirirá una primera idea del poder del lenguaje de programación BASIC. Finalmente, aprenderá a *guardar* un programa permanentemente en una cinta de cassette, de modo que pueda utilizarlo multitud de veces. Este capítulo termina con una guía del consumidor para la *adquisición* de software, y una discusión de los pros y los contras de comprar programas en lugar de escribírselos usted mismo.

Capítulo 3 —*El argumento cobra cuerpo*— le inicia a usted en un curso rápido y fácil de BASIC. Usted aprenderá a mandar al ordenador que repita ciertos trabajos una y cien

veces, a tomar decisiones fundadas en información del momento, y a visualizar los resultados en la pantalla del televisor, donde podrá estudiarlos. También se pondrá al corriente de las *dos* clases de realizaciones gráficas del T/S 1000, y, sólo como diversión, verá un programa completo que le permitirá trazar dibujos en la pantalla del televisor.

Capítulo 4 —*Tome cinco*— se dedica por entero a números y cálculos en el ordenador. Descubrirá usted la manera de almacenar grandes cantidades de datos numéricos en su ordenador, adecuada y eficazmente. Verá cómo puede utilizar su ordenador como una "supercalculadora". Finalmente, el programa principal de este capítulo le enseñará a trazar diagramas de barras para toda suerte de información numérica.

Capítulo 5 —*Palabras, palabras, palabras*— describe las características del BASIC diseñadas para el manejo de *series;* es decir, de información ajena a los números. Este capítulo presenta un divertido programa que quizá le ayude a superar esos momentos difíciles de la vida en los que uno, simplemente, no sabe decidir qué debe hacer.

El Apéndice A describe todo el vocabulario del BASIC, es decir, todas las palabras-mandato que aparecen en el teclado. Usted podrá utilizar este apéndice como herramienta de referencia que le ayude a escribir programas.

El Apéndice B resume los códigos de errores del T/S 1000.

El presente libro está estructurado de tal forma que la información ofrecida en un capítulo se utiliza en el siguiente, con lo cual cada capítulo tiene sus bases en los anteriores. Para emplear el presente libro con éxito, debería leerlo y estudiarlo teniendo el ordenador a mano, ensayando cada instrucción nueva en el momento en que se la ofrecen, introduciendo y pasando todos los programas a medida que los encuentra. Si se toma todo el tiempo que necesite para dominar cada peldaño antes de pasar al siguiente, terminará este libro comprendiendo a la perfección las facultades de su ordenador y la mejor manera de servirse de ellas.

Lo más importante que se lleva usted de la tienda junto

con su Timex Sinclair 1000 es el magnífico entusiasmo de un nuevo propietario de ordenador. No lo pierda. Deje que este libro le enseñe lo que necesita saber para sacar todo el provecho de su nuevo ordenador.

Nota para los usuarios del ZX81:

También ustedes pueden emplear este libro como una introducción práctica, activa a su nuevo ordenador. El ZX81 ofrece los mismos mandatos de programación, tiene el mismo teclado y utiliza el mismo equipo adicional que el T/S 1000. La única diferencia considerable entre ambos aparatos radica en la capacidad de sus respectivas memorias. Mientras el T/S 1000 empieza con 2K de memoria, el ZX81 sólo tiene 1K. Unos cuantos programas de este libro, entre los más largos, requieren 2K de memoria; pero si usted quiere reforzar su ZX81 con el módulo opcional de 16K de memoria, podrá pasar también estos programas más largos en su ordenador.

CAPITULO 1
REPARTO DE PERSONAJES

INTRODUCCION

Como usted ya sabe, probablemente, el ordenador Timex Sinclair 1000 —con ser, como es realmente, una máquina asombrosa y poderosa, albergada dentro de su lisa cajita de plástico negro— le sirve de poco a usted por sí solo. Para emplearlo provechosamente debe agregarle otro material. Algunos de estos artículos adicionales son enseres domésticos corrientes: un televisor y un grabador de cintas de cassette. Otros son aparatos especiales diseñados específicamente para trabajar específicamente con el ordenador T/S 1000: por ejemplo, la impresora y el módulo de memoria adicional. La mayoría de estos suplementos participan en dos clases generales de tareas: enviar informacion al ordenador, o recibirla de él. A estas dos tareas se las define con los nombres de *input* y *output,* respectivamente.

En este primer capítulo echaremos un vistazo rápido al ordenador y a parte del equipo —necesario u opcional— que se le puede agregar. A medida que vayamos explorando las tareas de los diferentes componentes, aprenderemos a conectarlos debidamente al ordenador. También hablaremos de los diferentes papeles que usted, usuario del ordenador, puede representar. Al final del presente capítulo, estará usted en disposición de empezar a trabajar con su ordenador.

EL PROTAGONISTA: EL T/S 1000

Eche una mirada al aparato en sí. Vale la pena pasar un

buen rato examinándolo y tomando nota de varios aspectos de su diseño antes de ponerse a trabajar con él. Las figuras 1.1., 1.2. y 1.3. le presentan el ordenador desde ángulos distintos. Examinamos cada una por turno.

En la parte delantera del ordenador está el área del teclado. Por medio de este teclado, usted dará órdenes y suministrará datos a la máquina. De modo que el teclado es el principal instrumento de input del ordenador. Al principio, quizá se resista usted a creer que una lámina plana de plástico pueda actuar realmente como un teclado. Pero pronto descubrirá que las áreas de las teclas son *sensibles al tacto.* Esto significa que, cuando el ordenador está en marcha, una leve presión de un dedo en una tecla hará que el teclado envíe un mensaje al ordenador.

Las 26 letras del alfabeto y las 10 cifras, dispuestas en el mismo orden que en una máquina de escribir, dominan el teclado. Pero junto con las letras y cifras, el teclado también presenta un surtido de palabras, símbolos y diminutos bloques gráficos. Las palabras son los mandatos e instrucciones que puede dar al ordenador para exigirle que haga una determinada cosa. Al leer este libro y empezar a experimentar con el ordenador, usted se enterará del significado de cada una de dichas palabras.

Estas palabras-mandato se incluyen directamente en el teclado por una razón muy sencilla. En lugar de tener que escribir tales mandatos letra por letra, basta con pulsar una sola tecla para enviar el mandato o la instrucción completos al ordenador. Al principio, quizá no le parezca excesivamente conveniente este procedimiento; durante los primeros días, perderá más tiempo buscando las palabras que el que le hubiera costado el escribirlas letra por letra. Pero la práctica le familiarizará con la disposición del teclado, y sus dedos se dirigirán automáticamente hacia las teclas que necesite. Naturalmente, en este teclado no podrá llegar al extremo de escribir con ambas manos con la pericia de un mecanógrafo consumado; pero le sorprenderá lo bien que llega a escribir al cabo de unos cuantos días de usarlo.

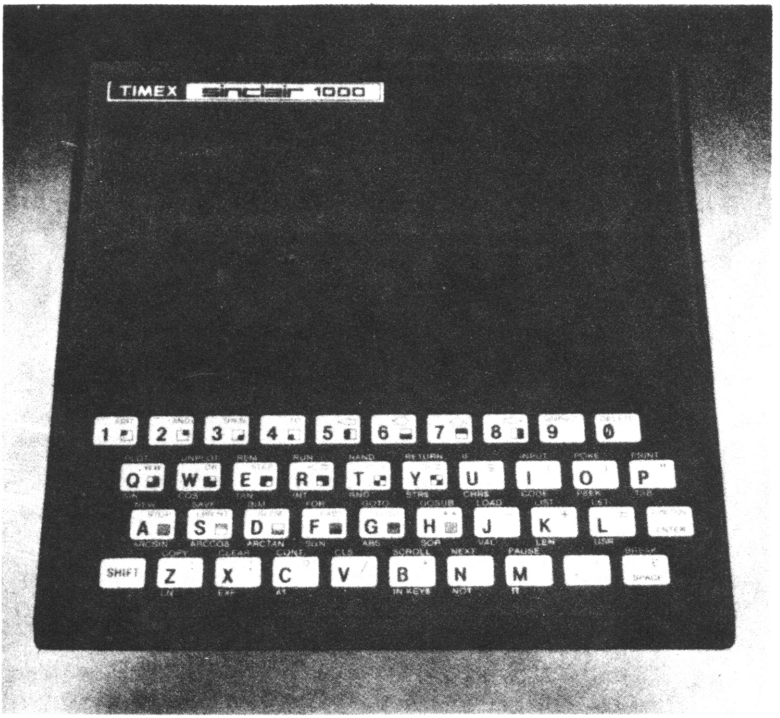

Figura 1.1.: El teclado del ordenador T/S 1000

Como que algunas teclas tienen hasta cinco palabras o símbolos asociados con ellas, se necesita un sistema para "conmutar" ("shifting") el empleo del teclado de una determinada clase de símbolos o palabras a otra. El teclado del T/S 1000 cuenta con diferentes clases de conmutados, y en el Capítulo 2 aprenderemos a manejarlos todos. Lo mismo que las posiciones de las palabras, los métodos de conmutado serán como una segunda naturaleza para usted en un tiempo brevísimo.

Ahora fíjese en el costado izquierdo de su ordenador (que le mostramos en la Figura 1.2) donde encontrará cuatro clavijeros. La función de cada uno está claramente especificada por un rotulito: el primero sirve para conectar el tele-

visor al ordenador; los dos del centro son para el grabador de cintas, y el último, con el rotulito de "9V DC", es para el suministro de corriente.

Junto con el ordenador van las clavijas, los cordones y otro material que necesitará usted para efectuar las conexiones. Hagamos un inventario rápido. Un cordón mide un metro, poco más o menos, y tiene un enchufe en cada lado. Este es para conectar con el TV. Por consiguiente, puede proceder usted a enchufar uno de sus extremos (cualquiera de los dos) al clavijero del ordenador rotulado "RV". Dentro de unos momentos veremos cómo se enchufa el otro extremo al televisor. Digamos de paso que usted no corre el menor peligro al tocar esas clavijas; ni siquiera pasa la corriente.

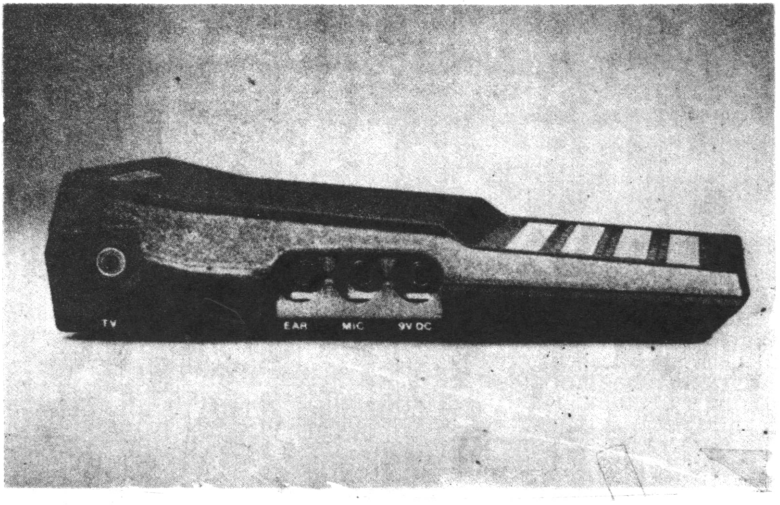

Figura 1.2.: Vista izquierda del ordenador

También debería tener usted un cordón más corto con un par de clavijas en cada extremo. Este es para el cassette. De momento, déjelo aparte (pero no lo pierda); la conexión con la grabadora la examinaremos detalladamente en el Capítulo 2.

Finalmente, encontrará usted un tercer cordón con un adaptador AC (corriente alterna) en un extremo y una clavija en el otro. El adaptador se enchufa a cualquier toma de 120 voltios de la pared (o a un cordón extensor, si es más conveniente) y la clavija pequeña encaja con el clavijero "9V DC" del costado del ordenador. Si quiere ensayar, puede enchufarla en seguida. Una vez enchufado, el ordenador estará en marcha y a punto para funcionar. (Por desgracia, usted no sabrá lo que hace mientras no lo conecte a un televisor). Es buena idea desenchufar el adaptador cuando el ordenador tenga que pasar un largo periodo de tiempo sin ser utilizado.

En la parte trasera del ordenador (que se ve en la Figura 1.3.) hay una abertura de plástico de unas dos pulgadas de largo y media de ancho. Esta abertura deja ver el borde de una tablita delgada a la cual conectará usted la unidad de memoria suplementaria o la impresora, si decide comprar alguna de estas piezas de equipo opcionales. De ambas hablaremos en el presente capítulo. (Si ya posee una, o las

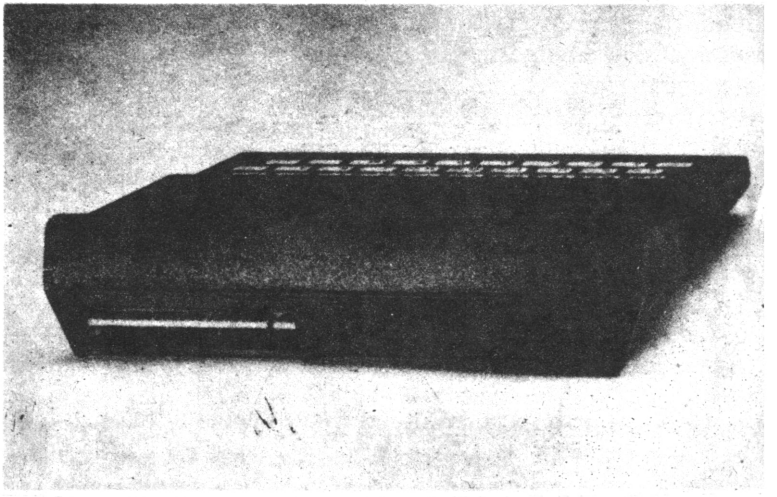

Figura 1.3. La parte trasera del ordenador.

dos, una palabra de advertencia: No las conecte ni las desconecte mientras el ordenador esté enchufado a la corriente).

Ahora que hemos examinado ya la parte exterior del ordenador, ¿qué diremos de la interior, del *contenido* de la caja negra? ¿Qué necesita saber usted del diseño interior del T/S 1000 para convertirse en un consumado usuario de ordenador? Indudablemente, usted ha oído hablar ya y ha leído algo sobre los adelantos en la tecnología de los circuitos integrados ("chips") causantes de la revolución del ordenador personal. También es posible que esté enterado de que al chip "encargado de pensar" de su T/S 1000 lo llaman un microprocesador Z80. Pero, ¿qué cantidad exactamente de esta información necesita dominar a fondo antes de disputar una partida de ajedrez con el ordenador, o de utilizar este aparato para elaborar el presupuesto de la familia?

La respuesta, en caso de que no la imagine, es *ninguna.* Todo lo que necesita saber se encuentra en el exterior: la colección de mandatos grabada en el teclado, y la información que aparece en la pantalla de su televisor. Para todo lo que le afecta a usted, la arquitectura interna de su ordenador se podría resumir tan caprichosamente como lo hizo Ian Frazer en el siguiente pasaje de su cuento corto "The Killion":

> A las personas no familiarizadas con los ordenadores les ayuda un poco, a veces, imaginárselos como unos objetos bastante voluminosos y complicados. El tamaño de los ordenadores varía desde el de un cubito de hielo hasta un parque de atracciones como los Meadowlands de New Jersey, incluida la zona de aparcamiento. Por dentro, el ordenador tendrá un alambre rojo corto conectado a una terminal por un extremo y a otra terminal por el otro extremo. Luego habrá un alambre azul conectado también a sendas terminales por los dos lados; luego un alambre verde; luego un alambre amarillo; luego uno naranja; luego uno color rosa, etc., etc. *

O puede usted dar crédito al cuento del equipo de pulgas

* Reproducido con autorización: c 1982 Ian Frazer. Originalmente en *The New Yorker.*

magníficamente amaestradas... El significado último de todas estas sandeces es el siguiente: Su nuevo ordenador está diseñado de tal manera que usted no tiene por qué preocuparse de su estructura y su funcionamiento internos. (Fíjese en que la cajita de plástico negro que contiene su ordenador no está construida para ser abierta). Como muchos otros aparatos de su casa —desde el teléfono al eliminador de desperdicios—, usted no necesita ser un experto en su tecnología para emplear plena y provechosamente la máquina en cuestión.

Dicho esto, deberíamos observar también que hay una infinidad de libros y revistas que le explicarán todo lo que quiera saber de la tecnología de los procesadores y los ordenadores. A medida que vaya trabajando con su ordenador nuevo y aprenda más y más a utilizarlo, es posible que se sienta interesado por estos temas y quiera leer algo sobre ellos. Aunque no necesita tales conocimientos para servirse del ordenador; usted decide qué quiere investigar y hasta qué punto.

Finalmente, como usuario reciente de un ordenador, quizás le inquiete un poco el peligro de estropear el aparato. Tranquilícese. Pocas cosas puede hacer usted para dañar su T/S 1000. Cierto, de vez en cuando leerá un aviso previniéndole contra esto o aquello. (Y luego están las precauciones dictadas por el sentido común: no deje que el perro le hinque los dientes; no lo use mientras se baña; no lo tire desde la ventana de su apartamento del décimo piso, etc., etc.) Pero disfrutará más su experiencia con el aparato si sabe que con el uso normal no puede estropearlo. Ciertamente, nada de lo que pueda escribir en el teclado perjudicará jamás a la hardware (al material). Podrá cometer errores que, de momento, producirán respuestas incorrectas o pérdidas de datos; pero una vez reconocidos tales errores, lo más probable es que pueda corregirlos. Las equivocaciones forman parte del proceso de aprendizaje; el ordenador siempre perdona.

Hemos contemplado la unidad principal del ordenador; veamos para qué sirven sus complementos.

LOS PAPELES SECUNDARIOS

El televisor

Puede conectar usted cualquier televisor que tenga en casa a su T/S 1000. Lo más conveniente sería un modelo pequeño, portátil; pero puede trabajar con el que tenga, sea de la clase que sea.

El papel del televisor en resumen es decirle todo lo que usted necesita saber sobre las actividades del ordenador. Cuando escriba unas órdenes en el teclado, las verá aparecer en la pantalla antes de ordenar definitivamente al aparato que las cumpla. A menudo escribirá usted una serie de mandatos que querrá dar de una sola vez al ordenador para que los vaya cumpliendo sucesivamente. Una serie así de mandatos es lo que llamamos un *programa.* Antes de indicar al ordenador que cumpla los mandatos de un programa, usted puede visualizar este mismo programa, línea por línea, en la pantalla todo el tiempo que quiera, a fin de estudiarlo y cerciorarse de que está bien. Esta visualización de un programa se llama un *listado.* Si uno encuentra un error en un programa, puede *corregir* la parte equivocada; en la pantalla, aparecerán ante sus ojos los diversos pasos del proceso de corrección. Finalmente, cuando usted *pase* el programa (es decir, cuando mande al ordenador que realice lo que le ordena dicho programa) verá, la mayoría de las veces, los resultados en la pantalla del televisor. Estos resultados pueden ser numéricos, de textos o incluso gráficos, porque el T/S 1000 tiene dos clases de facultades gráficas.

Naturalmente, el ordenador siempre conserva el control de lo visualizado en la pantalla. Cuando usted se haya habituado a leer la información que el ordenador le ofrece en la pantalla, empezará a reconocer algunos sutiles detalles de su manera de funcionar. Por ejemplo, según como esté distribuida la información en la pantalla, usted podrá adivinar cuándo el ordenador se encuentra escaso de espacio para la memoria. Además, el ordenador también le presentará

en la pantalla unos mensajes —que usted deberá aprender a descifrar— advirtiéndole cuando haya incurrido en una equivocación. En la mayoría de los casos, usted sabrá dónde está, exactamente, la equivocación y por qué esta equivocación no le ha gustado al ordenador.

Si el teclado es el medio con que usted cuenta para comunicarse con el ordenador, la pantalla del televisor es el más importante y completo que el ordenador tiene para comunicarse con usted. Al principio quizás le parezca que la pantalla le presenta demasiada información para poder asimilarla, pero pronto se habituará a entender los signos y síntomas más sutiles aparecidos en la pantalla, y será capaz de prestar toda su atención a las informaciones que el ordenador le esté dando.

Además, cuando escriba un programa, tendrá muchas cosas que decir acerca de dónde y cómo sitúe el ordenador la información en la pantalla. Varios de los mandatos que pueda darle al ordenador están directamente relacionados con la aparición y posición de números, textos y gráficos.

Su ordenador conecta con el televisor a través de la cajita plateada que le dieron al comprarlo, al modulador RF (o "caja conmutadora"). De un costado de la caja sale un alambre terminado en un par de conectadores en forma de U y que se conectan al televisor). Mire la parte posterior del aparato de televisión y localice los dos pares de pernos con los rotulitos de UHF y VHF respectivamente. Para el T/S 1000, tendrá que conectar el modulador RF al VHF. Aflojando los dos tornillos, meta los conectores en forma de U debajo, y luego apriételos otra vez.

En la superficie del modulador RF hay un conmutador que se desliza arriba y abajo. En la posición superior se encuentra el rotulito "TV", y en la inferior habrá el de "ordenador" o "juego". Cuando quiera emplear su T/S 1000, deberá asegurarse de que el conmutador esté en la posición inferior. La superior le permite a usted mirar la televisión sin desenchufar el modulador RF.

Finalmente, en un extremo del modulador RF hay un

clavijero similar al de Tv del costado del ordenador. Ahí es donde enchufa usted el largo cordón procedente del ordenador. Hecho esto, la conexión con el televisor es completa.

Usted puede poner el televisor en el canal 2 o en 3. En la cara inferior del ordenador, encontrará un conmutador que se debe colocar de forma que concuerde con el canal de TV que haya escogido usted. En general, suele ser mejor usar el canal que no empleen para las emisiones de televisión en el territorio donde viva usted; pero puede ensayar los dos y elegir el que parezca darle una imagen más clara.

Ahora ponga el televisor en marcha, con el control de volumen puesto al mínimo. Enchufe el adaptador que suministra la energía al ordenador. Inserte la clavijita del otro extremo del cordón del adaptador en el clavijero "9V DC" del ordenador. Si ha hecho bien todas las conexiones, debería ver una K pequeñita en "video inverso" (es decir, en blanco sobre negro) en el ángulo inferior izquierdo de la pantalla, tal como mostramos en la Figura 1.4. Esta K, inicial de "keyword" ("palabra clave") le indica que el ordenador está preparado para aceptar mandatos. Ahora puede ajustar el brillo y contrastar controles del televisor para obtener la mejor imagen posible. (Las posiciones no serán siempre las mismas que le gustan a usted para ver programas de televisión). Si no queda satisfecho de la imagen, ni siquiera después de unos ajustes, pruebe si le va mejor con el otro canal.

Si no consigue imagen alguna, repase todas las conexiones:

- El adaptador de energía debería estar metido en el enchufe de la pared; la clavijita del otro extremo del cordón debería estar metida en el clavijero "9V DC" del costado del ordenador.
- Los conectores en forma de U de la caja de conmutación deberían estar aplicados a los tornillos de VHF (no a los de UHF) de la parte trasera del televisor, y en el interruptor para el uso del ordenador debería estar abierto.

El cordón largo debería estar bien enchufado a la caja de conmutación por un extremo y al clavijero TV del costado del ordenador.

- El canal de TV que haya sintonizado (sea el 2 o el 3) debería corresponder a la situación del conmutador localizado en la parte inferior del ordenador.

Con el televisor unido al ordenador en marcha, usted está preparado para escribir mandatos y someter a prueba el poder del T/S 1000. En realidad, es muy posible que usted decida no conectar ningún otro equipo al ordenador, en sus comienzos. Usted puede aprender muchísimo sobre su recién adquirido aparato sin otra cosa que un televisor que le mues-

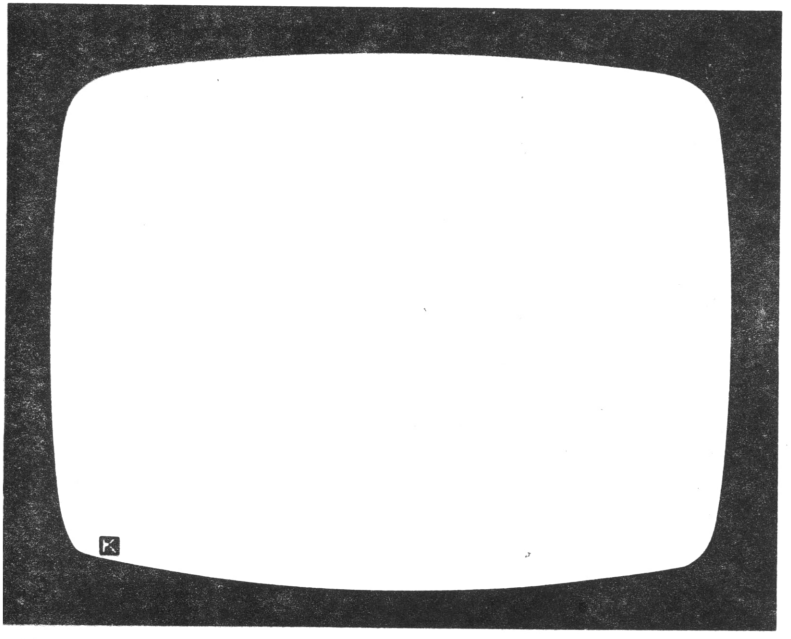

Figura 1.4.: Puesta en marcha del ordenador: La K indicadora

tre lo que está haciendo el ordenador. Sin embargo, con el tiempo, a medida que adquiera más experiencia, probablemente querrá añadir otras facultades a su sistema. Es posible que quiera añadir memoria suplementaria al ordenador, a fin de escribir programas más largos. Quizá también quiera empezar a guardar sus programas de manera permanente... o acaso quiera disponer de una manera cómoda para pasar en su ordenador programas de otras personas. Y, finalmente, es posible que quiera registrar sobre el papel algunos resultados obtenidos por el ordenador. Las secciones siguientes del presente capítulo describen las diferentes piezas de equipo mediante las cuales podrá conseguir todas estas cosas.

Memoria suplementaria

El ordenador posee dos clases de memoria: una es permanente, y la otra, temporal. Ya puede usted familiarizarse con sus nombres, que son: ROM (de Read Only Memory: memoria de leer solamente) y RAM (de Random Access Memory: memoria de acceso al azar) respectivamente. La ROM, o memoria permanente, almacena toda la información que define la "personalidad" del T/S 1000. Esta información determina qué mandatos puede darle usted al ordenador, y cómo los interpretará y cumplirá el aparato. La información guardada en la ROM no se pierde nunca, ni siquiera cuando se le corta la energía al ordenador.

La otra clase de memoria, la RAM, almacena información sobre las actividades que realice el ordenador por el momento. Por ejemplo, contiene los mandatos o el programa que haya escrito usted en el teclado. Sigue la cuenta de las imágenes que el ordenador esté enviando a la pantalla de televisión. También conserva datos que haya presentado usted al ordenador, y almacena los resultados intermedios y finales de los cálculos que usted le encarga al ordenador. Por consiguiente, la RAM es muy dinámica; la información que almacena puede cambiar de un momento a otro. Lo que

más importa recordar de la RAM es esto: cuando uno corta la corriente, toda la información almacenada en la RAM se pierde.

La memoria del ordenador se mide en unidades llamadas *bytes*. Un byte es la cantidad de memoria que el ordenador necesita para almacenar la información que le transmite un golpecito dado a una tecla del teclado. De modo que si usted pulsa uno cualquiera de los mandatos o los caracteres imprimidos en el teclado, el ordenador necesita un byte para registrarlo.

Ahora bien, su T/S 1000 tiene incorporados unos 2.000 bytes de RAM. (En realidad, solemos decir que tiene 2K bytes, expresión en la que la K equivale a 1'024). Usted verá que con esto basta para escribir programas cortos, pero interesantes y útiles, que el ordenador llevará a cabo. Es una memoria más que suficiente para que usted pueda trabajar mientras aprende a utilizar el ordenador.

No obstante, llegará el momento en que usted quizás empiece a desear que el ordenador tenga más memoria. Quizá quiera escribir por sí mismo programas más largos, o quiera pasar programas muy largos escritos por otras personas. Cuando llegue tal momento, podrá omprar un módulo de memoria suplementaria para agregarlo a su ordenador. La Timex produce un módulo de 16K de RAM, y otras compañías producen módulos que le proporcionarán hasta 64K. La cantidad de memoria que necesite usted dependerá del uso que finalmente quiera dar a su ordenador.

Estos módulos se deslizan dentró de la abertura de la parte trasera del ordenador. Si ha comprado ya alguno, puede agregarlo inmediatamente. (Le aviso una vez más de que debe desenchufar la corriente siempre que agregue o desconecte un módulo de memoria). Todos los programas que escribiremos en este libro se adaptarán a 2K de memoria; no obstante, hablaremos de algunos que usted podría comprar y que necesitarían 16K.

Si todavía no ha adquirido ningún módulo suplementario de memoria, quizás debería esperar un poco antes de tomar

una decisión. Como hemos mencionado, los encontrará siempre en el mercado, y con los 2K de RAM incorporados al aparato puede obtener una introducción completa a las posibilidades del ordenador.

Una grabadora

Recuerde que hemos definido un programa como una secuencia de instrucciones que uno presenta de una sola vez al ordenador. Al *pasar* un programa, se le manda al ordenador que cumpla las instrucciones, una tras otra, en una secuencia predeterminada. El otro punto que se debe recordar acerca de los programas que uno escribe es el siguiente: se almacenan en la RAM, o sea, en la memoria temporal. Esto significa que si usted está trabajando con un programa y de pronto desconecta el ordenador del suministro de energía —accidental o intencionadamente— el programa se le habrá perdido.

Pero el ordenador le brinda una bonita solución a este problema. Usted puede almacenar programas grabándolos directamente desde el ordenador a una cassette. Más tarde, cuando quiera pasar el programa de nuevo, puede pasar la cinta y el ordenador almacenará nuevamente su contenido en la RAM. Claro está, semejante proceso requiere unas conexiones bastante delicadas entre el grabador de cintas y el ordenador; en el Capítulo 2 hablaremos de ellas. Pero el grabar programas no exige un equipo caro ni sofisticado. Si tiene en casa algún grabador de cintas, lo más probable es que le sirva para su ordenador. Si no lo tiene, sin duda querrá comprarlo algún día; pero debería decidirse por uno barato.

La impresora

Otra pieza de equipo cuya compra puede usted tomar en consideración es la impresora diseñada para incorporarla al T/S 1000. Lo mismo que el módulo de memoria suplemen-

tario, la impresora se conecta a la tablilla que queda al descubierto en la parte trasera del ordenador, y también deberá desenchufar el aparato antes de conectarla. La impresora utiliza rollos de un papel especial; los documentos que produce, aunque legibles, son solamente de calidad mediana. A pesar de lo cual, la facultad de producir un recuerdo impreso del trabajo del ordenador puede ser con frecuencia un bien muy valioso.

El ordenador posee varios mandos que le permiten a usted el controlar la impresora desde el teclado, directamente. Usted puede, por ejemplo, mandarle al ordenador que liste un programa con la impresora o que copie sobre el papel el contenido de la pantalla del televisor. También puede escribir programas que envíen sus resultados a la impresora en lugar de enviarlos a la pantalla. Si decide comprar una impresora, descubrirá que es extremadamente fácil utilizarla para producir toda suerte de registros de su ordenador. (El Apéndice A describe los mandos que empleará para hacer funcionar la impresora: COPY, LLIST, y LPRINT).

EL PRODUCTOR, DIRECTOR Y REDACTOR: USTED

Luego fijamos nuestra atención en *usted,* y en los diferentes papeles que puede representar al poner en marcha su ordenador.

Hemos visto que se trata de un aparato diseñado para aceptar y cumplir ciertos mandatos, que usted puede comunicarle pulsando las teclas adecuadas. Si echa una mirada al teclado, verá que hay una docena, aproximadamente, de palabras que el ordenador entiende. Usted puede imaginarse estas palabras —junto con las reglas específicas para utilizarlas— como una especie de idioma: un lenguaje de *ordenador.* Este lenguaje de ordenador es mucho más sencillo que un lenguaje humano; posee un vocabulario muy limitado y una gramática sencilla y regular. Por esta razón, el aprenderlo le será mucho más fácil y cómodo que el aprender francés o italiano. No le costará ningún trabajo recordarlos en

cuanto haya aprendido sus nuevos significados en el contexto de las instrucciones al ordenador acerca de lo que debe hacer.

Hay muchos lenguajes de ordenador distintos. El incorporado al T/S 1000 —y, en verdad, a la mayoría de ordenadores pequeños— es el llamado BASIC (acrónimo de Beginner's All-purpose Symbolic Instruction Code). En estos próximos años, usted oirá hablar más y más del BASIC. Siendo como es un lenguaje tan sencillo, la gente de todas partes está aprendiendo a emplearlo para resolver toda clase de problemas con los ordenadores. Hasta los niños de las escuelas primarias aprenden a escribir programas en BASIC. Dentro de poco, el programar en BASIC se convertirá en una habilidad tan corriente en la casa y la oficina como, por ejemplo, el escribir a máquina o el utilizar una calculadora de bolsillo.

La popularidad del BASIC aumenta precisamente porque para utilizarlo no es necesario ser un experto en ordenadores. Y usted se quedará pasmado cuando lo aplique en su ordenador y vea el sinfín de posibilidades que encierra. En realidad, este lenguaje sencillo tiene mucho en común con otros que los programadores profesionales de ordenadores emplean en aparatos mucho mayores que el T/S 1000.

El primer papel que representará usted con su ordenador nuevo será, pues, el de programador de BASIC. Cuando se le ocurra una tarea que quiera encomendar al ordenador, se encontrará automáticamente distribuyéndola de la manera siguiente:

1. Dividirá la tarea en una serie de pasos elementales que el ordenador pueda realizar uno después de otro. Estos pasos pueden incluir cálculos, tareas repetitivas, y hasta decisiones que usted querrá que el ordenador tome basándose en los datos que tenga a su disposición.
2. Determinará la manera más eficaz de expresar estos pasos como una serie de instrucciones en BASIC. Si el programa ha de ser largo, con muchas instrucciones, usted querrá empezar, probablemente, anotándoselo en un papel, más

bien que escribiéndolo directa y extemporáneamente en el teclado del ordenador.

3. Una vez elaborado un programa, lo introducirá, línea por línea, en el ordenador mediante el teclado. Luego lo ensayará, para ver si realiza verdaderamente la tarea que le había asignado. Si el programa ha de hacer calculos con datos numéricos, acaso lo ensaye usted con varias colecciones de datos de prueba antes de quedar convencido de su bondad.
4. Si los resultados de su programa no son correctos, repasará usted las instrucciones en BASIC que escribió y probará de adivinar dónde se equivocó. Muy a menudo, la misma naturaleza de los resultados desacertados le indicará inmediatamente lo que esté mal; en otras ocasiones, los errores quizás resulten más sutiles ¡y usted meditará durante horas —o hasta días— antes de resolver el rompecabezas de su programa!

El primer programa de BASIC lo verá usted en el Capítulo 2. Lo cierto es que en este primer programa se concentrará más en el empleo del teclado del T/S 1000 que en el significado de las instrucciones en BASIC, pero muy pronto redactará usted sus propios programas. Advierta que el proceso de elaborar un programa, siguiendo los cuatro pasos esbozados antes, incluye un curso de entrenamiento en la detección y corrección de errores. Lo cual no debe inquietarle. El T/S 1000 identificará algunos de los errores en que usted incurra mientras vaya escribiendo el programa. Otras equivocaciones se pondrán de manifiesto cuando lo pase. Veremos que el ordenador tiene varias maneras de ayudarle a descubrir y corregir equivocaciones de sus programas.

Cuando compre el ordenador, se fijará seguramente en que también puede adquirir unos programas —escritos por otras personas— ideados para ser utilizados en un T/S 1000. Dichos programas están almacenados en cassettes, y son, todos, bastante caros. Pero ésta es otra manera de utilizar su ordenador: pasar en él programas de otras personas para

realizar tareas específicas. En el Capítulo 2 hablaremos de las ventajas y desventajas de comprar programas (como política opuesta a la de escribírselos usted mismo). En estos momentos, usted quizás se esté formulando la siguiente pregunta: ¿Por qué habría de tomarse uno la molestia de aprender a programar en BASIC si ya encuentra en el mercado programas escritos y no tiene que hacer otra cosa que pasarlos?

En verdad, muchas personas que compran el ordenador T/S 1000 quizás nunca aprendan a escribir un programa en BASIC. Se limitarán siempre a escoger y comprar programas en los comercios, según las necesidades concretas que tengan. He ahí un empleo muy legítimo del ordenador; nadie *tiene que* aprender a programar para poder utilizarlo provechosamente. Sin embargo, si usted no prueba de aprender el BASIC y escribir sus propios programas, perderá una magnífica oportunidad de instruirse por sí mismo. Si aprende a elaborar sus propios programas, el ordenador será para usted una herramienta valiosa que podrá emplear siempre para sus necesidades particulares en este campo. Por añadidura, el proceso de aprender a programar un ordenador le dará una visión nueva y más penetrante de un instrumento cuya importancia para nuestra sociedad crece todos los días.

RESUMEN

Para comunicarnos con el ordenador, hemos de conectarle equipo de input y de output. En el T/S 1000 el ingenio de input más importante es el teclado que lleva incorporado. Este teclado exhibe de manera adecuada todos los mandatos que puede darle usted al ordenador, de forma que cada palabra puede introducirse pulsando una sola tecla.

La pantalla del televisor es el mayor instrumento de output, y suministra gran parte de la información sobre los programas, sus resultados y la acción del momento del ordenador. Además, uno puede agregar una impresora al ordenador, si quiere unos registros de output más permanentes.

También se puede conectar al ordenador una sencilla grabadora de cintas de cassette, si uno quiere guardar programas que haya escrito, o cuando quiera pasar programas adquiridos. Finalmente, disponemos de módulos de memoria suplementaria para aumentar (temporalmente) la capacidad de memoria RAM del ordenador.

Las palabras que usted ve en el teclado componen el vocabulario del lenguaje BASIC. Aprender a programar en BASIC, si bien no es absolutamente esencial para utilizar el ordenador, logrará, probablemente, que su experiencia con este aparatito resulte más eficiente y agradable.

CAPITULO 2
ACTO PRIMERO:
INTRODUZCA SU PROGRAMA

INTRODUCCION

Si usted ha seguido fielmente las instrucciones detalladas en el Capítulo 1, su ordenador está en marcha, el aparato de televisión conectado, y usted está a punto de empezar el trabajo. La primera tarea que le espera es la de aprender a utilizar el teclado del ordenador; he ahí la meta principal de este capítulo. Empezaremos mirando algo más detenidamente el lenguaje BASIC. Veremos qué aspecto tiene un programa, y hablaremos, en términos generales, de cómo se estructura. Luego usted introducirá un programa corto en su ordenador. Durante este proceso, verá cómo le ayuda la máquina a descubrir y corregir los errores que pueda cometer.

A continuación, verá la manera de emplear una grabadora con el ordenador. Observaremos con esmerado detalle las conexiones que tiene que efectuar usted para guardar programas en una cassette, y para cargar en el ordenador un programa guardado en una cinta. Usted ensayará guardando el programa que haya introducido en el ordenador.

Esto nos llevará de la mano hasta una tercera cuestión: los programas que puede comprar en forma de cassettes. Examinaremos un ejemplo, y luego terminaremos el capítulo con una guía del consumidor en la compra de cassettes para su T/S 1000.

EL ORDENADOR APRENDE LA LECCION

Naturalmente, todo buen ordenador debe aprender a ser

amable, de manera que el primer programa que examinaremos le guiará de forma que realice las siguientes tareas:

1. Preguntar cómo se llama usted.
2. Trazar un diseño en la pantalla del televisor.
3. Imprimir un saludo en la pantalla.

Usted quizá considere frívolas tales tareas para un programa de ordenador; pero de momento, la utilidad no es lo que nos interesa primordialmente. Lo que pretende este programa es introducirle a usted plenamente en el empleo del teclado. Al terminar de escribir el programa, habrá visto ejemplos de todos los servicios que el teclado puede prestarle.

El lenguaje de ordenador BASIC

El programa de "saludo" lo mostramos en la Figura 2.1. Tómese un momento para estudiarlo antes de ponerse a escribirlo en el teclado. Puede aprender muchísimo sobre el lenguaje BASIC con sólo mirar un programa escrito en él.

En el Capítulo 1 hemos aprendido que el BASIC tiene un vocabulario muy limitado, compuesto principalmente de palabras familiares en inglés. Ciertamente, el programa de saludo contiene unas cuantas palabras, pocas, que parecen escritas en una especie de taquigrafía abreviada (CLS, LEN, CHR$); pero la mayoría de las otras son sobradamente conocidas. Esto no significa que usted pueda entender inmediatamente qué frutos dará cada una de las líneas del programa; pero al menos puede ver que, al fin y al cabo, el programa no parece intimidar en exceso.

¿Qué nos dice, pues, este programa sobre las características del BASIC? Primero, que un programa en BASIC está estructurado en líneas, y cada línea está numerada. Este programa tiene 13 líneas. La longitud de un programa puede variar desde una sola línea hasta centenares, dependiendo de la tarea que esté destinado a realizar.

Las líneas del presente programa están numeradas desde

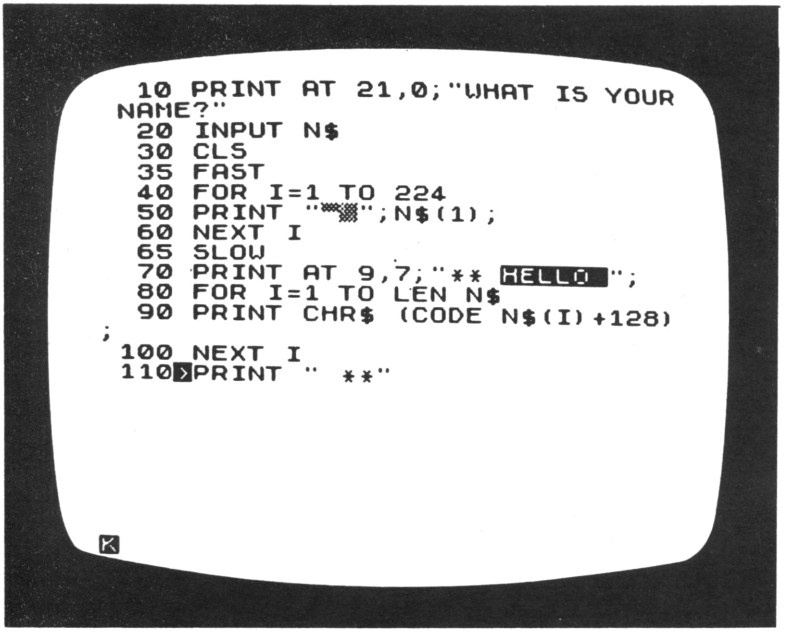

Figura 2.1.: El programa de saludo

10 hasta 110; pero advierta que los intervalos no son uniformes. La mayoría de números salen de sumar 10 al anterior; pero algunos se han formado sumándole cinco solamente. Cuando uno escribe un programa en BASIC es libre de numerar las líneas de la manera que se le antoje. Para el ordenador, el único significado de los números de línea es que le señalan el orden en que debe cumplir las instrucciones del programa. Este programa lo habríamos podido numerar desde 1 a 13, ó desde 100 a 1300, y el ordenador lo hubiera aceptado lo mismo que ahora. La ventaja de numerar las líneas a intervalos regulares, de 10 en 10, por ejemplo, radica en que muchas veces, después de haber escrito un programa, uno quiere añadirle líneas —ideas de última hora— después

del principio y antes del final. Lo cual resulta facilísimo, siempre que el sistema de numerar haya dejado "espacio" para líneas adicionales. Por ejemplo, en el programa que ofrecemos aquí, las líneas 35 y 65 fueron añadidas después de haber escrito lo demás.

Se da el caso de que dos de las líneas numeradas de este programa —la 10 y la 90— ocupan más de una línea en la pantalla del televisor. El ordenador estructura la información de tal manera que una *línea de la pantalla* puede constar de 32 caracteres, y en cambio una línea en BASIC puede ser mucho más larga. Cuando una línea del programa sobrepasa los 32 caracteres, sencillamente, se continúa en la siguiente línea de la pantalla. Esto puede ocasionar algunos cortes inusitados, extraños en las líneas de un programa —por ejemplo, el último carácter de la línea 90, un punto y coma, ha tenido que pasar en solitario a la línea siguiente de la pantalla— pero no afecta en modo alguno a la lectura que el ordenador hace de la línea ni a su manera de ejecutarla.

La primera palabra de todas las líneas de un programa en BASIC para el T/S 1000 es siempre una *palabra clave (keyword).* Las palabras clave le dicen al ordenador qué clase de instrucción se le va a dar. A cada palabra clave debe seguirla su "gramática" particular; es decir, el ordenador sabe qué clase de pautas de instrucción ha de esperar después de cada palabra clave. En el teclado, hay 26 palabras clave, situadas, una por una, sobre las teclas correspondientes a las letras. La Figura 2.2. muestra el emplazamiento de las palabras clave.

Cuando uno escribe una línea numerada de un programa BASIC en el teclado del ordenador. el aparato almacena la línea en su memoria y la reproduce en la pantalla para que podamos verla. Pero *no* empieza a cumplir las instrucciones del programa hasta que se lo ordenamos. Simplemente, registra el programa y espera que uno le diga qué debe hacer luego. Veremos cómo funciona este procedimiento cuando hayamos descrito la manera de introducir el programa.

Figura 2.2.: Las palabras clave

El teclado. Palabras clave y modos

Por el momento, la pantalla de su televisor está en blanco, si exceptuamos la K en vídeo inverso del ángulo izquierdo inferior. Usted recordará que la K significa "keyword", o palabra clave. Esta *pista,* a la que su manual del usuario denomina *cursor*), es la manera que tiene el ordenador de decirle que está preparado para aceptar cualquier palabra clave del teclado. Cuando empiece a escribir el programa, verá otras pistas —entre ellas L, G, F y S, todas en vídeo inverso—. Estas pistas le explican a usted en qué modo —o modalidad— está el teclado; es decir, qué clase de palabras o símbolos puede introducir a continuación en el ordenador.

Nos ocuparemos de cada línea de este programa por pulsación; usted debería escribir las líneas una por una a medida que vaya leyendo las descripciones. (También leerá una explicación breve, general de lo que *hace* cada línea; pero, de momento, concéntrese en el teclado, no en el significado de las líneas. Más tarde tendrá tiempo de sobras para dominar bien el BASIC, cuando haya aprendido a utilizar el teclado).

Entrando el programa, línea por línea

La primera línea del programa es el mandato de PRINT. PRINT le manda al ordenador que exhiba cierta información en la pantalla del televisor. En esta primera línea, a PRINT le sigue la palabra AT (EN), que es una de las varias maneras posibles de especificar exactamente en qué *lugar* de la pantalla quiere usted que aparezca dicha información. En el caso presente, tal "información" es una pregunta, WHAT IS YOUR NAME? (¿COMO SE LLAMA USTED?, aunque literalmente sería ¿CUAL ES SU NOMBRE?) que aparece entre comillas al final de la instrucción de PRINT.

Escribamos pues la línea. Si no es la primera vez que pulsa usted las teclas de su ordenador, habrá observado que

no necesitan mucha presión para marcar; no se precisa más que un ligero contacto. De todos modos, convendrá que se fije en la pantalla mientras escribe, para asegurarse de si la tecla que ha tocado ha enviado de verdad su mensaje al ordenador.

Como hemos visto, lo primero que debe marcar usted al empezar cada una de las líneas del programa en BASIC es su número correspondiente. Para la primera línea, pulse los dígitos 1 y 0. A medida que introduzca cada carácter, verá que la pista K se desplaza un lugar hacia la derecha, y las cifras aparecen a su izquierda.

A continuación, busque el mandato PRINT. Está situado sobre la tecla [P] (en el extremo derecho de la segunda fila, contando desde arriba). Pulse la tecla y vea qué sucede en la pantalla. La palabra PRINT aparece de pronto en la línea. Además, la pista de vídeo inverso se transforma en una L. Esto le dice a usted que ha dejado el modo de las palabras clave y ahora está en la modalidad de las letras; la tecla que pulse ahora registrará una letra o una cifra, no una palabra clave.

De modo que para introducir una palabra clave basta con pulsar una sola tecla, aquella que la contiene. En realidad, *no se pueden* escribir las letras de una palabra clave. Si usted se entretuviera pulsando las letras P—R—I—N—T, el ordenador no las reconocería como la palabra clave PRINT. Siempre que en la gramática especial del BASIC sea correcto el introducir una palabra clave, el ordenador se lo avisa a usted exhibiendo la pista [K]. Si no ve usted el vídeo inverso K, ya sabe que no puede introducir una palabra clave.

La palabra siguiente de la primera línea es AT. Esta palabra representa una de las 25 funciones disponibles en su teclado. Las funciones aparecen debajo de todas las teclas de letras (exceptuada la V). Las mostramos en la Figura 2.3. Algunas de estas funciones son genuinamente matemáticas, tales como seno (SIN), coseno (COS) y el logaritno natural (LN). Si usted pretende utilizar el ordenador para cuestiones científicas o matemáticas, no cabe duda, estará ya al

corriente de tales funciones. Otras de las contenidas en el teclado no tienen nada de matemáticas. Puede usted considerarlas funciones programadoras, porque cuando esté familiarizado con su empleo, le simplificarán muchas tareas de programación. AT, por ejemplo, es una "función" que le permite especificar, de modo muy conveniente, el emplazamiento de la información en la superficie de la pantalla.

Todas las funciones —matemáticas o de otra especie— tienen una cosa en común. Para escribirlas, es preciso "desplazar" ("shift") el teclado hacia el modo de la función. Para verificar este desplazamiento (o conmutado) aprieta usted dos teclas a la vez:

- La tecla de SHIFT, situada en el ángulo inferior izquierdo del teclado, y
- La tecla FUNCTION/ENTER, situada en el costado derecho del teclado, inmediatamente debajo de la tecla de la [P].

Primero sitúe un dedo sobre la tecla SHIFT; luego toque la tecla FUNCTION, y vea qué pasa en la pantalla. La pista en vídeo inverso deja de ser una L y se transforma en una F. Ahora se encuentra usted en el modo de la función. Puede apretar cualquiera de las teclas que tienen funciones debajo, y la función aparecerá en la pantalla, formando parte de la línea del BASIC.

Busque la función AT en el teclado. Está debajo de la tecla [C]. Púlsela, y AT se integra en la línea. Una línea que ahora dice:

```
10 PRINT AT
```

Advierta que la pista en vídeo inverso ha cambiado de nuevo, ha vuelto a ser la L.

Lo mismo que las palabras clave, las funciones se han de introducir con una sola presión de la tecla correspondiente. Para el ordenador, las letras A-T escritas así, una después de

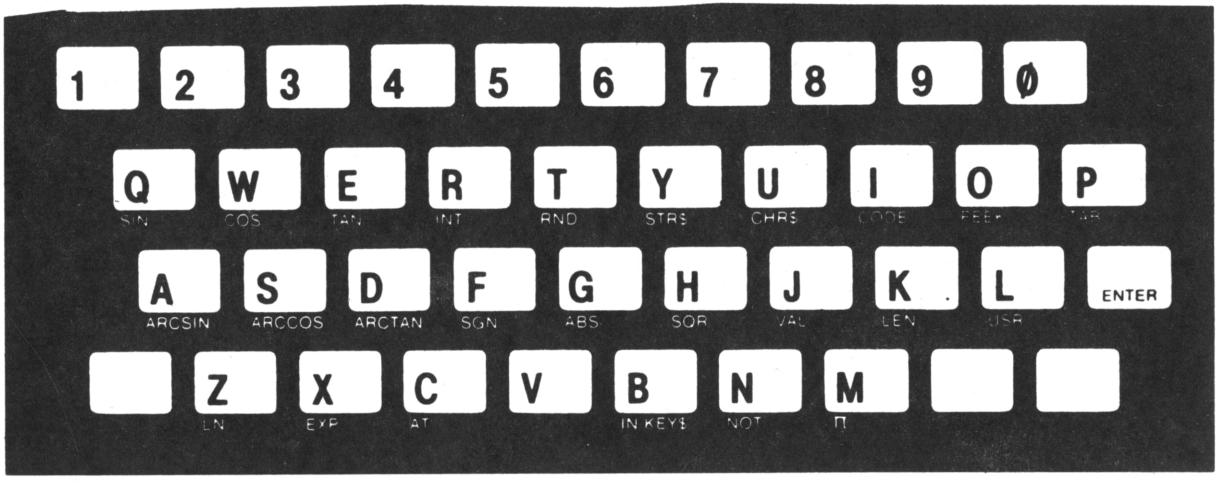

Figura 2.3.: Las funciones

la otra, no significarían lo mismo que el pulsar la tecla AT en modo de función.

A continuación hemos de escribir dos números en la línea. Como es posible que se haya figurado ya, estos números representan la "dirección" del lugar de la pantalla en donde el ordenador imprimirá el mensaje. (Los números 21,0 significan fila 21 y columna 0. Y veremos exactamente cómo funciona eso cuando estudiemos la función AT, en el Capítulo 3).

Antes de seguir con la escritura en el teclado, hemos de averiguar qué se debe hacer cuando uno se equivoca al pulsar las teclas. Tales errores son inevitables; quizás usted ya haya cometido uno, y se pregunte, un tanto confuso, qué debe hacer ahora.

El ordenador le proporciona una manera fácil de "borrar" todo lo que escriba en una línea. Una vez más, tendrá que apretar dos teclas a la vez:

- la tecla de SHIFT, y
- la de [0] (cero).

Debería fijarse en dos cosas acerca de la tecla [0], situada en el ángulo superior derecho del teclado. Primera, distinguir entre la cifra 0 y la letra O; la cifra cero está cruzada por una raya. (Dicha raya también aparece en la pantalla). Segundo, la tecla cero tiene escrita en rojo la palabra DELETE ("borrar") sobre la cifra. De manera que, apretando SHIFT y DELETE conjuntamente, usted puede "borrar" cualesquiera errores que cometa al escribir.

Pruébelo ahora, una vez. Escriba unas letras que no pertenezcan a su línea. Luego apriete la tecla SHIFT con un dedo y toque la DELETE con otro. Fíjese en lo que sucede en la pantalla. La letra L retrocede hasta la letra que estorbaba, y usted puede continuar escribiendo.

Ahora escriba los dos números, empezando por el 21. Al 21 ha de seguirle una coma (,) y al 0 un punto y coma (;). Esta puntuación, aunque quizás no la entienda todavía, es

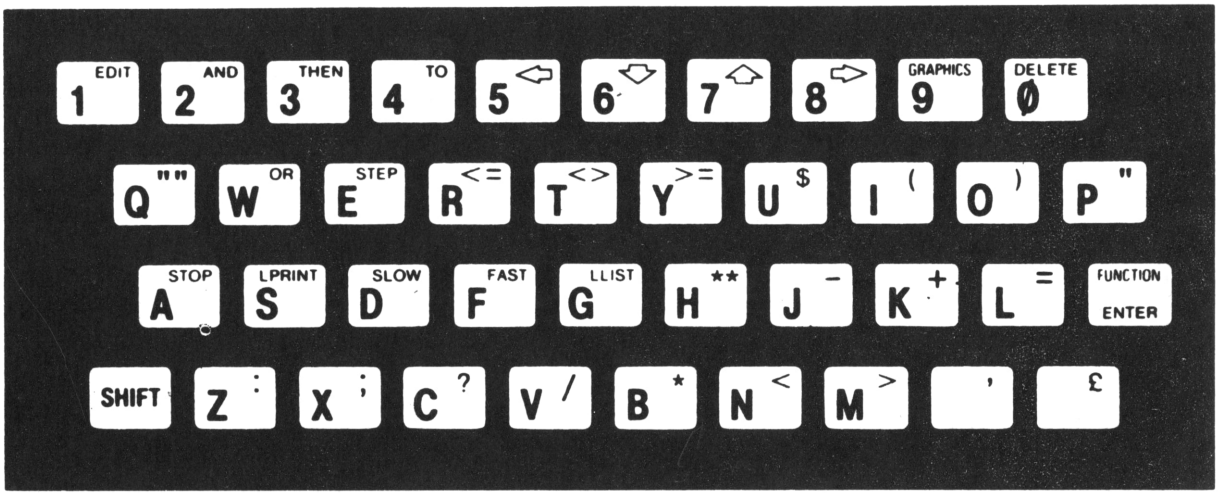

Figura 2.4.: los caracteres shift

una parte esencial del significado de la línea, de modo que es preciso escribirla correctamente. La coma está situada en el área inferior derecha del teclado, en la misma tecla que el punto (.). El punto y coma está situado en la tecla [X]. Tanto la coma como el punto y coma aparecen en rojo en el teclado.

La verdad es que cada tecla del teclado tiene un carácter o una palabra grabados en rojo. Los denominaremos caracteres "shift" (o sea, de "desplazamiento", o de "conmutado"). Para escribirlos, es preciso apretar primero la tecla de SHIFT, y luego, con otro dedo, tocar la tecla correspondiente. En la figura 2.4. mostramos todos los caracteres shift.

Pruebe de escribir la coma. Si primero le saliera un punto, en lugar de una coma, ello significaría que no aprieta la tecla con bastante fuerza (o no la aprieta en el lugar adecuado, exactamente). Borre el punto (SHIFT-DELETE) y pruebe otra vez. Luego escriba el cero y el punto y coma, para completar la dirección. La línea de su pantalla debería quedar así:

```
10 PRINT AT 21,0;
```

El resto de la línea es el verdadero mensaje que se imprimirá en la pantalla cuando el ordenador ejecute la instrucción de PRINT. El mensaje está entre comillas. El carácter comilla está en la tecla [P]; y es un carácter shift. Si aprieta SHIFT-P; la comilla aparecerá en la pantalla. Ahora escriba el mensaje: WHAT IS YOUR NAME? (¿COMO SE LLAMA) letra por letra. (La tecla espaciadora está en el ángulo inferior derecho del teclado). Cuando escriba la letra N de NAME advertirá que ya no le queda espacio en la línea de la pantalla. Cuando escriba la letra A, el ordenador pasa automáticamente a la línea siguiente. El mensaje termina con un signo de interrogación, que es un carácter shift (de desplazamiento) localizado en la tecla [C].

Antes de que escriba usted el interrogante final, vamos

a experimentar con otra característica especial de su ordenador. Normalmente, usted terminaría de escribir la línea y luego apretaría la tecla ENTER para decirle al ordenador que la línea está completa. Sin embargo ahora, para ver qué sucede cuando uno se equivoca apriete la ENTER *antes* de escribir el signo de interrogación final.

Detección automática de errores

Lo que verá usted en la pantalla después de apretar ENTER se lo mostramos en la figura 2.5. A continuación de la pista L ha aparecido otra. Una S, que significa *error de sintaxis*. La palabra *sintaxis* se refiere a las reglas de la "gramática" especial que tiene usted que seguir siempre que escriba un programa en BASIC. Cuando escriba una línea que viole una de tales reglas gramaticales, el ordenador lo advierte al momento, y se niega a dar entrada a la línea como parte del programa. La pista S es la manera amable que tiene el ordenador de decirle que usted se ha equivocado en algo, y tiene que fijarse más en la línea en cuestión.

Por supuesto, en este caso, como usted ha cometido el error intencionadamente, ya sabe que está mal. Le falta el interrogante final. Bien, cuando escribe el signo de interrogación (SHIFT-P) la pista S desaparece, y la línea ha quedado completa. Apriete de nuevo la tecla ENTER y vea qué pasa. La Figura 2.6. muestra los resultados. La línea bien escrita asciende hasta la parte superior de la pantalla. Esto significa que el ordenador la ha almacenado en su memoria como la primera (y en este caso la única) línea de un programa. La pista K ha reaparecido en el ángulo inferior izquierdo de la pantalla, anunciándole que ahora el ordenador está preparado para las instrucciones siguientes.

La segunda línea: Empleo del corrector

Escribamos ahora la segunda línea del programa, la 20:

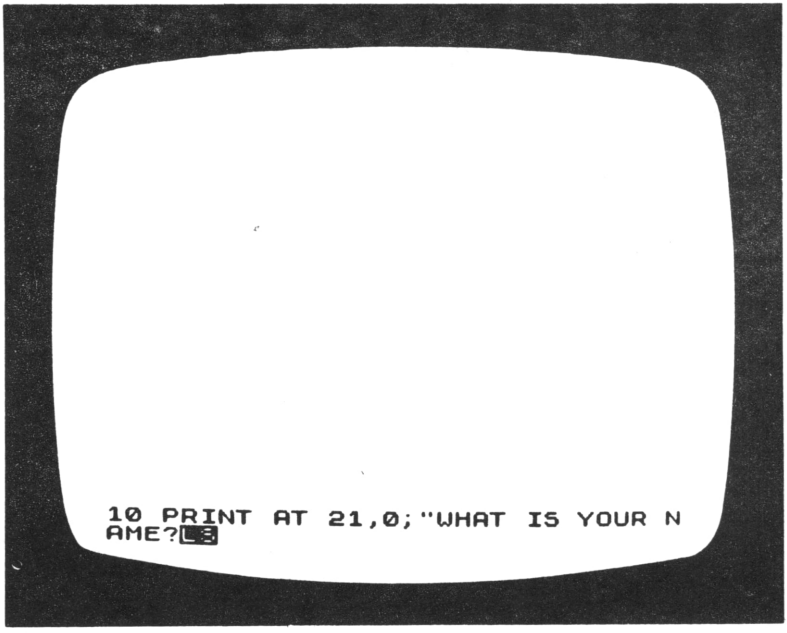

Figura 2.5.: El ordenador responde a un error de sintaxis

```
20 INPUT N$
```

La palabra clave INPUT manda al ordenador que interrumpa la realización de un programa, temporalmente, y espera que usted escriba alguna información en el teclado. Utilizando el mandato de INPUT en el programa, se puede crear una especie de *diálogo* entre el ordenador y la persona que lo utiliza. Más adelante, usted verá cómo funciona, exactamente, este proceso, cuando termine de escribir este programa y luego lo pase.

Los caracteres N$ después de la palabra clave INPUT representan lo que nosotros llamamos un *nombre variable.* Pero no se deje intimidar por esta terminología. Una variable es simplemente un lugar que el ordenador reserva en su memoria para un tipo de información especificado. Se pue-

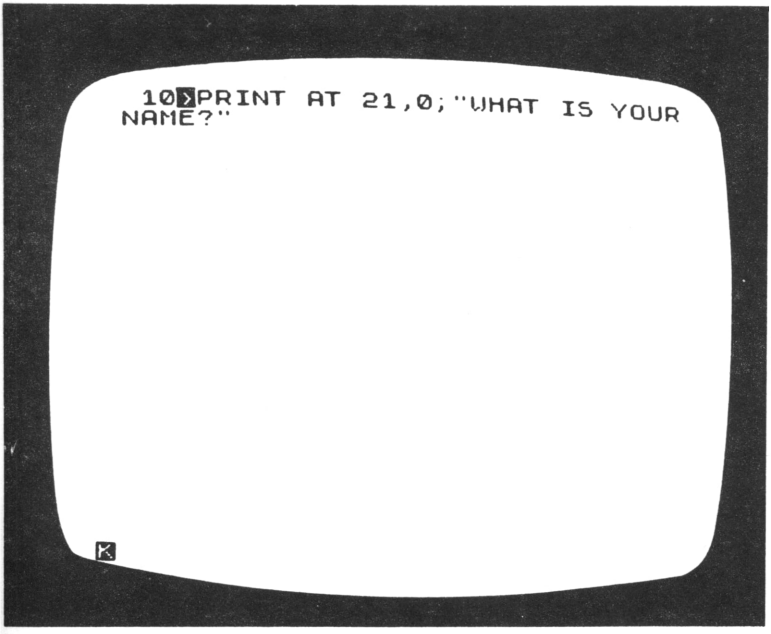

Figura 2.6.: Entrando acertadamente una línea de programa

den definir muchas variables para emplearlas en un programa, pero es preciso dar nombre a cada una, y de esta manera el ordenador puede identificarlas por sus nombres. Los datos contenidos en una variable pueden cambiar muchas veces en el transcurso de un programa —por esto la llamamos variable— pero el *nombre* no cambia nunca.

Cuando el ordenador cumple el mandato de INPUT N$, espera primero que le introduzcan alguna clase de datos mediante el teclado. Cuando llegan los datos, el ordenador los almacena en una variable a la que denomina N$. En lo sucesivo, siempre que usted aluda al nombre N$ el ordenador sabrá que le interesan los datos almacenados en esta variable.

Se pueden definir dos clases de variables para utilizarlas en un programa: variables numéricas, y variables para texto

(no numéricas). En lenguaje de ordenadores, a un renglón de datos no numéricos lo llamamos una serie, o cadena (string). Para definir una variable en serie, es necesario especificar *con el nombre de la variable* que queremos que el ordenador reserve espacio de memoria para una serie. Esto se hace añadiendo el símbolo del dólar ($) al final del nombre de la variable.

Repitamos, pues, que INPUT N$ le dice al ordenador que acepte una serie que le dicta el teclado y la almacene en algún lugar de su memoria bajo el nombre de N$. Si el concepto de variable todavía se le antoja algo brumoso, no se preocupe demasiado por el momento. Más adelante lo repasaremos. Acuérdese de que nuestro objetivo principal, en estos instantes, consiste en aprender a utilizar el teclado. Si hablamos de nombres de variables en este punto es, en realidad, para ayudarle a comprender otro elemento de comprobación de errores que posee el ordenador: el corrector.

Cuando escriba la línea 20 en el teclado, hágalo intencionadamente, primero, como sigue, omitiendo adrede el carácter $:

```
20 INPUT N
```

Gramaticalmente, es una construcción perfecta. Dice: "Acepta unos datos *numéricos* y almacénalos en la memoria bajo el nombre de N". Datos *numéricos,* porque el nombre de la variable no tiene el carácter "$", que indica que se trata de una variable en serie. De momento, introduzca la línea de esta manera. Digamos de paso que la palabra INPUT está situada encima de la tecla [I]. Escriba la línea y apriete la tecla ENTER. La línea subirá hacia la cima de la pantalla, inmediatamente debajo de la 10.

Al escribir un programa, incurrirá con frecuencia en este error. Habrá introducido en él una línea perfectamente lícita, pero que no es la que usted quería. Y como la línea es buena, gramaticalmente hablando, el ordenador no tiene manera de saber que existe una irregularidad y, por consi-

guiente, la acepta tal como está escrita. Una vez introducida la línea, al examinar el programa en la pantalla, usted reconocerá el error, probablemente, y querrá subsanarlo. El T/S 1000 proporciona una manera extraordinariamente sencilla de corregir tales errores. A tal recurso lo llamamos el *corrector (editor)*.

Como la línea que usted quiere corregir es la última que introdujo, puede emplear el corrector directamente. Busque la palabra EDIT en el teclado; está situada en la tecla del [1]. Y está en rojo, con lo cual usted ya sabe que requiere un desplazamiento. Apriete la tecla SHIFT y luego toque la EDIT. La figura 2.7. le muestra lo que ocurrirá en la pantalla.

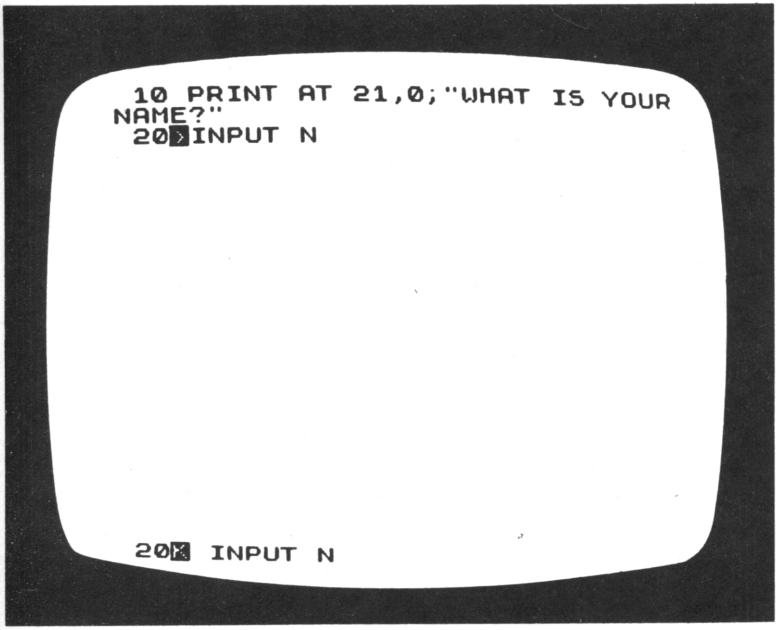

Figura 2.7.: Corregir la línea escrita más recientemente

Puede ver que una copia de la línea ha descendido hasta el fondo de la pantalla, hasta el área donde aparecen las líneas nuevas cuando las escribe. La pista K aparece dentro de la línea, inmediatamente después de su número. Ahora usted puede utilizar las teclas en flecha de dirección hacia la izquierda y hacia la derecha (SHIFT-5 y SHIFT-8, respectivamente) para mover la pista hacia atrás y adelante en la línea. Para borrar, puede situarla inmediatamente a la derecha del carácter o palabra que quiera suprimir, y luego apretar SHIFT-DELETE, como si se tratara de una línea nueva.

En el caso presente, usted quiere añadir un carácter a la línea, concretamente el $ al final del nombre de la variable. Apriete SHIFT-8 (es decir, apriete a la vez la tecla SHIFT y la tecla [8]) dos veces para mover la pista hasta la posición correcta, justo depués de la N. El símbolo del dólar es también un carácter *shift* (de desplazamiento) situado sobre la tecla [U]. Apriete SHIFT-U para añadir el carácter a la línea. Luego apriete ENTER para introducirle la versión corregida. Y entonces verá aparecer en la cima de la pantalla la línea corregida. Y, una vez, más, la pista K aparecerá en el ángulo inferior izquierdo; el ordenador estará preparado para el siguiente mandato que le transmita.

Hasta el momento, usted sólo ha introducido dos líneas del programa, pero en el proceso aprendió muchas cosas sobre su teclado y su ordenador. Repasemos brevemente lo aprendido:

- Cuando empieza a escribir una línea en el ordenador esta línea aparece en el fondo de la pantalla. Si comete un error en este punto, puede "borrar" el carácter últimamente escrito apretando la tecla SHIFT y la tecla DELETE al mismo tiempo.
- La primera palabra de cada línea es una palabra clave, que usted introduce de una sola pulsación. Luego la pista se transforma en una L, en vídeo inverso, significando *modalidad de letra.* Si quiere introducir una

función, tiene primero que conmutar a *modalidad de función* apretando SHIFT-FUNCTION. También las funciones se introducen con una sola presión de la tecla correspondiente.

- Todos los caracteres y las palabras que aparecen en rojo en el teclado son caracteres *shift* (de desplazamiento). Para escribir uno de ellos, debe apretar primero la tecla SHIFT y, sin soltarla, apretar a continuación la tecla correspondiente.
- Cuando ha completado una línea, apriete ENTER para decirle al ordenador que almacene dicha línea como una parte del programa. Si la línea no contiene errores sintácticos, ascenderá hacia lo alto de la pantalla, con el resto del listado del programa.
- Si usted pretende introducir una línea que tiene un error de sintaxis, el ordenador lo descubrirá y exhibirá la pista S para anunciarle a usted dónde está el error. Usted puede utilizar las teclas con flecha hacia la izquierda y hacia la derecha (SHIFT-5 y SHIFT-8) para mover la pista por el interior de la línea, y la tecla DELETE (SHIFT-0) para "borrar" los errores. También puede añadir palabras y caracteres, si es necesario.
- Para coregir una equivocación en una línea ya introducida en el programa, se utiliza el recurso de corrección, que se invoca escribiendo SHIFT-1.

Para las restantes líneas del programa iremos más deprisa, pero usted debería acordarse de emplear los recursos que posee el ordenador para corregir errores siempre que lo necesite. Cuando termine, su pantalla debería tener el mismo aspecto que el listado de programa de la Figura 2.1.

Terminando el programa

Cada una de las dos líneas siguientes consiste en instrucciones de una sola palabra. La línea 30 es:

30 CLS

La palabra clave CLS, situada sobre la tecla de la [V] manda al ordenador que limpie la pantalla de toda suerte de información. La instrucción de la línea 35 pone al ordenador en el modo de *cálculo-rápido* del cual hablaremos en los Capítulos 3 y 4):

35 FAST

La palabra FAST (RAPIDO) es un carácter de desplazamiento, situado en la tecla [F].

Las tres líneas siguientes, de la 40 a la 60, trabajan juntas para instruir al ordenador a repetir ciertas tareas muchas veces. A la acción resultante de esta serie de líneas la llamamos un *bucle (loop),* porque el ordenador cumple los mandatos de dichas líneas y luego empieza de nuevo y las repite indefinidamente. Las palabras clave que indican esta clase de bucle son FOR (línea 40) y NEXT (línea 60). En este ejemplo particular, la tarea que se debe repetir se define en una sola línea (la 50) pero es posible construir un bucle que repita muchas líneas. El final del bucle se señala siempre mediante la línea NEXT. La línea FOR, además de indicar el comienzo del bucle, especifica exactamente cuántas veces hay que repetirlo. En el Capítulo 3 examinaremos los detalles sintácticos del bucle FOR/NEXT.

Cuando uno ve los resultados de este programa, no le resulta nada difícil imaginar cuál es la tarea repetida. Entremos la línea 40:

40 FOR=1 TO 224

La palabra clave FOR está situada sobre la tecla [F]. El signo igual ( ) y la palabra TO son caracteres *shift* (de desplazamiento o conmutación) situados en la tecla [L] y en la tecla [4] respectivamente.

La línea 50 presenta un nuevo carácter del teclado toda-

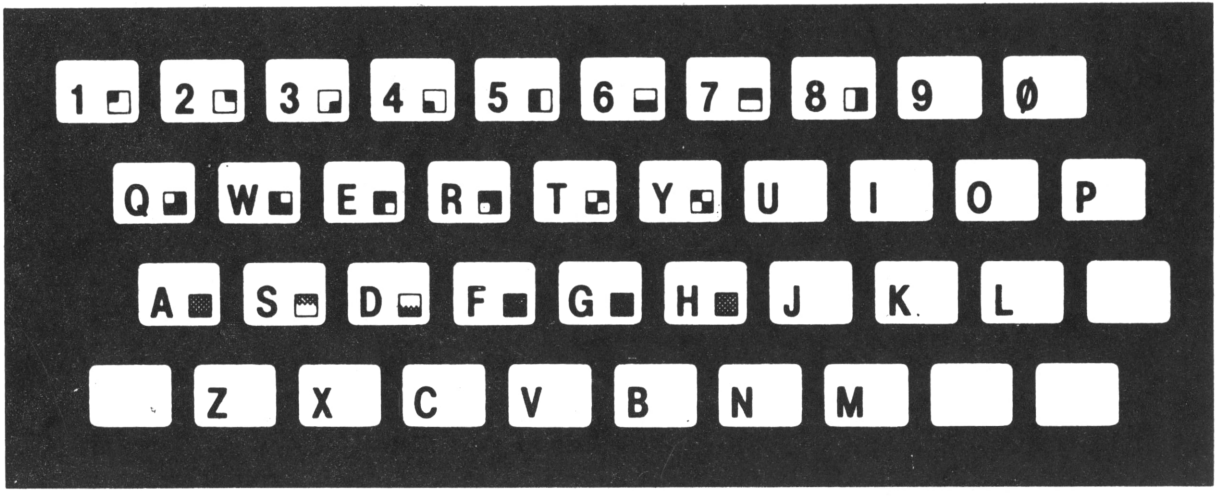

Figura 2.8.: Los caracteres gráficos

vía. Mire otra vez la línea 50 de la Figura 2.1. para ver cuál es este nuevo carácter. La línea consiste en una instrucción PRINT. Después de la palabra clave PRINT, hay dos *caracteres gráficos* dentro de unas comillas. El ordenador T/S 1000 tiene 20 caracteres gráficos, que se pueden visualizar en cualquier combinación y en cualquier lugar de la pantalla del televisor. En el teclado, aquellos caracteres están situados en las ocho primeras teclas de cifras y en doce de las teclas de letras, tal como se ve en la Figura 2.8.

Para escribir estos caracteres, usted debe pasar primero a la *modalidad de gráficos.* Empiece la línea escribiendo su número con el teclado y con las comillas de apertura:

```
50 PRINT "
```

La pista L le dice que está actualmente en la modalidad de letras. Para pasar a la de gráficos, apriete la tecla SHIFT y luego, al mismo tiempo, toque con otro dedo la [9]. (Advierta que la tecla [9] tiene la palabra GRAPHICS en rojo, sobre la cifra). Cuando haga esto, la pista cambiará y aparecerá una G en vídeo inverso, diciéndole a usted que ahora está en el modo de gráficos.

Todos los caracteres gráficos del teclado son caracteres *shift.* Los dos que usted necesita para esta línea están situados en la tecla [S] y en la tecla [H]. Empiece apretando SHIFT-S. El carácter aparecerá en la línea, y el ordenador permanecerá en el modo de gráficos. Ahora apriete SHIFT-H para el segundo carácter. La Figura 2.9. le muestra el aspecto que tendrá su pantalla en este momento.

Para completar la línea es preciso salirse ahora del modo de gráficos y volver al de letras. Para ello, simplemente, apriete de nuevo SHIFT-9. Recuerde precisamente que la palabra GRAPHICS, de la tecla [9], sirve lo mismo para eliminar el modo de gráficos que para introducirlo.

De retorno al modo L, usted puede terminar de escribir la línea introduciendo la siguiente secuencia de caracteres:

";N$ (1);

La puntuación tiene siempre gran importancia, de modo que no se pierda ningún detalle de la misma. El 1 del paréntesis que sigue a N$ se refiere al primer carácter de la serie almacenada en la variable N$. (Usted recordará esto cuando vea los resultados del programa). Los paréntesis de apertura y cierre son ambos caracteres *shift,* situados en la tecla [I] y en la tecla [O] respectivamente.

La línea 60 completa el bucle:

60 NEXT I

La palabra NEXT (LUEGO o A CONTINUACION) está situada sobre la tecla [N]. La línea 65 vuelve a situar el ordenador en el modo de *cálculo-lento,* del cual hablaremos en los Capítulos 3 y 4:

65 SLOW

La palabra SLOW (LENTO) es un carácter *shift*, situado en la tecla [D].

La línea 70 utiliza nuevamente las facultades gráficas especiales del T/S 1000. El modo de gráficos le permite a usted el pulsar caracteres de vídeo inverso, además de los 20 caracteres gráficos especiales. Para ver cómo funciona esto, empiece escribiendo la primera parte de la línea:

70 PRINT AT 9,7;"**

(Después de las comillas de apertura, queda un espacio, y luego vienen dos asteriscos. El asterisco es un carácter de desplazamiento, situado en la tecla [B]). Luego pase al modo de gráficos (SHIFT-9) y escriba la palabra HELLO, sin apretar la tecla SHIFT. Verá usted que la palabra entera aparece en caracteres blancos sobre fondo negro; es decir, en vídeo inverso. Ahora pulse la tecla de espaciado; un espacio en

vídeo inverso es, simplemente, un cuadrado negro. El espacio es el final del mensaje; por lo tanto, abandone el modo de gráficos y escriba los dos caracteres últimos de la línea: el signo final de interrogación y el punto y coma. Apriete la tecla ENTER para que la línea pase a formar parte del programa.

Las tres líneas siguientes —80, 90 y 100— componen otro bucle FOR/NEXT. En este punto, ya no debería encontrar dificultad alguna en escribir las líneas con el teclado. Como utilizan tres funciones de programación nuevas, usted tendrá que pasar al modo de función tres veces. La línea 80 emplea la función LEN, situada debajo de la tecla [K]:

```
80 FOR I  1 TO LEN N$
```

La función LEN (abreviación de LENGHT: largo, longitud) suministra la *longitud* en caracteres, de una serie (o cadena). La línea 90 emplea la función CHR$ (debajo de la tecla [U]) y la función CODE (debajo de la tecla [I]):

```
90 PRINT CHR$ (CODE N$ (I) + 128);
```

Esta quizá sea la instrucción más compleja del programa. Brevemente, convierte los caracteres de la cadena (o serie) N$ en sus equivalentes en vídeo inverso. Estudiaremos las funciones CHR$ y CODE en el Capítulo 5. La línea 90 contiene otro elemento nuevo: el signo más ( ). Se trata de un carácter de desplazamiento, situado en la tecla [K].

La línea 100 termina el bucle FOR/NEXT:

```
100 NEXT I
```

Y, finalmente, la línea 110 es otra instrucción de PRINT:

```
110 PRINT "* *"
```

Advierta que hay tres caracteres dentro de las comillas: primero un espacio, luego dos asteriscos.

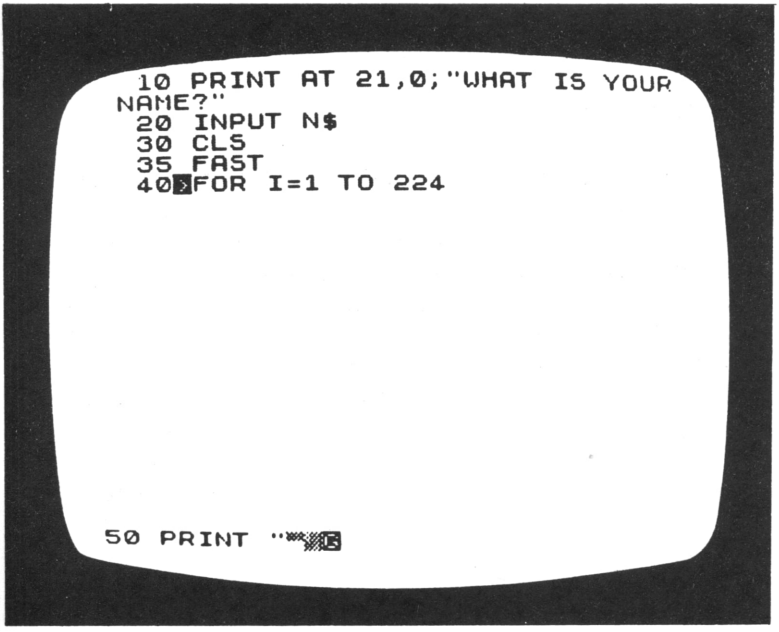

Figura 2.9.: Escribiendo los caracteres gráficos

Ahora ha introducido usted el programa entero en el ordenador, y está preparado para ver qué pasa cuando el ordenador cumple sus instrucciones. De todos modos, antes de pasar al programa haga un par de cosas. Primero, compare cuidadosamente la pantalla, línea por línea, con el listado del programa que mostramos en la Figura 2.1. Asegúrese de que cada carácter esté exactamente bien. Si encontrase algunos errores, no sería difícil corregirlos. Salte adelante, hasta la sección del presente capítulo titulada: "Corrigiendo cualquier línea del programa", en la que verá la manera de efectuar cambios adicionales en su programa. (Usted ya sabe emplear el corrector; lo único que le falta aprender es la manera de especificar qué línea quiere corregir).

La segunda cosa que debe hacer antes de pasar el programa es revisar todo lo que aprendió sobre los modos del

teclado. Estudie la Figura 2.10., que le resume las diferentes pistas.

¡ACCION!

La palabra clave RUN (CORRE, o PASA), situada sobre la tecla [R], manda al ordenador que empiece a ejecutar las instrucciones del programa que guarda en la memoria. Cuando pase usted programa, cuya introducción en el ordenador le ha costado tantas fatigas, reconocerá, probablemente, la fuente o al menos parte de la acción en alguna de las líneas más comprensibles que ha escrito. En este punto, como aún no ha estudiado BASIC, la idea que tenga de cómo trabaja el programa se deberá más bien a impresiones; no será nada sistemática. A pesar de lo cual, quedará impresionado al ver cuan bien sabe deducir el significado de la mayoría de líneas del programa después de ver los resultados.

Su conversación con el ordenador

Escriba la palabra RUN directamente, sin hacerla preceder de un número. Aprieta la tecla ENTER y vea qué pasa.

Por primera vez, el listado de las líneas del programa ha desaparecido de la pantalla. Naturalmente, las líneas siguen almacenadas en la memoria del ordenador, pero ahora la pantalla se necesita para exhibir los *resultados* del programa.

La acción se realiza en dos pasos distintos, como verá usted pronto. En el primer paso, la pregunta:

WHAT IS YOUR NAME (¿COMO SE LLAMA USTED?)

aparece en la pantalla. La pista L, entre comillas, está impresa debajo de la pregunta. Esto le dice que el ordenador está esperando que escriba usted una respuesta en el teclado. Adelante, pues, escriba su nombre. (La Figura 2.11. muestra el aspecto que tendrá con el nombre de HILDA). Si usted

comete un error al escribir, puede utilizar la tecla DELETE (SHIFT-0) para borrarlo, tal como lo hacía al escribir el programa.

Cuando haya escrito su nombre, apriete la tecla ENTER. La pantalla se quedará en blanco durante unos momentos, y luego... bueno, lo verá por sí mismo. (La Figura 2.12 presenta el saludo a Hilda). Fíjese en que inteligentemente ha incorporado el ordenador la inicial de su nombre en el diseño de fondo.

Antes de continuar, debería usted hacer una pausa para meditar un momento sobre la ilusión creada por su programa. Da la sensación de que *el ordenador* le ha dirigido una pregunta perfectamente cortés, ha absorbido su respuesta, y ahora ha correspondido de manera algo extravagante, pero

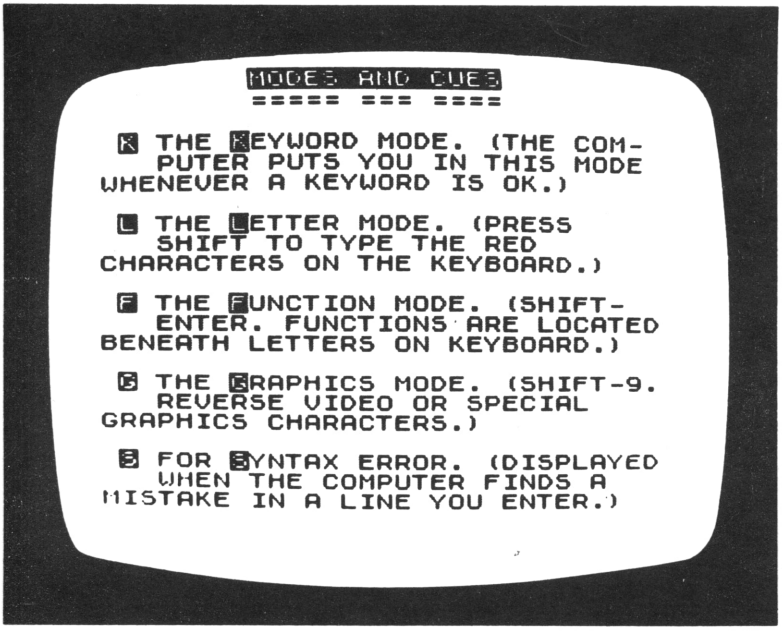

Figura 2.10.: Los modos y pistas

correcta, y que parece inteligente. Por supuesto, usted sabe que este "diálogo" tiene algo de farsa, dado que cada detalle de la acción estaba perfectamente controlado por el programa que introdujo en el ordenador. A la máquina le importa un comino cómo se llama usted y no siente el menor deseo de conseguir su amistad. Pero usted puede crear la *ilusión* de la amabilidad, o de cualquiera otra clase de reacción, según la manera de escribir sus programas. Más aún, la persona que responde a la pregunta visualizada en la pantalla no ve, ni poco ni mucho, el programa, que parece escondido entre bastidores. (Sólo como experimento, pase el programa de nuevo ante otra persona y fíjese en cómo reacciona ésta). El crear esta ilusión de inteligencia —de responder correctamente a un input humano— es una parte esencial de la natu-

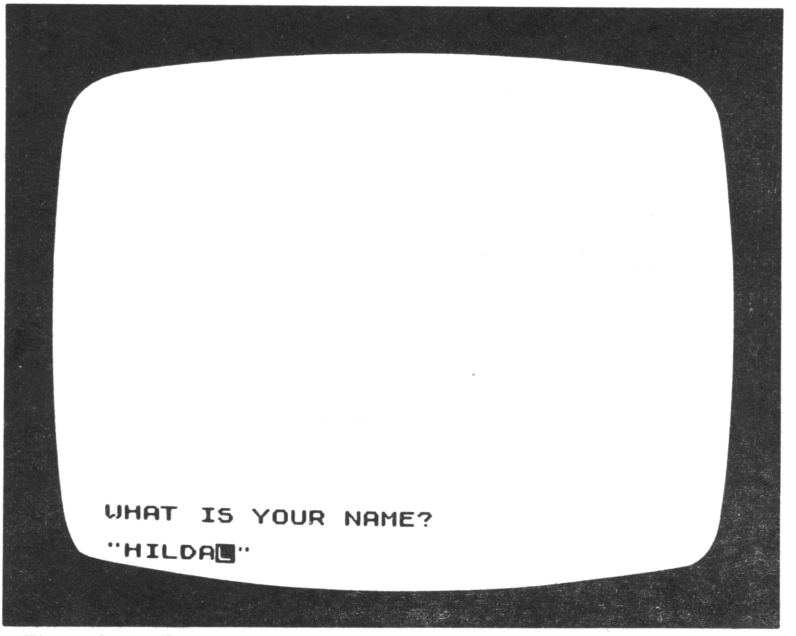

Figura 2.11.: Responda a una pregunta en la pantalla

raleza íntima de un programa. El ordenador no es capaz de hacer nada si no se le han dado instrucciones detalladas que le guíen; pero, irónicamente, si usted aprende a escribir unos programas inteligentes, la inteligencia la atribuirán al ordenador. Nadie se acordará de que el programa esté ahí.

Volviendo a la pantalla del televisor, tome nota de un mensaje corto que aparece en el ángulo inferior izquierdo. El ordenador ha terminado ya de ejecutar todas las líneas del programa, y los números del rincón representan una especie de informe de la situación de los resultados del programa:

0/110

En capítulos posteriores examinaremos detalladamente la clase de información que esta referencia puede darle. En el ejemplo que ponemos aquí, el cero significa que el programa quedó completado por entero, y no surgieron problemas especiales durante su ejecución.

CORREGIR CUALQUIER LINEA

Si ha terminado de admirar la amable salutación del ordenador, apriete ahora la tecla de ENTER. El saludo desaparece y el listado del programa retorna a la pantalla. Usted puede pasar el programa tantas veces como quiera, siempre introduciendo la palabra clave RUN; el ordenador no se cansará nunca de repetirlo.

Además, siempre puede revisar el programa, si quiere. Puede hacerlo de diferentes maneras, según la clase de revisión que quiera efectuar:

- Puede añadir una línea nueva, simplemente, introduciéndola mediante el teclado y asegurándose de asignarle un número que no haya dado ya a otra línea del programa.
- Puede borrar cualquier línea escribiendo su número correspondiente y luego apretando la tecla ENTER. (Sa-

biendo esto, debe tener cuidado en no borrar una línea accidentalmente escribiendo un número que corresponda a una línea de su programa).

- Puede reescribir una línea entera con una instrucción nueva marcando el número de la línea en cuestión. La línea antigua desaparecerá, y la nueva ocupará su puesto.
- Finalmente, si sólo tiene que efectuar un ligero cambio en una línea, puede *corregirla*. Lo ensayaremos ahora.

Quizás haya observado un pequeño detalle del listado del programa que hasta el momento no hemos mencionado: la flechita de vídeo inverso que señala la última línea que usted introdujo en el programa. Si escribió las líneas en secuencia,

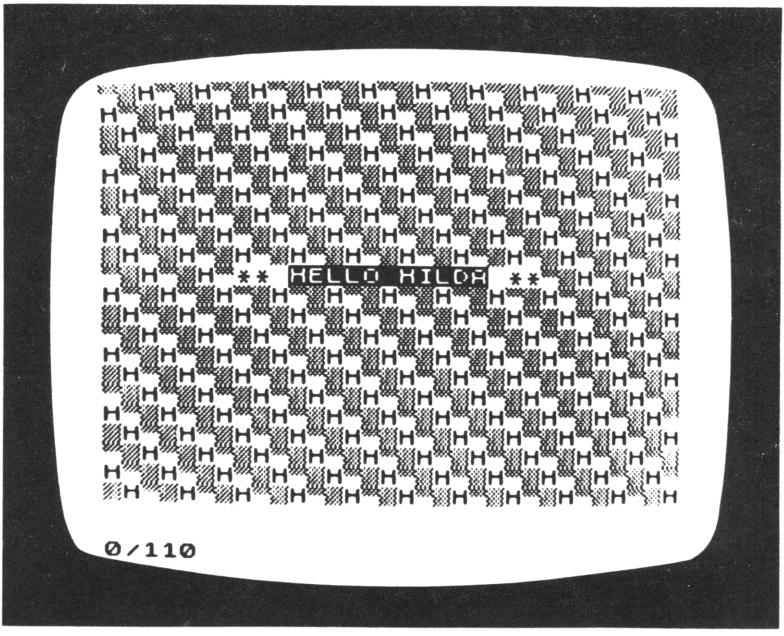

Figura 2.12.: El saludo del ordenador

esta flecha señala ahora la línea 110. (Puede verla en la Figura 2.1.). La flecha indica la *línea actual,* es decir, la que quedará corregida si usted aprieta la tecla ENTER. Pero usted puede mover la flecha arriba y abajo del listado del programa apretando flechas en dirección de arriba y abajo del teclado (SHIFT-6 y SHIFT-7).

Digamos, por ejemplo, que le gustaría cambiar el saludo que el ordenador exhibe en la pantalla. Usted quiere cambiar HELLO por HOLA. Esta parte del saludo está en la línea 70 del programa, de modo que esa es la línea que quiere corregir.

Si la flecha de línea actual señala la 110, tiene que moverla para arriba cuatro veces para corregir la línea 70. He ahí los pasos que seguirá:

1. Con un dedo apriete la tecla SHIFT, y luego la tecla [7] cuatro veces. Cada vez que la apriete, el listado reaparecerá en la pantalla, y la flechita ascenderá una línea.
2. Ahora, con la flecha señalando la línea 70, apriete la tecla EDIT (SHIFT-1) y una reproducción de la línea descenderá hasta el fondo de la pantalla, preparada para ser corregida.
3. Utilizando la tecla de flecha hacia la derecha (SHIFT-8) mueva la pista L hasta la derecha de la letra O de HELLO. Entonces apriete la tecla DELETE (SHIFT-0) cuatro veces, para borrar cuatro letras.
4. Conmute hacia el modo de gráficos (SHIFT-9) y escriba las nuevas letras -OLA.
5. Finalmente, sálgase del modo de gráficos (SHIFT-9 otra vez) y apriete la tecla ENTER. La versión revisada de la línea 70 aparecerá como parte del listado del programa.

Ahora pase el programa para asegurarse de que sigue funcionando. Todo debería estar igual, excepto que HOLA substituirá a HELLO en el saludo.

Una nota adicional sobre el mover la flecha de línea actual en listado del programa: Cuando esté trabajando con un programa muy largo, quizá necesite una manera más rápida de mover la flecha de una línea a otra, especialmente si se trata de líneas muy distantes. En este caso puede emplear la palabra clave LIST. Digamos que usted quiere situar la flecha en la línea 40. Introduzca el mandato:

LIST 40

Su programa quedará listado en la pantalla desde la línea 40 en adelante. Ahora pulse la tecla EDIT, y una copia de la línea 40 descenderá hasta el fondo de la pantalla para ser corregida.

PARA LA ACTUACION SIGUIENTE...

En el Capítulo 1 hablábamos de la necesidad de una cassette a fin de proporcionarnos un medio permanente de almacenamiento de programas. Usted recordará que los programas están almacenados en la memoria temporal del ordenador, con lo cual, al desenchufar el aparato, se pierden.

En esta sección, usted aprenderá a utilizar una grabadora de cassettes en combinación con el ordenador. Si posee una grabadora prepárela. Procederemos a una sesión de prácticas con el programa de saludo.

Guardar el programa en una cinta

Busque el cordón corto que tiene un par de clavijas en cada extremo. Si tiene dos clavijas es para que usted pueda conectar los clavijeros del auricular y el micrófono de su ordenador a la vez. El conectarlas al mismo tiempo, puede interferir, algunas veces, con el proceso del grabado. Las clavijas son de colores distintos, con lo cual usted puede hacer concordar fácilmente la clavija de un extremo con su pareja del otro extremo.

Para grabar un programa en una cassette de su ordenador, conecte primero la clavija rotulada "MIC" (en el costado izquierdo del ordenador) al clavijero del micrófono del grabador de cintas. Asegúrese de enchufar firmemente ambos extremos del cordón en los clavijeros acertados. (Si descubre que la clavija queda demasiado suelta en el ordenador, acaso tenga que sujetarla manualmente en el clavijero, tanto durante las sesiones de grabado como durante las de reproducción).

Los dispositivos de volumen y de tono del grabador de cintas tienen mucha importancia. Empiece con el dispositivo del volumen colocado a la mitad de la potencia máxima, más o menos. Ponga el dispositivo de tono en la nota más alta y/o en la más baja. Si la primera tentativa que hace por grabar un programa no tiene éxito, pruebe de aumentar el volumen.

Ponga una cassette nueva en su registrador de cintas, y enróllela hasta el comienzo, si es necesario. Si el grabador de cinta tiene un contador, póngalo a cero, a fin de poder guardar constancia de los puntos de principio y final de la grabación.

La palabra clave que se emplea para guardar un programa en una cassette es SAVE, situada sobre la tecla [S]. El mandato de SAVE (GUARDA, o AHORRA) requiere un título de programa, escrito entre comillas. Usted debería imaginar un título que le recuerde exactamente qué hace el programa. A éste lo llamaremos SALUDO (GREETING). Escriba el mandato, pero no lo introduzca todavía:

```
SAVE "GREETING"
```

Ahora apriete el botón de RECORD (REGISTRO, GRABADO) del grabador de cinta, déjelo rodar unos momentos y luego apriete la tecla ENTER del ordenador. La pantalla se quedará en blanco unos momentos. Cuando vea que se llena de líneas horizontales (de una pulgada —2'54 cm.— aproximadamente de anchura) que oscilan rápidamente arriba y abajo, sabrá que el programa está pasando al grabador

de cintas. El tiempo de permanencia de tales franjas en la pantalla depende de la longitud del programa. Grabar el programa de saludo sólo precisará unos segundos. Los programas muy largos pueden requerir varios minutos.

Completado el grabado, la pantalla se quedará en blanco de nuevo excepto por el mensaje:

0/0

presentado en el ángulo inferior izquierdo. Esto indica una excelente realimentación del programa. Oprima la tecla ENTER y el programa aparecerá listado en la pantalla.

Si la pauta de franjas no cesa cuando debería, algo ha funcionado mal. Pare el grabador de cinta y apriete la tecla de SPACE de su ordenador. (Advierta que sobre esta tecla está escrita la palabra BREAK —CORTE—). Siempre puede apretar la tecla BREAK para interrumpir el pase de un programa, o un proceso de grabación o de carga).

Las causas más probables de fracaso cuando usted trata de guardar o cargar un programa son:

1. Conexiones malas. (Compruebe si las clavijas están bien metidas en los clavijeros correspondientes).
2. Disposiciones incorrectas de volumen o de tono. (Tiene que ensayar cuál es la mejor disposición de volumen para su grabador de cintas).

LA COMPRA DE SOFTWARE

El dominar la conexión del grabador de cinta hasta el punto de ser usted capaz de grabar y recargar programas fácil y expertamente quizá requiera cierta cantidad de paciencia y perseverancia por su parte. Pero la recompensa a sus esfuerzos, aparte del almacenamiento permanente de sus propios programas, será la facultad de utilizar programas escritos y grabados por otras personas. En particular, podrá comprar programas —algunos bastante sofisticados— que

acaso le brinden perspectivas nuevas sobre el empleo de su ordenador.

La cantidad de software disponible en cassettes para el T/S 1000 está aumentando rápidamente. (Digamos de paso que *software* es palabra que usamos para distinguir los programas de ordenador de los componentes de *hardware* del sistema del ordenador). En categorías de interés general tales como negocios, finanzas domésticas, educación y juegos, encontrará una buena selección de programas para elegir. Muchos de tales programas requieren el módulo de 16K de memoria suplementaria, pero algunos se pueden utilizar en la unidad de BASIC del ordenador sin memoria adicional.

Para darle a usted una idea del programa que podrá comprar, veremos un ejemplo en la sección siguiente de este capítulo. Revisaremos un programa llamado VU-CALC, que es, probablemente, uno de los más populares entre los disponibles para el T/S 1000.

El VU-CALC, un programa de hoja distribuida

Es posible que usted haya oído hablar de la familia de software llamada *programas de hoja distribuida (spreadsheet programs)*. Estos programas le permiten organizar grandes cantidades de datos numéricos y realizar con ellos muchos cálculos rápida y eficazmente. Los programas de hoja distribuida son actualmente asequibles para la mayoría de ordenadores personales realmente grandes. Dos ejemplos de tales programas son el VisiCalc ® y el SuperCalc ™, ambos extraordinariamente populares entre usuarios de ordenadores pequeños. Los programas de hoja distribuida simplificarán y acelerarán cualquier tarea de computación que implique el mantenerse al corriente de grandes colecciones de datos numéricos y efectuar cálculos con ellos.

El VU-CALC, diseñado para el T/S 1000 con módulo de memoria suplementaria, es el "hermano menor" de dichos programas. Aunque no es tan potente ni tan polifacético como los programas de hoja distribuida diseñados para orde-

nadores más grandes, es un instrumento de programación muy digno de ser tenido en cuenta por toda persona que se enfrente con la molestia de unas tareas aritméticas prolongadas.

La Figura 2.13 muestra un ejemplo sencillo de un VU-CALC de hoja distribuida. El VU-CALC le proporciona a usted una hoja de trabajo grande y vacía, de 36 columnas por 26 filas, en la cual puede escribir palabras, números y fórmulas. Cada lugar de la hoja queda designado por una "dirección" de dos coordenadas: una letra (que indica la fila) y un número (que indica la columna). Así, por ejemplo, el lugar F01 de la hoja de la Figura 2.13 contiene la palabra NET. Los lugares F02 y F03 contienen cantidades de ingresos netos correspondientes a los años 1981 y 1982 respectivamente.

Cuando haya empezado usted a escribir números en su hoja distribuida, puede crear datos adicionales escribiendo fórmulas que se refieran a los datos ya existentes en la hoja. Por ejemplo, los lugares G02 y G03 muestran cantidades de impuestos sobre la renta calculadas sobre los valores de los ingresos netos contenidos en los lugares F02 y F03, respectivamente. En el ángulo inferior izquierdo de la pantalla puede ver la fórmula que se empleó para calcular el valor del lugar G03:

```
F03 * .35
```

(En esta fórmula, el asterisco representa la multiplicación. La fórmula aparece en la pantalla porque el *cursor* de la hoja de trabajo está actualmente en G03). Usted sólo tiene que escribir la fórmula una vez; luego puede aplicarla, si lo desea, a muchas filas o columnas de datos. Lo más importante de todo es que tales fórmulas son dinámicas; se recalculan cada vez que algún dato de la hoja de trabajo cambia.

Aunque uno sólo ve tres columnas y nueve filas de hoja de trabajo a la vez, la pantalla del televisor actúa como una "ventana" móvil sobre la hoja entera. Para ver otras partes

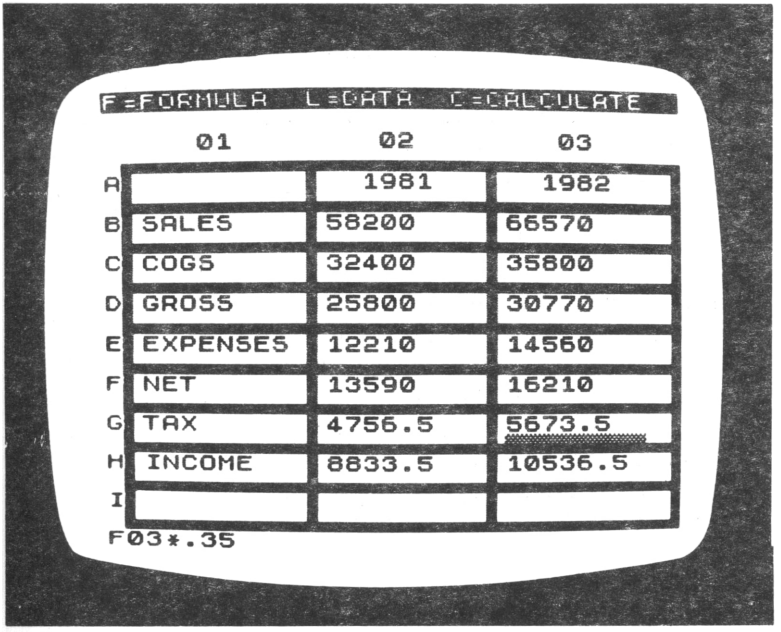

Figura 2.13.: Ejemplo de VU-CALC

de la hoja en cuestión, se puede "mover la ventana" apretando las cuatro teclas de dirección situadas en la cima del teclado.

El VU-CALC también permite conservar datos de hoja distribuida en cassette, junto con el programa. De esta manera se pueden conservar registros permanentes de los datos que pusiéramos en hojas distribuidas importantes.

Una Guía del Consumidor para comprar software destinada al T/S 1000

Desgraciadamente, la software de cassette disponible para el ordenador es relativamente cara. Si compra usted media docena de programas, le costarán tanto dinero como le costó el propio ordenador. Por esta razón, debería aprender a por-

tarse como un comprador meticuloso, selectivo y bien informado cuando se disponga a comprar software. Debería encontrar un comerciante que le permita probar un programa antes de adquirirlo. (Muchos comerciantes tienen "muestras de prueba" de todas las software que venden, para que el cliente pueda inspeccionarlas).

Cada programa va acompañado de "documentación", un panfletito explicando la manera de utilizar el programa. Tómese el tiempo de leer la documentación de un programa que esté pensando comprar. La documentación debería describir clara y sencillamente todo lo que usted debe saber para pasar y emplear el programa con éxito. Si la documentación es mala, es posible que pase tanto tiempo probando de entender el programa que la compra efectuada no le rinda beneficio alguno.

Aquí tiene unas preguntas que debería hacerse antes de comprar una nueva pieza de software:

- Si el programa está orientado hacia los negocios o las finanzas domésticas, ¿le será útil en muchas ocasiones distintas? ¿Realiza tareas concretas de computación que usted necesita efectuar? (A medida que usted adquiere pericia en el BASIC, será capaz de escribir muchos programas por sí mismo, diseñados especialmente para las tareas que deba llevar a cabo).
- Si el programa es un juego, ¿se trata de un juego con el que guste entretenerse muchas veces, repetidamente?
- Si tiene usted unas preguntas que hacer sobre la manera de utilizar el programa, ¿será capaz de encontrar las respuestas, bien mediante la documentación, bien consultando al comerciante? Cuando usted prueba el programa en la tienda, ¿parece hacer lo que la documentación afirma que hará?
- ¿Cómo reacciona el programa cuando usted comete un error? ¿Le proporciona a usted una manera fácil de corregir errores? Al ensayar el programa, debería usted

cometerlos intencionadamente y tomar nota de los resultados.

Preguntas como éstas le ayudarán a decidir si un determinado programa merece que usted invierta dinero y tiempo en él.

RESUMEN

El aprender a utilizar el teclado del ordenador es cosa fácil en cuanto uno domina los diversos modos en los que se puede situar dicho teclado. Dado que cada línea de un programa BASIC tiene que empezar con una palabra clave, el ordenador siempre comienza con el modo K. A continuación de la palabra clave, una instrucción puede contener letras, números y caracteres de desplazamiento (el modo L); gráficos o caracteres en vídeo inverso (el modo G) y funciones (el modo F).

El ordenador tiene varias maneras importantes de ayudarle a reconocer y corregir todas las equivocaciones en que pueda incurrir usted al escribir un programa en el teclado. Si entra usted una línea que contenga un error de sintaxis, el ordenador la rechazará, exhibiéndola en el fondo de la pantalla con la pista S señalando el error. Para corregir la línea, usted puede utilizar las teclas con flechas hacia la derecha o hacia la izquierda para mover la pista hasta la posición adecuada de la línea, y luego puede añadir o borrar caracteres. Por otra parte, si introduce una línea sintácticamente correcta que posteriormente desee revisar, puede emplear la facultad de EDIT (CORREGIR) para efectuar el cambio.

El dominio de la conexión del grabador de cassettes con el ordenador quizá requiera cierta cantidad de ensayos por parte de usted. No obstante, cuando haya aprendido a emplear un grabador de cassettes para almacenar y cargar programas, su ordenador será para usted una herramienta muchísimo más valiosa. Usted podrá empezar a sopesar las ven-

tajas relativas de la software comprada, y empezará a formarse una biblioteca de programas escritos por usted mismo.

CAPITULO 3
EL ARGUMENTO COBRA CUERPO: UN BREVE, GRAFICO, CURSO DE BASIC

INTRODUCCION

En este capítulo aprenderá usted lo esencial del BASIC, al mismo tiempo que se divierte un poco con las características gráficas de su ordenador. Las técnicas programadoras que adquiera no sirven únicamente, claro está, para escribir programas graficos. Usted empleará los mismos mandatos y las mismas técnicas más adelante, cuando empiece a escribir otra clase de programas. Pero los ejercicios gráficos de este capítulo le ofrecen una manera muy agradable de empezar a manejar el BASIC. En todo momento, podrá *representarse* exactamente lo que esté haciendo el ordenador en respuesta a las instrucciones que usted le dé.

El T/S 1000 tiene dos sistemas distintos de producir gráficos en la pantalla. Usted ha visto un ejemplo de uno de ellos en el Capítulo 2, con el empleo de caracteres graficos especiales y la exhibición de letras en vídeo inverso. Empezaremos el presente capítulo investigando detalladamente ambos sistemas. Luego recorreremos a paso vivo un breve curso de BASIC, relativo a los mandatos que utilizará más frecuentemente en sus programas. Verá las ilustraciones de cada mandato en un número de programas pequeños, y, para resumir lo aprendido, trabajará con un programa de gráficos corto que dibuja rectángulos en la pantalla.

El programa final de este capítulo es relativamente largo, aunque encaja perfectamente dentro de los 2K de memoria que le proporciona su ordenador. Usted podrá emplearlo para trazar muchas clases de dibujos en la pantalla. Y tales dibujos irán mejorando más y más a medida que su pericia en

el empleo del programa aumente. El programa en sí merece un estudio minucioso, puesto que ilustra varias técnicas importantes que le ayudarán a escribir programas más claros y eficaces.

MANDATOS INMEDIATOS

En el Capítulo 2 hemos hablado de tres características de un programa BASIC:

- las líneas están numeradas;
- cada línea empieza con una palabra clave
- el ordenador únicamente *almacena* las líneas, sin efectuar ninguna de las instrucciones, hasta que usted escribe RUN.

Sucede que el ordenador también le permite a usted emplear muchas teclas (aunque no todas) como mandatos *inmediatos*. Es decir, usted puede escribir una instrucción, sin un número de lista que la preceda, y el ordenador la cumplirá inmediatamente, sin esperar que usted pulse la tecla de RUN. A veces usted querrá utilizar en seguida mandatos para explorar las reacciones del ordenador ante una instrucción u otra.

Por ejemplo, introduzca el mandato siguiente en su teclado:

```
PRINT "T/S 1000"
```

Y verá que el ordenador imprime inmediatamente T/S 1000 en la cima de la pantalla. Ahora si aprieta la tecla de ENTER otra vez, verá que el ordenador *no* ha guardado esta instrucción de PRINT. Los mandatos inmediatos se cumplen una sola vez, y luego se pierden.

En la actualidad, usted ya tiene otros varios ejemplos de mandatos inmediatos. Sabe que SAVE y LOAD se utilizan para el almacenamiento y la recuperación de programas en

cassettes. LIST manda al ordenador que visualice las líneas del programa en la pantalla. RUN hace que el ordenador empiece a realizar las líneas del programa. Otro mandato que quizás tenga ocasión de emplear muy pronto es NEW, que destierra el programa actual de la memoria del ordenador, a fin de que usted pueda escribir uno nuevo. Todas estas instrucciones suelen darse como mandatos inmediatos... aunque también pueden utilizarse como líneas de programas.

Otros mandatos, tales como PRINT, se usan normalmente como parte de un programa, pero se pueden ejecutar también como mandatos inmediatos. También utilizaremos PRINT como mandato inmediato en la sección siguiente de este capítulo, con objeto de efectuar varios experimentos rápidos.

DOS SISTEMAS DE GRAFICOS

La instrucción PRINT AT

Cuando usted emplea el mandato de PRINT para visualizar información en la pantalla, todos los caracteres que escribe ocupan una cantidad igual de espacio. Imagínese la pantalla como una cuadrícula, dividida en filas y columnas. El área de la pantalla es de 22 filas de largo y 32 columnas de ancho. (La parte del fondo, debajo de la fila 22, se la reserva el ordenador como espacio de trabajo. Como usted sabe bien a estas alturas, es el área donde aparecen las líneas nuevas que usted va escribiendo. El mandato PRINT no le sirve jamás para tener acceso a esta área).

Utilizando la instrucción PRINT AT, usted puede especificar exactamente en qué lugar de la pantalla quiere que aparezca la información. Tal como vio en el Capítulo 2, a PRINT AT le siguen siempre dos números. Estos números representan una "dirección" de la pantalla. La Figura 3.1. le ayudará a entender cómo se determinan estas direcciones. El primer número que sigue a PRINT es la *fila,* y el segundo

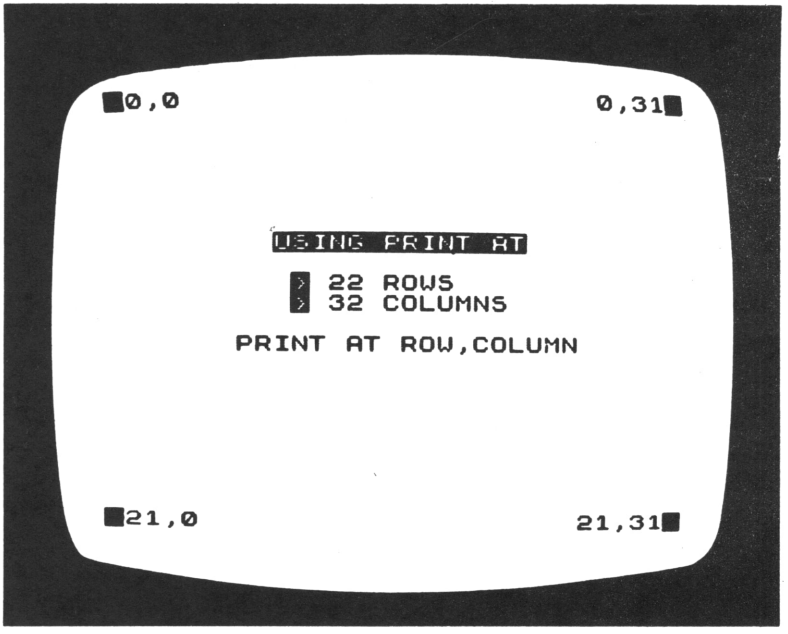

Figura 3.1.: Empleo de la instrucción PRINT AT

número es la *columna* de la dirección. De modo que la forma correcta de la instrucción PRINT AT es:

PRINT AT fila, columna; "alguna información"

Advierta una vez más que los números que representan la fila y la columna están separados por una coma, y que la dirección está separada de la información por un punto y coma.

Ahora contemple de nuevo la Figura 3.1. La dirección empieza en el ángulo superior izquierdo de la pantalla. Si lo prefiere, a esta situación puede considerarla el *origen*. Su dirección es 0,0. Observe la dirección de los otros tres ángulos de la pantalla. Los números de las filas crecen a medida

que se desciende, y los de las columnas aumentan a medida que uno se desplaza hacia la derecha.

Para asegurarse de que entiende bien este sistema de direcciones, estudie las instrucciones siguientes, y trate de determinar aproximadamente en qué lugar de la pantalla situará el mensaje cada una de ellas:

```
PRINT AT 19,5; "X"
PRINT AT 7,7; "¿A DONDE IRA ESTO?"
PRINT AT 15,25; "HOLA"
```

Ahora introdúzcala como mandatos inmediatos, una después de otra y vea si predijo acertadamente cuáles serían los resultados. Advierta que cuando el mensaje contiene más de un carácter, el *primero* es el que estará situado en la dirección indicada; el resto de los caracteres seguirá a continuación en la misma línea.

Se puede emplear una simple instrucción de PRINT para visualizar varios mensajes en lugares diferentes. Ensaye el mandato siguiente:

```
PRINT AT 3,3; "AQUI"; AT 20,25: "ALLI"
```

En resumen, PRINT AT se puede usar para imprimir cualquier carácter, palabra, número, o carácter especial de gráficos en cualquier posición de la pantalla de 22-por-32.

PLOT Y UNPLOT

El ordenador le proporciona además un segundo método de enviar gráficos a la pantalla del televisor, utilizando las palabras clave PLOT y UNPLOT (TRAZA y DESTRAZA). Estos mandatos organizan la pantalla de un modo algo distinto y requieren un sistema de direcciones diferente.

La Figura 3.2. resume el empleo de PLOT y UNPLOT. El mandato de PLOT produce un cuadrito negro, llamado pixel (de las palabras inglesas *picture element* elemento de

cuadro) en un lugar especificado de la pantalla. Cada realización de PLOT pone un solo pixel en la pantalla. Los cuatro ángulos de la pantalla de la Figura 3.2. contienen pixels. El área de la pantalla puede exhibir 44 filas por 64 columnas de pixels. Recuerde el sistema de dirección de cuadrícula de la instrucción PRINT AT. Si usted dividiese cada cuadro de esta cuadrícula en cuatro cuadritos menores, vería el tamaño de un pixel.

El origen de las direcciones pixel está en el ángulo inferior izquierdo, en vez de en el superior derecho. Este lugar tiene la dirección 0,0. Las direcciones de filas crecen hasta 43 a medida que se asciende por la pantalla, y las direcciones de columnas aumentan hasta 63 a medida que se desplaza uno hacia la derecha. La forma general de la instrucción PLOT es:

PLOT columna, fila

Para familiarizarnos con los pixeles y el sistema de direcciones utilizados para el mandato PLOT, introduzca las siguientes líneas, una tras otra, como mandatos inmediatos, y observe dónde sitúa cada una su pixel:

```
PLOT 5,5
PLOT 55,3
PLOT 40,40
```

La instrucción UNPLOT "borra" un pixel de la pantalla. Toma la misma forma que el mandato PLOT:

UNPLOT columna, fila

En los programas del presente capítulo veremos un buen número de mandatos PRINT AT, PLOT y UNPLOT, de modo que asegúrese usted bien de entender cómo funcionan, antes de seguir leyendo. Si todavía no está bien impuesto de los dos sistemas diferentes de organizar la pantalla, revise

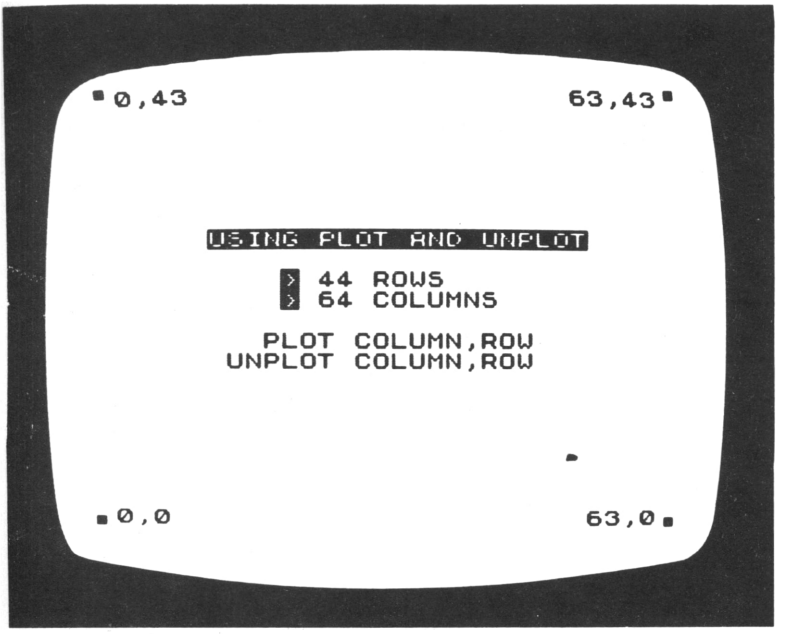

Figura 3.2.: Empleo de los mandatos PLOT y UNPLOT

las figuras 3.1. y 3.2. y siga ensayando con mandatos inmediatos.

UN CURSO RAPIDO DE BASIC

Esta sección le introducirá en la mayoría de los elementos esenciales de la programación en BASIC para el T/S 1000: definir variables; leer información de input; repetir instrucciones; controlar el orden del funcionamiento, y tomar decisiones. Usted encontrará muchos programas cortos de muestra que ilustrarán cada punto. Importa mucho que introduzca cada uno de tales programas en su ordenador y los ensaye al mismo tiempo que va leyendo. Si no entiende por qué producen los resultados que dan, retroceda y revise la explicación dada.

Variables revisitadas

Hemos visto que una variable es sencillamente un lugar en la memoria del ordenador reservado para un determinado tipo de información. Se pueden definir muchas variables diferentes para emplearlas en un mismo programa. A fin de poder acceder fácilmente a la información, cada variable tiene un nombre. La información contenida en una variable puede cambiar, pero el nombre sigue siendo el mismo. También hemos visto que hay dos *clases* de variables: numéricas y de texto, o *en cadena* (o serie).

Se pueden definir variables —o el valor contenido en una de ellas— mediante la declaración de LET. La forma general de la declaración de LET es:

LET NOMBRE VALOR

Esta declaración, parafraseada, significa que "almacena VALOR (VALUE) en una variable llamada NOMBRE (NAME)". Cuando el ordenador efectúa una declaración LET, primero evalúa la información de la derecha del signo igual, y luego almacena ésta en la variable que se menciona en la izquierda de dicho signo.

Veamos unos ejemplos. La declaración de LET se puede usar como un mandato inmediato, de modo que uno puede entrar las líneas siguientes directamente, sin números de línea. Digamos que usted quiere definir una variable llamada INGRESOS para almacenar la cantidad de dinero que ganó este año. Su mandato podría ser una cosa así:

LET INGRESOS 15.000

Introduzca este mandato y anote los resultados. Usted ve solamente el mensaje 0/0 en el ángulo inferior izquierdo de la pantalla, significando que el ordenador ha cumplido el mandato. Ahora, para asegurarse de que INGRESOS ha quedado bien definido, entre:

```
PRINT INGRESOS
```

Usted verá que el *valor* de la variable INGRESOS —15.000— está imprimido en la parte superior de la pantalla. De modo que, cuando incluya un nombre de variable en una declaración de PRINT, el ordenador exhibe el *valor* contenido en dicha variable en la pantalla.

Usted es libre de elegir el nombre que prefiera para una variable. Siempre que sea posible, es mejor escoger uno que describa el significado de la variable. En el caso de las variables *numéricas,* a usted le corresponde también el decidir sobre la longitud del nombre de la variable. Muchas personas prefieren utilizar nombres de variables largos, tales como INGRESOS, porque con ellos es más fácil leer el programa. Uno ve al instante qué significa la variable en cuestión. Las desventajas de los nombres largos son: primera, que se tarda más en escribirlos con el teclado, y, segunda, que ocupan más espacio en la memoria del ordenador. Por estas razones, quizás usted se sorprenda escribiendo nombres más cortos, abreviados, tales como ING, o, simplemente, I.

Una vez definida una variable con un determinado nombre, el ordenador reconocerá el nombre exacto que usted haya especificado. Si por accidente escribe mal un nombre de variable, o trata de abreviarlo después de haberlo definido ya, el ordenador no sabrá qué quiere decirle. Por ejemplo, ahora que ha definido la variable INGRESOS, trate de introducir cada una de las líneas siguientes de PRINT:

```
PRINT INGRESOS
PRINT ING
PRINT I
```

Ninguno de estos nombres se puede emplear para acceder a la información de la variable INGRESOS. Después de cada uno de estos mandatos, usted verá el mensaje:

```
2/0
```

en la parte inferior izquierda de la pantalla. El 2 es un código para uno de los mensajes de error del ordenador; significa sencillamente que usted ha usado un nombre de variable no definido. (En el Apéndice B puede ver los códigos de todos los mensajes de error).

En un programa, las variables son los medios de almacenar piezas de información que tendremos que utilizar otra vez. Además de la información numérica, quizás usted también quiera almacenar caracteres, palabras o texto en la memoria del ordenador. Como hemos visto, los elementos de información no numérica son llamados series, o cadenas, y las variables que los contienen se llaman variables en serie o en cadena. Como recordará usted, el nombre de una variable en cadena debe terminar con el carácter "$". A diferencia de los nombres de las variables numéricas, los de variables en cadena deben tener una longitud de dos caracteres, exactamente: una letra y el carácter $.

Aquí tiene unos ejemplos de declaraciones LET que definen las variables en serie:

```
LET A$ = "2344 CALLE SEXTA"
LET C$ = "BERKELEY"
LET S$ = "CA"
```

Entre estas tres declaraciones y luego escriba el mandato:

```
PRINT A$; C$; S$
```

El resultado será:

```
2344 CALLE SEXTA BERKELEY CA
```

imprimido en la parte superior de la pantalla. Esto no le enseña nada nuevo acerca de la declaración de PRINT; ésta puede contener tantos nombres de variables como usted quiera. El ordenador visualizará, simplemente, los contenidos de cada variable de la pantalla. Advierta que cada

variable está separada por un punto y coma en la declaración de PRINT; la cual manda al ordenador que exhiba los contenidos de las variables uno al lado del otro en la misma línea.

Ahora ensaye estas dos declaraciones de PRINT:

```
PRINT "YO VIVO EN"; C$; "."
PRINT "MIS INGRESOS SON $"; INGRESOS; "."
```

Estas líneas le muestran que una declaración de PRINT puede incluir cualquier combinación de elementos; mensajes lineales en serie, nombres de variables numéricas. La primera de estas dos declaraciones debería poner el mensaje:

```
YO VIVO EN BERKELEY
```

en la cima de la pantalla. ¿Qué hace la segunda declaración? Como nuevo experimento, utilice mandatos LET para cambiar el valor de C$ por el nombre de su ciudad y el valor de INGRESOS por el de los que perciba. Luego introduzca estas dos declaraciones de PRINT por segunda vez y vea los resultados.

Hay varias maneras de despejar variables —y los datos que contienen— echándolas fuera de la memoria del ordenador. La más fácil, por supuesto, consiste en cortar la corriente eléctrica; pero si lo hace así, lo perderá todo, incluido el programa en el que esté trabajando. En vez de interrumpir la corriente, puede utilizar la palabra clave:

```
CLEAR
```

que está situada sobre la tecla [X]. Esta tecla echa fuera de la memoria todas las variables y los datos; pero deja intacto el programa, que conserva todas sus líneas intactas. Escriba este mandato ahora, y luego introduzca:

```
PRINT INGRESOS
```

Y obtendrá el mensaje de error 2/0, diciéndole que la variable INGRESOS ya no está definida.

Finalmente, cada vez que pase un programa, utilizando la palabra clave RUN, las variables y los datos que hubieran quedado de pases anteriores del programa quedan eliminados de la memoria.

Una segunda manera de definir variables, además de la declaración LET, es mediante la declaración INPUT. La estudiaremos a continuación.

LEER INFORMACION MEDIANTE EL TECLADO

En el T/S 1000 BASIC, la declaración de INPUT tiene una sola forma:

```
INPUT nombre
```

en la que "nombre" puede ser el de cualquier variable.

Como ha visto usted en el programa de saludo del Capítulo 2, INPUT dice al ordenador que espere unos datos que le serán introducidos mediante el teclado, y que luego los almacene en la variable mencionada en la declaración de INPUT. Esta variable puede ser, ora una todavía no definida, ora una que ya tenga un valor. En este segundo caso, el valor antiguo se pierde, y el nuevo, que la máquina lee en el teclado, ocupa su puesto.

INPUT no se puede utilizar como un mandato inmediato, de manera que escribiremos un programa corto para ensayar con él. Introduzca mediante el teclado las seis líneas siguientes:

```
PRINT AT 21,0; "¿NOMBRE?"
20 INPUT N$
30 PRINT AT 21,0; "¿EDAD?"
40 INPUT E
50 CLS
60 PRINT N$; ",EDAD"; E
```

Cuando pase este programa, verá que la acción se interrumpe dos veces para que usted introduzca información mediante el teclado. Primero escriba su nombre y luego su edad. Cuando haya terminado, aparecerá en la parte alta de la pantalla un mensaje como el siguiente:

JOHN DOE, EDAD 31

Hay varias cosas importantes que se pueden aprender de este programa y de la acción que origina. En primer lugar, ¿qué exhibe exactamente el ordenador en la pantalla cuando está esperando el input del teclado? Si usted vigilase bien durante el pase del programa, observaría dos clases diferentes de visualizaciones en la pantalla. Cuando el ordenador espera una cadena, exhibe la pista L (en vídeo inverso, naturalmente) entre comillas. Cuando espera datos numéricos, en cambio, exhibe solamente la pista L. De este modo distingue el ordenador entre una declaración de INPUT que contenga un nombre de variable en cadena y una que contenga el nombre de una variable numérica.

Las líneas 10 y 30 de este corto programa ilustran un rasgo importante de diseño para una programación clara y eficaz. Siempre que escriba usted programas que tengan declaraciones de INPUT, debe pensar cuidadosamente en la persona que debe usarlos. Esta persona necesitará le digan qué información debe escribir con el teclado cuando la acción se detenga para input. Por esta razón, es una buena idea imprimir un "prompt" ("indicador" o "invitación") en la pantalla siempre que el ordenador está esperando input. Las líneas 10 y 30 son ejemplos de tales indicadores. Advierta que tales líneas colocan sus indicadores o invitaciones (prompts) cerca del fondo de la pantalla. Se hace así para que el indicador aparezca lo más cerca posible del "eco" de la información que se está introduciendo por el teclado.

La línea 50 de este programa despeja la pantalla, a fin de que la línea 60 aparezca en su parte superior. Estudie con cuidado la línea 60; advierta que el mensaje entre comi-

llas (",EDAD") debe suministrar una coma y dos espacios para separar la información de las dos variables:

```
60 PRINT N$;",EDAD";E
```

Para resumir, hemos visto que las variables se pueden definir de una de dos maneras: con la declaración LET, o con la declaración INPUT. En general, no se puede utilizar una variable en un programa si no se ha definido de una de estas dos maneras. (Existe una excepción a esta regla, y la veremos en la sección siguiente del presente capítulo). En cuanto se ha definido una variable y se le ha dado un valor, se puede emplear una declaración de PRINT para exhibir su valor en la pantalla. También se puede *cambiar* el valor contenido en una variable con declaraciones LET e INPUT subsiguientes. Continuaremos investigando este asunto en el transcurso del presente capítulo.

Antes de seguir adelante, echemos una mirada todavía a otro programa más corto que ilustra el empleo de variables. Escriba la palabra clave NEW para eliminar el último programa de la memoria de su ordenador. Ahora escriba las cinco líneas siguientes:

```
200 PRINT AT 21,0; "HORIZONTAL"
210 INPUT H
220 PRINT AT 21,0; "VERTICAL"
230 INPUT V
240 PLOT H, V
```

En realidad, estas líneas forman parte de un programa más largo que hemos ido elaborando sobre la marcha, pero usted puede pasarlas como si formasen un programa completo. Cuando pase el programa, usted verá dos indicaciones de input en la pantalla, primero:

```
HORIZONTAL
```

en cual punto usted debería entrar un número comprendido entre 0 y 63; luego:

VERTICAL

a lo cual usted debería responder con un número entre 0 y 43. Cuando haya entrado los dos números, el programa visualizará un pixel en la pantalla, y precisamente en la dirección que usted ha especificado con los dos números.

Asegúrese de comprender cómo funciona el programa. Las líneas 210 y 230 contienen las instrucciones INPUT, definiendo las variables numéricas H y V respectivamente. Después la línea 240 utiliza los valores de H y V como dirección en un mandato PLOT:

240 PLOT H, V

Las instrucciones PLOT y UNPLOT no requieren números de línea para la dirección del pixel. En cambio, pueden tomar variables para la dirección, siempre que tales variables estén definidas, y siempre que los valores que contengan formen una dirección lícita de pixel.

A continuación investigaremos las declaraciones FOR/NEXT; usted recordará del Capítulo 2 que tales declaraciones se pueden emplear para hacer que el ordenador cumpla unas instrucciones muchísimas veces.

REPETIR INSTRUCCIONES

Un ejemplo de la forma más sencilla de la declaración FOR es:

FOR I 1 TO 20

En esta declaración, a la I se la puede llamar *variable de control,* porque controla el número de veces que la declaración FOR, y las que le siguen se repetirán. El nombre

de la variable de control puede tener una extensión de una sola letra. *No* es necesario definir la variable de control por anticipado mediante una declaración LET o una INPUT. (Esta es la excepción a la regla).

Veamos la declaración FOR en acción. FOR y NEXT no se pueden emplear como mandatos inmediatos, por consiguiente, miraremos un programa corto:

```
10 FOR I=1 TO 20
20 PRINT "REPETICION"
30 NEXT I
```

Recuerde que las líneas FOR y NEXT representan la cima y el fondo de un "bucle". Todas las líneas intermedias se repetirán un número especificado de veces. Aquí está cómo utiliza el ordenador la variable de control. En este caso, para decidir el número de repeticiones:

1. En el comienzo, a la I se le da el valor que aparece justo después del signo igual, que en este caso es 1.
2. Todas las declaraciones situadas entre FOR y NEXT se efectúan.
3. Cuando el ordenador llega a la línea NEXT, aumenta (incrementa) los valores de I en una unidad, y luego se refiere de nuevo a la línea FOR.
4. Si I todavía es *igual o menor* que el número especificado *después* de la palabra TO (20 en este caso) el ordenador repite todas las líneas hasta volver a la NEXT. Si es mayor que el número que aparece después de TO, entonces el bucle se interrumpe.

Escriba estas tres líneas en su ordenador y páselas. La Figura 3.3. muestra los resultados. Como puede ver, la instrucción de PRINT en la línea 20 ha sido cumplida 20 veces. En otras palabras, la variable de control I ha sido incrementada desde 1 hasta 20; después de cada incremento, la instrucción PRINT se efectuaba una vez.

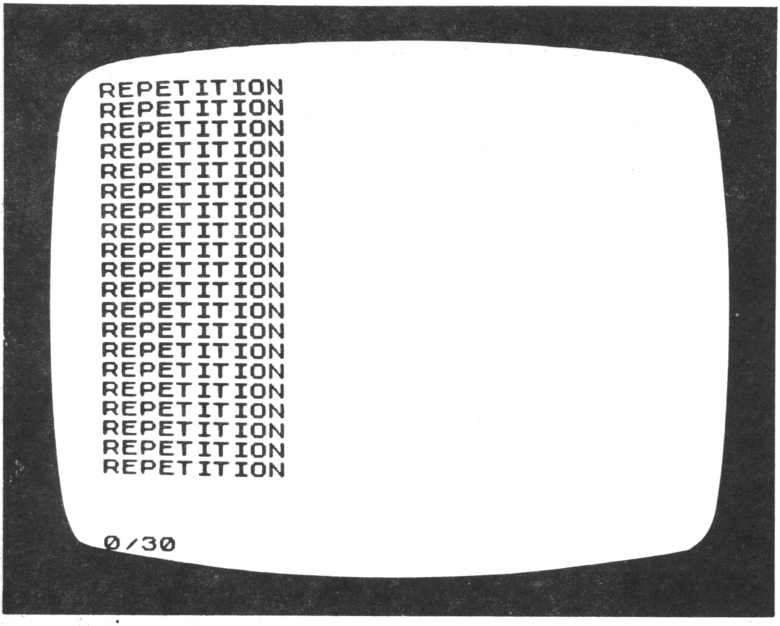

Figura 3.3.: Repetición

Para explorar más aún el concepto de repetición, examinaremos dos variaciones de este programa corto, cada una de las cuales implica el efectuar unos ligeros cambios en la línea 20. Para efectuar estos cambios utilice el carácter EDIT de su ordenador.

Primero, cambie la línea 20 de manera que tenga este aspecto:

```
20 PRINT "REPETICION";
```

Hemos realizado dos cambios: hemos añadido un espacio entre la N y las comillas finales, y hemos añadido un punto y coma al final de la línea. Cuando usted haya efectuado estos cambios, pase el programa de nuevo; el resultado debería ser tal como se ve en la Figura 3.4. este pequeño ejercicio sólo se propone recordarle, simplemente, la función

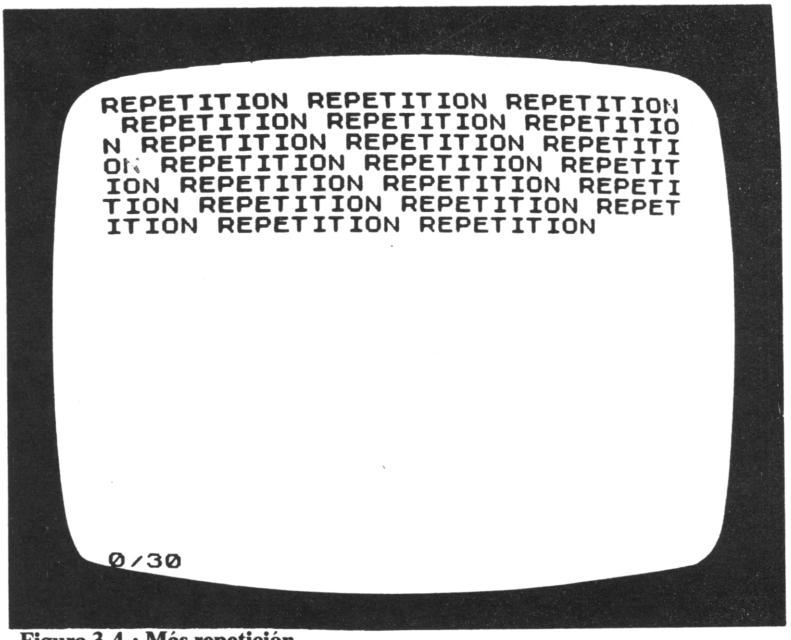

Figura 3.4.: Más repetición

de un punto y coma en una declaración PRINT. El punto y coma le dice al ordenador que *no* pase a una línea nueva. (Solemos decir que el punto y coma impide un *alimento de línea).* Así, cuando una declaración de PRINT *termina* en un punto y coma, la declaración siguiente de PRINT que se realiza empezará allí donde terminó el último mensaje.

Como experimento final, cambie la línea 20 como sigue:

```
20 PRINT AT I,I; "REPETICION"
```

y pase el programa por tercera vez. Los resultados se ven en la Figura 3.5.

Esta última versión del programa ilustra un punto esencial: *La variable de control de la declaración FOR se puede utilizar como una variable en otra clase de declaraciones que aparecen dentro del bucle.* El hecho de que la variable de

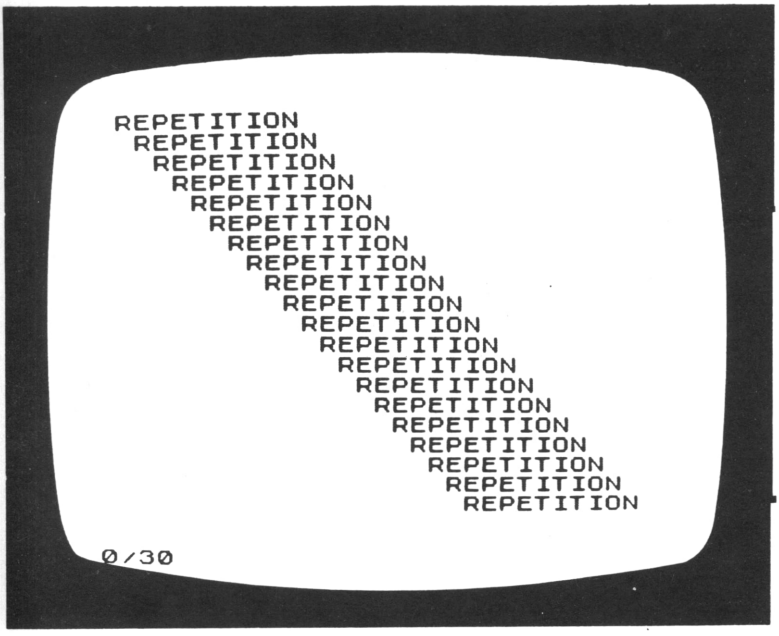

Figura 3.5.: Más repetición aún

control reciba un valor incrementado cada vez que se repite el bucle la convierte en una herramienta extraordinariamente valiosa para ciertas clases de acciones repetitivas. En la Figura 3.5. usted puede ver que la variable de control I determinaba las direcciones para cada actuación de la declaración PRINT AT. Como el valor de I se incrementó de 1 a 20, la declaración PRINT AT se ejecutó como:

PRINT AT 1,1; "REPETICION"
PRINT AT 2,2; "REPETICION"
PRINT AT 3,3; "REPETICION"
PRINT AT 4,4; "REPETICION"

...etc., etc.

Continuemos investigando este último punto un poco más

con otro ejemplo. Revise su programa de tres líneas de forma que tenga el aspecto siguiente:

```
10 FOR I = 20 TO 40
20 PLOT I,I
30 NEXT I
```

En esta versión, emplearemos el valor de la variable de control para determinar las direcciones de pixeles producidos por el mandato de PLOT. Pase el programa, y verá una línea diagonal de pixeles pasando de la dirección 20,20 hasta la 40,40. Advierta que la variable de control I no empieza en 1 en el programa, sino que va incrementada de 20 hasta 40 por la especificación contenida en la declaración FOR.

Cambiando la línea 20 en:

```
20 PLOT I,20
```

usted puede producir una línea horizontal de pixeles. Igualmente, la línea:

```
20 PLOT 20,I
```

producirá una línea vertical de pixeles cuando pase el programa. Usted debería ensayar cada una de estas versiones del programa.

En todos los ejemplos que hemos visto hasta el momento, la variable ha sido incrementada en 1 para cada repetición del bucle FOR. En algunos casos se querrá cambiar la *cantidad de información* hasta cierto número distinto de 1. Esto puede conseguirse con la palabra STEP; por ejemplo:

```
PARA I = 2 TO 40 STEP 2
```

[STEP es un carácter *shift* (de desplazamiento, o conmutado) situado en la tecla [E].]

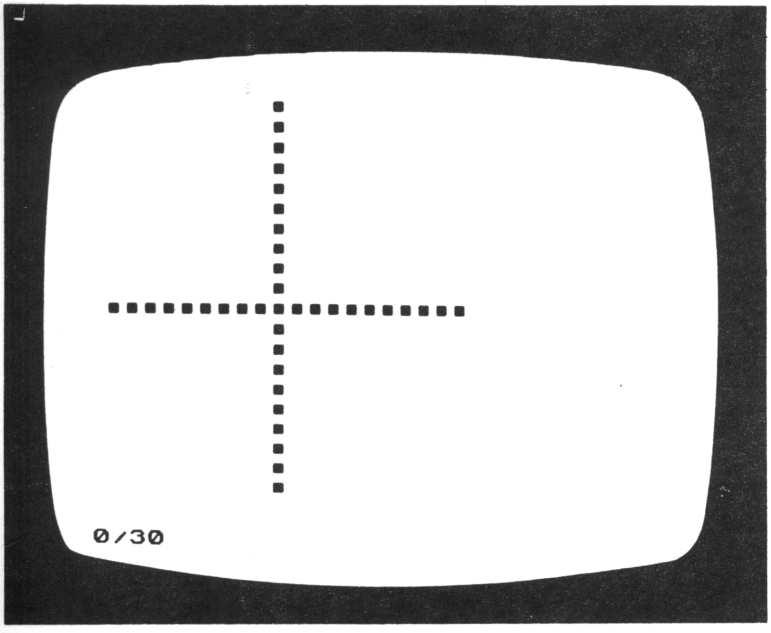

Figura 3.6.: Investigando STEP

Para ver cómo funciona STEP, estudie la versión siguiente de nuestro programa:

```
10 FOR I = 2 TO 40 STEP 2
20 PLOT I,20
25 PLOT 20,I
30 NEXT I
```

En este programa, la variable de control I, tomará los valores 2, 4, 6, 8, ..., 40. Como resultado, las declaraciones PLOT producirán líneas rotas, porque las demás direcciones de lo largo de la línea se saltarán. (Este programa también muestra lo fácil que es añadir líneas a un bucle FOR. La línea 25, la segunda declaración de PLOT produce una segunda línea en la pantalla). Pase el programa. La Figura

3.5. muestra lo que aparecerá en la pantalla. Estudie este programa y sus resultados antes de seguir adelante cuidadosamente. Asegúrese de comprender por qué hace lo que hace.

En todos los programas que hemos examinado hasta el momento, el orden en que se ejecutan las líneas está determinado únicamente por los números de línea. El ordenador empieza con el número de línea menor y va subiendo, línea por línea, hasta haber ejecutado la última del programa. A veces ésta es una manera adecuada de designar el programa, pero con frecuencia surgirán razones imperativas para realizar ciertas líneas fuera del orden. Hablaremos de algunas razones de hacerlo de este modo y en la próxima sección examinaremos las palabras clave que brindan tal posibilidad.

Controlar la Acción

La palabra clave GOTO (situada en la tecla [G]) se puede utilizar para decirle al ordenador que realice una línea fuera de secuencia. A GOTO le suceden, simplemente, los números de línea que se quiere que el ordenador efectúe a continuación. El mandato GOTO puede enviar el control del programa adelante hasta un número de línea mayor que el que viene a continuación:

```
90 GOTO 300
```

o puede enviar el control para atrás, hasta un número de líneas menor:

```
90 GOTO 10
```

El mandato GOTO es sencillo, pero los motivos para utilizarlo son a menudo complicados. Generalmente, conviene usarlo con parquedad en un programa, porque un exceso de GOTO puede hacer casi imposible predecir con alguna

certidumbre los resultados del programa. Pero hay algunas situaciones en las que no se puede evitar el empleo del GOTO.

En los dos programas mayores de este capítulo, usaremos el GOTO para crear un "bucle infinito". El programa, o una determinada parte del mismo, se repetirá una y otra vez; puesto que el programa en cuestión carece de provisiones que le pongan fin jamás. El teclado le proporciona a usted un medio de interrumpir semejante programa artificialmente. Basta, simplemente, con que usted pulse la tecla de BREAK, situada en el ángulo inferior derecho del teclado.

Las líneas siguientes son un ejemplo de bucle infinito:

```
10 PRINT AT 10,13; "FLASH"
20 PRINT AT 10,13; "     "
30 GOTO 10
```

Usted puede seguir la secuencia del programa antes de pasarlo. Primero se cumplen las declaraciones de las líneas 10 y 20, luego la línea 30 envía el control del programa al comienzo nuevamente. Este proceso continúa eternamente. Pase el programa para ver los resultados. Este ejemplo, por supuesto, resulta bastante frívolo; los programas subsibuientes emplearán GOTO con mejor criterio.

Otro mandato BASIC que instruye al ordenador para que salte fuera de secuencia hasta un lugar nuevo del programa es GOSUB. Esta instrucción significa "pasa a una subrutina". Lo mismo que al mandato GOTO, le sigue un número de línea; por ejemplo:

```
GOSUB 200
```

A fin de entender GOSUB, usted ha de saber qué son las subrutinas y para qué se emplean. Con frecuencia, en un programa que realiza cierto número de trabajos diferentes, puede ser preciso ejecutar determinadas tareas, pequeñas y bien definidas, una y otra vez en diferentes puntos del programa. Cuando usted se ponga a escribir un programa así,

reconocerá tales tareas y necesitará una manera conveniente y económica de llevarlas a cabo cuando surja tal necesidad. El BASIC le permite aislar dichas tareas en unidades llamadas subrutinas. Cuando haya escrito usted una subrutina, puede "llamarla" en cualquier punto del programa. La subrutina realizará su trabajo, y luego devolverá el control del programa al punto donde se originó la "llamada".

Veamos un ejemplo. Recuerde el programa corto que hemos examinado y que lee una dirección horizontal y vertical dada con el teclado, y luego utiliza esta dirección para colocar un pixel en la pantalla. Este programa se convertirá en una subrutina de un programa mayor que pronto escribiremos. He aquí el aspecto que tendrá como subrutina:

```
200 PRINT AT 21,0; "HORIZONTAL"
210 INPUT H
220 PRINT AT 21,0; "VERTICAL"
230 INPUT V
240 PLOT H,V
250 RETURN
```

Observe que se ha añadido una línea: una instrucción de RETURN en la línea 250. RETURN es una palabra clave situada sobre la tecla [Y]. Toda subrutina ha de tener un RETURN; la instrucción dice, sencillamente, que la tarea de la subrutina ha quedado completada. Cuando el ordenador encuentra la instrucción de RETURN, devuelve automáticamente el control del programa al lugar donde se llamó a la subrutina.

Para llamar esta subrutina, pues, le basta a usted escribir sencillamente el mandato:

```
GOSUB 200
```

Miraremos dos programas cortos que emplean esta subrutina. Usted debería escribir los dos programas en su ordenador, junto con las líneas de la subrutina, y ensayarlos.

Aquí está el primero:

```
10 PRINT "USTED PUEDE SITUAR 5 PIXELS"
20 FOR I = 1 TO 5
30 GOSUB 200
40 NEXT I
50 STOP
```

Este programa le permite colocar cinco pixeles en la pantalla. Empieza imprimiendo un mensaje en la cima de la pantalla (línea 10) y luego introduce un bucle FOR. La declaración GOSUB de la línea 30 forma parte del bucle. Aquí está la secuencia de acontecimientos generada por este bucle:

1. La declaración FOR utiliza la variable de control I, para determinar el número de repeticiones.
2. La declaración GOSUB envía el control del programa hasta la línea 200, donde se realiza la tarea de input (introducir) un pixel y de situarlo.
3. La última línea de la subrutina (250) RETURN controla el programa nuevamente hasta la línea 40, la declaración de NEXT.
4. NEXT incrementa la variáble de control I, y el bucle continúa.

Por esta secuencia de acontecimientos la subrutina de la línea 200 será llamada cinco veces, permitiéndole situar cinco pixeles en la pantalla; y entonces el programa quedará completo. La línea 50 tiene gran importancia:

```
50 STOP
```

La palabra clave STOP (carácter *shift* situado en la tecla [A]) se usa para decirle al ordenador que el programa ha terminado. En este caso, el cuerpo principal del programa está formado por las líneas que va de la 10 a la 50. Cuando

el bucle FOR ha completado sus cinco repeticiones, usted no quiere que el ordenador vuelva a la línea 200 para realizar nuevamente la subrutina. De modo que es preciso decirle explícitamente al ordenador que pare, papel que desempeña la línea 50. Esta es la primera vez que hemos tenido que utilizar la palabra clave STOP. Hasta el momento hemos visto programas cumplidos línea por línea, desde la primera hasta la última. Cuando el ordenador ejecuta la última línea, desde la primera hasta la última. Cuando el ordenador ejecuta la última línea, se para automáticamente. Pero en este programa las seis líneas últimas (de la 200 a la 250) forman una subrutina, que sólo queremos efectuar por medio del mandato GOSUB.

Aquí está el segundo ejemplo de un programa que utiliza nuestra subrutina:

```
10 PRINT "COLOQUE TANTOS PIXELES COMO
   QUIERA"
20 GOSUB 200
30 GOTO 20
```

Este programa le permite a usted colocar en la pantalla tantos pixeles como quiera. Las líneas 20 y 30 forman un bucle infinito. Ahí verá usted cómo funciona el programa:

1. La línea 20 convoca a la subrutina.
2. La subrutina hace su tarea y luego retorna el control a la línea 30.
3. La instrucción GOTO de la línea 30 envía el control a la línea 20.

Pase el programa para ver cuáles son, exactamente, sus efectos. Puede continuar direcciones de pixel todo el rato que quiera. Después de leer cada dirección, el ordenador colocará un pixel en la pantalla. Para interrumpir el programa, tendrá que introducir la palabra STOP en lugar de escri-

bir una dirección. La tecla BREAK no actúa cuando el ordenador espera para leer input procedente del teclado.

Hemos de hablar de un último asunto antes de pasar a los programas mayores de este capítulo. En la selección siguiente veremos la manera de usar la palabra clave IF (SI, condicional) para instruir al ordenador de que tome decisiones durante el pase de un programa.

Decisiones, Decisiones...

Usted querrá, a menudo, incrementar enormemente el poder y la conveniencia de un programa incluyendo instrucciones que den al ordenador el poder de "decidir" entre dos cursos de acción distintos. Es posible que haya topado ya con un problema en el programa de colocación de pixeles que se pudiera solucionar añadiéndole unas líneas de decisión. Primero veremos el problema, y luego veremos cómo se soluciona.

Pase nuevamente el segundo programa de pixeles. Ya sabe que cuando vea la invitación de

HORIZONTAL

debe introducir un número igual o menor que 63. Del mismo modo, cuando vea la indicación (prompt) de

VERTICAL

ha de entrar un número no mayor de 43. Ello se debe a que la dirección mayor que la instrucción de PLOT puede manejar es la de 63,43. Toda dirección mayor que ésta caería fuera del área de la pantalla. Quizás haya visto ya qué le pasa al programa si accidentalmente usted entra una dirección "ilegal". Si no lo ha visto, ensáyelo ahora. Introduzca el número 64 para la parte horizontal de la dirección, y 44 para la vertical. Ambos números son mayores de los que PLOT puede manejar. Como resultado, no se marcará en la pantalla nin-

gún pixel, y lo que sucederá será que el pase del programa se interrumpirá y aparecerá el mensaje de error:

B/240

en el ángulo inferior izquierdo de la pantalla.

Si mira el Apéndice B, encontrará el significado de este mensaje de error. El código de error "B" significa que el número que ha introducido usted está fuera del conjunto de números aceptables. En este caso, el ordenador define este conjunto, o alcance para la palabra clave PLOT. Advierta que la segunda parte del mensaje de error le da el número de línea —240— donde la actuación del programa se interrumpió. Si mira de nuevo la subrutina verá que, sin duda alguna, la línea 240 es la que contiene el mandato de PLOT.

Para evitar este problema, sería muy conveniente que el ordenador pudiera comprobar los valores input de las variables H y V antes de pasar a cumplir el mandato de PLOT. Si se descubriera que H o V contenían valores demasiado altos para que PLOT pudiera manejarlos, lo ideal sería retornar a las declaraciones de INPUT para que dieran un valor aceptable. Esto es exactamente lo que mandaremos al ordenador que haga, utilizando las declaraciones IF.

Una descripción general de la declaración IF podría ser algo así:

IF (declaración verdadera o falsa) THEN (palabra clave mandato)

Como usted puede ver, la declaración IF contiene dos partes. La primera empieza con la palabra THEN (carácter de desplazamiento situado sobre la tecla [3]). Después de IF viene una declaración que el ordenador dictamina como verdadera o falsa. Esta declaración suele tomar la forma de una igualdad o una desigualdad. Dentro de un momento veremos cómo se escriben tales declaraciones. Después de la palabra

THEN viene un mandato dirigido al ordenador, expresado en forma de una instrucción de palabra clave.

La declaración IF origina uno de dos cursos posibles de acción:

1. Si la declaración que puede ser verdadera o falsa es verdadera, entonces el ordenador ejecuta el mandato expresado después de THEN.
2. Si la declaración es falsa, entonces el ordenador se salta el mandato que sigue a THEN y, sencillamente, pasa a la línea siguiente del programa.

Usted puede ver la potencia de la declaración IF. Por este medio, se puede conseguir que los programas respondan exacta y eficazmente a diferentes situaciones que se producen durante el pase de uno de ellos.

Veamos exactamente cómo escribir la declaración IF. La declaración verdadera o falsa implica el aprender la manera de utilizar una nueva serie de símbolos del teclado. Son los de igualdad y desigualdad:

(es igual a)
(es menor que)
(es mayor que)
(es menor o igual que)
(es mayor o igual que)
(no es igual a)

Todos estos símbolos son caracteres de desplazamiento *(shift)* situados, respectivamente, en las teclas [L], [N], [M], [R], [Y], y [T]. Dichos símbolos se emplean para comparar dos valores separados, que pueden ser numéricos o no numéricos. He aquí algunos ejemplos de declaraciones que utilizan estos símbolos:

I 5
K J

```
M  16
S$ "SI"
H  15
```

Todos los ejemplos citados comparan bien los valores de dos variables diferentes, o el valor de una variable y un valor literal. Todas estas declaraciones son verdaderas o falsas. Por ejemplo, en la primera de estas declaraciones, si la variable I contiene el valor 4, la declaración es falsa, porque 4 no es mayor que 5. Si I contiene el valor 7, la declaración es verdadera, porque 7 es mayor que 5.

He aquí unos ejemplos de declaraciones IF que utilizan las declaraciones verdadero-o-falso:

```
IF I  5 THEN PRINT "I ES DEMASIADO GRANDE"
IF K  J THEN INPUT J
IF M  16 THEN GOTO 200
IF S$  "SI" THEN STOP
IF H  15 THEN LET H  100
```

Advierta que en cada caso, a THEN le sigue una palabra clave. Todas las palabras clave del teclado se pueden usar después de THEN en una declaración IF, aunque algunas se utilicen más corrientemente que otras. Tenga bien presente que la instrucción que sigue a THEN *sólo* se ejecuta si la declaración verdadero-o-falso resulta ser *verdadera*. De lo contrario, el ordenador pasa, simplemente, a la línea siguiente del programa.

Ahora, para unos cuantos ejemplos significativos de IF, volvamos al problema que esbozábamos antes. La solución del problema requiere un perfeccionamiento de la subrutina que empieza en la línea 200. Cuando el ordenador lee cada valor procedente del teclado y asigna los valores a las variables H y V, respectivamente, nos gustaría añadir una *prueba* para asegurarnos de que ambos valores están dentro del campo que PLOT puede manejar. Podemos expresar fácilmente

esta prueba bajo la forma de una declaración IF situada directamente después de cada declaración de INPUT.

Aquí está la primera:

```
210 INPUT H
215 IF H  63 THEN GOTO 210
```

La línea 210 lee un valor para H. La línea 215 evalúa la declaración H 63 antes de decidir qué hará a continuación. Si la variable H contiene un valor mayor que 63, entonces se ejecuta la declaración GOTO. El control del programa regresa a la línea 210 para un nuevo valor de input. Si, en cambio, el valor H es menor o igual que 63, entonces el mandato GOTO queda sin efecto y el programa sigue normalmente hacia la declaración siguiente.

Podemos escribir una declaración IF similar directamente después de la declaración de INPUT para V. Este segundo IF, en la línea 235, evalúa la declaración V 43, y envía el control hacia la línea 230 nuevamente si V 43 es cierto.

La subrutina entera la presentamos en la Figura 3.7. Estúdiela cuidadosamente, y asegúrese de entender cómo funcionan las declaraciones IF. Luego ensaye con uno de los dos programas cortos que hemos descrito antes. En particular, cuando el programa esté en marcha, trate de escribir unos valores incorrectos para las partes horizontal y vertical de la dirección. Ahora, en lugar de terminar el programa, esta clase de error de input motivará, simplemente, que se rechace su valor. Por ejemplo, cuando ve la indicación (prompt):

```
HORIZONTAL
```

pruebe de escribir el número 66. Verá entonces que la invitación sigue siendo la misma, indicando que el 66 no sirve como dirección horizontal. Tiene que escribir otro valor.

Como repaso de todo lo aprendido por usted hasta el momento en este capítulo, miraremos ahora el programa de

gráficos que dibuja rectángulos en la pantalla. Nos referiremos al mismo como al "programa del rectángulo".

El programa del rectángulo

El cuerpo principal del programa lo mostramos en la Figura 3.8. El programa utiliza la subrutina que se ve en la figura 317, de modo que usted tendrá que incluir esas líneas cuando escriba su programa. Ahora entre todo el programa en el ordenador y páselo.

Este programa es, sencillamente, un ejercicio destinado a ilustrar un número de herramientas de programación en funcionamiento. A pesar de todo, puede emplearlo para producir unas imágenes interesantes en la pantalla. Vea có-

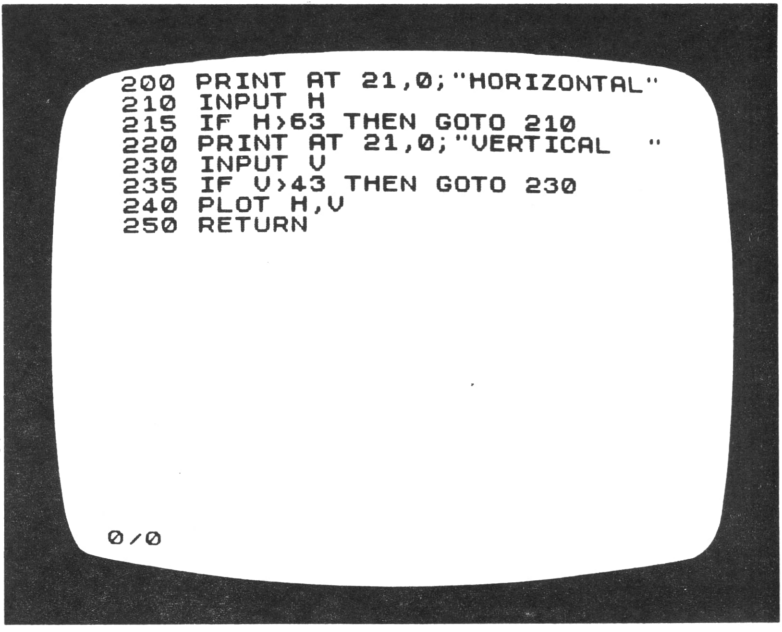

Figura 3.7.: La subrutina pixel, incluyendo declaraciones IF

mo procede: Empieza invitándole a usted a introducir dos direcciones pixel completas. En otras palabras, usted verá las invitaciones HORIZONTAL y VERTICAL dos veces, y el ordenador aguardará hasta que usted haya introducido un número después de cada invitación. Tan pronto como usted haya introducido una dirección completa y válida, aparecerá un pixel en la pantalla. Una vez determinados dos pixeles, el ordenador trazará un rectángulo en la pantalla, utilizando dichos pixeles como los dos cabos de una diagonal del rectángulo. Este proceso continúa todo el tiempo que usted quiera, y le permite trazar en la pantalla muchísimos rectángulos de formas y tamaños distintos. La Figura 3.9. nos presenta una muestra producida por este programa. Para terminar el programa, introduzca la palabra clave SOP.

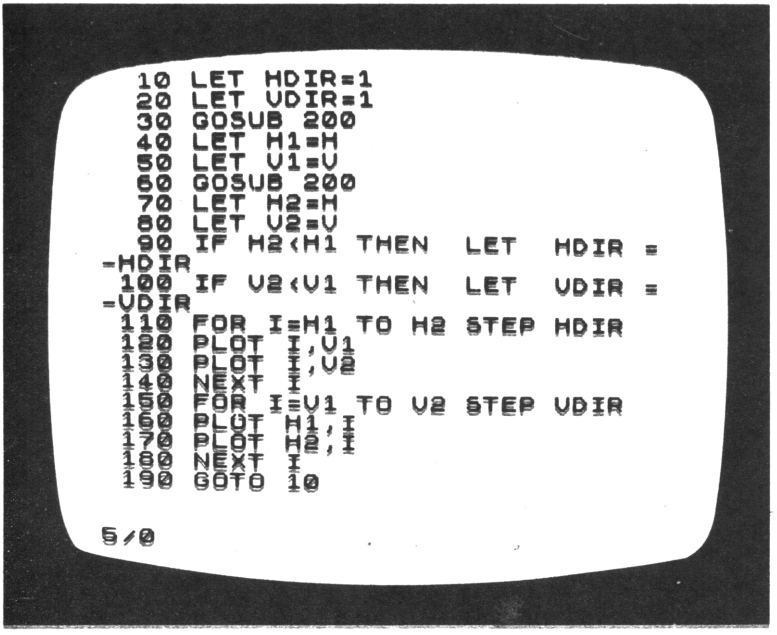

Figure 3.8: The Rectangle

Mientras usted experimenta con este programa, advierta que posee la facultad de dibujar los rectángulos en distintas direcciones. Para ver qué significa esto, empiece introduciendo las direcciones 5,5 y 25,25. Dado que la primera dirección es más baja que la segunda, y a la izquierda de esta segunda, el ordenador dibuja la cima y el fondo del rectángulo de izquierda a derecha, y sus lados, desde el fondo hacia lo alto. Ahora introduzca dos direcciones más: 45,43 y 25,23, en este orden. En este caso, la primera dirección que introdujo está arriba y hacia la derecha de la segunda, y en consecuencia el rectángulo es dibujado en direcciones contrarias a las del primero. La cima y el fondo han sido trazados de derecha a izquierda, y los lados desde la cima hacia el fondo. Esto puede parecer un detalle sin importancia, y sin embargo es una característica del programa cuidadosamente ideada. Examinemos las líneas del programa. Para mirar el listado de su pantalla de usted, entre STOP a fin de interrumpir el pase del programa, y luego apriete nuevamente ENTER. El programa es demasiado largo para que se visualice todo a la vez en la pantalla, pero recuerde que puede emplear la palabra clave LIST para contemplar diferentes partes del programa. Por ejemplo, si quisiera ver la subrutina de la línea 200, escribiría, simplemente:

```
LIST 200
```

Las dos primeras líneas del programa definen las variables HDIR y VDIR:

```
10 LET HDIR  1
20 LET VDIR  1
```

Los nombres de estas variables significan "dirección horizontal" y "dirección vertical", respectivamente. Durante el pase del programa, cada una de ellas contendrá el valor 1 o el valor —1, y, en consecuencia, se utilizarán para determinar la dirección en que se dibuje el rectángulo. Veremos la mane-

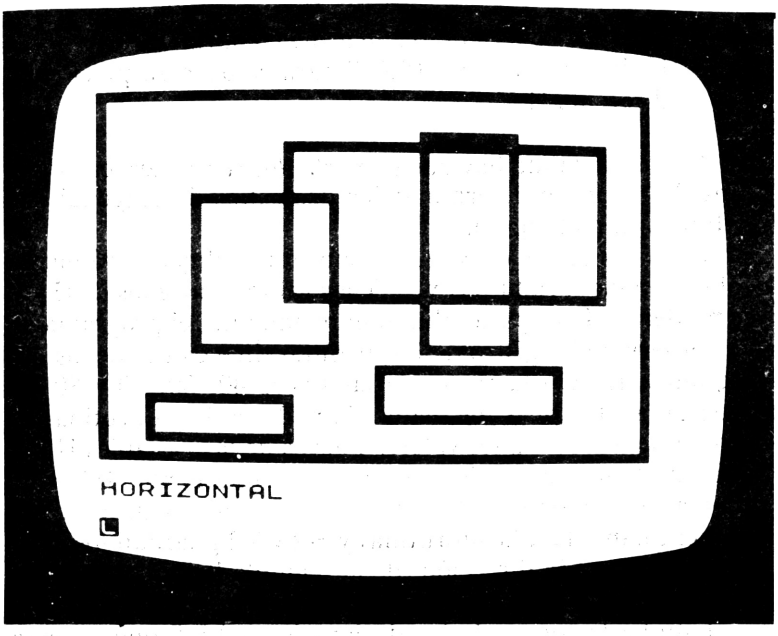

Figura 3.9: Resultados del programa del rectángulo

ra de utilizarlas, al contemplar las líneas subsiguientes del programa.

Las líneas 30 y 60 contienen llamadas a la subrutina de la línea 200:

```
GOSUB 200
```

Recuerde qué hace esta subrutina:

1. Imprime la invitación (prompt) HORIZONTAL en la pantalla y luego lee un valor (no mayor de 63) procedente del teclado. Este valor lo almacena en la variable H.
2. Imprime la invitación VERTICAL en la pantalla y luego lee un valor (no mayor de 43) procedente del

teclado. Este valor lo almacena en la variable V.
3. Utiliza el mandato PLOT para colocar un pixel en la dirección H,V.

A esta subrutina hay que llamarla *dos veces* para cada rectángulo, a fin de determinar los dos cabos de diagonal que definirán el rectángulo.

Cada vez que se convoque la subrutina, almacenará nuevos valores en las variables H y V. Y los valores antiguos de H y V se perderán. Pero para dibujar el rectángulo, el programa ha de conservar el registro de las distintas direcciones de las dos esquinas. He ahí la razón de las líneas 40, 50, 70 y 80 del programa. Después de cada llamada a la subrutina, el programa almacena los valores nuevos de las variables H y V a fin de "guardar" tales valores. Concretamente, H1 y V1 ahorran los valores de la primera dirección (es decir, de la primera llamada a la subrutina) y H2 y V2 guardan los valores de la segunda dirección (de la segunda llamada a la subrutina).

Estudie la forma de las declaraciones LET que asignan valores a las variables H1, V1, H2 y V2. Por ejemplo, fíjese en el primero:

```
40 LET H1 = H
```

Esta declaración, parafraseada, dice: Coge el valor actual de la variable H y almacénalo en la variable H1. Esta es la primera vez que hemos visto un nombre de variable a la derecha del signo igual en una declaración LET. Importa mucho que usted comprenda qué hace y *qué no hace* una declaración LET. Lo que hace es definir la variable H1 y darle un valor. Pero *no* afecta a la variable H en modo alguno. Después de cumplido el mandato LET, la variable H seguirá teniendo el mismo valor que antes.

Una vez determinadas estas dos direcciones, el ordenador deberá decidir en qué direcciones traza los rectángulos. Las líneas 90 y 100 controlan esta decisión. Recuerde que,

inicialmente, tanto la HDIR como la VDIR se ponen en el valor de 1. Como veremos, este valor indica que el rectángulo será trazado de izquierda a derecha y desde arriba para abajo. De modo que el objetivo de las líneas 90 y 100 es *cambiar* estas direcciones, si fuese necesario. Si el segundo valor horizontal, H2, es menor que el primero (es decir, situado a la izquierda del primero) el rectángulo se trazará de derecha a izquierda. Esto se indica cambiando el valor de HDIR a —1:

```
90 IF < H2  H1 THEN LET HDIR = —HDIR
```

Fíjese en el mandato LET que aparece después de la palabra THEN. La variable HDIR aparece en *ambos* lados del signo igual. Esta declaración LET dice: "Toma el *negativo* del valor actual almacenado en HDIR, y almacena este valor nuevo en HDIR". En otras palabras, dado que el valor original de HDIR era 1, el nuevo valor será —1. Por supuesto, el valor original se ha perdido.

A continuación hay dos bucles FOR en el programa. El primer bucle (líneas 110 a 140) traza las líneas de la cima y del fondo del rectángulo, y el segundo bucle (líneas 150 a 180) traza los costados. Al estudiar la manera de construir estos bucles, comprenderá usted el significado pleno de las variables de dirección, HDIR y VDIR.

Cada bucle contiene dos mandatos PLOT. Hemos visto ya cómo dos mandatos PLOT dentro de un bucle FOR pueden producir dos líneas separadas. En este caso, producen líneas paralelas. En el primer bucle, la variable I se incrementa desde el valor de H1 hasta el de H2:

```
110 FOR I = H1 TO H2 STEP HDIR
```

Las declaraciones PLOT subsiguientes tienen direcciones verticales fijadas en V1 y V2, respectivamente; y utilizan el valor variable de I como sus direcciones horizontales:

```
120 PLOT I,V1
130 PLOT I,V2
```

Por esta razón usted ve que las líneas de la cima y del fondo del rectángulo se trazan primero cuando se pasa el programa.

De la misma manera, el segundo bucle FOR traza los costados del rectángulo manteniendo las direcciones horizontales, H1 y H2, fijas y variando las verticales desde V1 hasta V2:

```
150 FOR I=V1 TO V2 STEP VDIR
160 PLOT H1,I
170 PLOT H2,I
180 NEXT I
```

Bien, ¿qué hay de la cláusula STEP de estas dos declaraciones FOR? La manera más fácil de ver por qué son necesarias consiste en hacerse la idea de introducir dos direcciones de pixel en las que la primera esté más arriba y más hacia la derecha que la segunda; por ejemplo:

dirección # 1: 45,43
dirección # 2: 25,23

Ahora substituya estos valores en sendas declaraciones FOR y vea que le dan:

```
110 FOR I = 45 TO 25
150 FOR I = 43 TO 23
```

Resulta que estos dos bucles FOR, tal como aparecen arriba, jamás cumplirán la instrucción ni una sola vez, porque el primer número de las declaraciones FOR es mayor que el segundo. A fin de hacer que estos bucles funcionen, tenemos que añadir una cláusula STEP que originará un *decrecimiento* (una *disminución*) de la variable de control, I, cada vez que el bucle se repita. Por este motivo las variables de

dirección, HDIR y VDIR, han de ponerse a −1 si la primera dirección es mayor que la segunda. En consecuencia, los bucles FOR para las direcciones antedichas serían:

```
110 FOR I = 45 TO 25 STEP -1
150 FOR I = 45 TO 23 STEP -1
```

La cláusula STEP especifica que la variable de control disminuirá en una unidad cada vez que se repita el bucle.

Si estos detalles todavía parecen un tanto desorientadores, pase al programa de nuevo, y piense en los dos bucles FOR mientras ve aparecer los rectángulos en la pantalla. Dibuje varios rectángulos, y observe las direcciones diferentes en que se dibujan. Ahora sabe que tales direcciones están determinadas por el orden en que se introducen las direcciones pixel.

La última línea del cuerpo principal de este programa (es decir, antes del comienzo de la subrutina) es la línea 190:

```
190 GOTO 10
```

Esta línea crea un bucle infinito. Usted puede seguir dibujando rectángulos todo el tiempo que quiera. El programa en sí no tiene provisión alguna para detenerse, por cuyo motivo usted tiene que emplear el mandato de STOP mediante el teclado a fin de terminar el pase del programa.

El programa final de este capítulo es también un programa de gráficos, aunque más interesante que el programa de los rectángulos. Nos referiremos a este programa final como "el programa del dibujo", porque está destinado a ayudarle a trazar dibujos en la pantalla. Empezaremos con instrucciones para utilizar el programa, a fin de que usted pueda escribirlo y empezar a divertirse con él inmediatamente. (El programa es largo, por consiguiente, acuérdese de copiarlo en una cassette inmediatamente después de haberlo introducido).

También encontrará una explicación línea por línea de

cómo funciona el programa. Incluiremos tal programa en el presente capítulo a fin de que usted profundice y amplíe su comprensión de todas las instrucciones de BASIC que aprendió hasta el momento actual. Los métodos y técnicas ilustrados en el programa de dibujos son algo más adelantados que todo lo que aprendió usted hasta ahora. Puede descubrir que tiene que detenerse en este programa algún tiempo antes de entender plenamente cómo funciona. Pero no se precipite. Puede gozar con el programa aun cuando no haya roto por completo la cáscara de su lógica interna.

EL PROGRAMA DE DIBUJOS

El listado del programa se ve en las Figuras 3.10., 3.11. y 3.12. Como puede ver usted, el programa es demasiado largo para visualizarlo todo de una vez en la pantalla. Cuando lo introduzca mediante el teclado, llenando la pantalla de líneas de instrucciones, encontrará que el ordenador imprime y reimprime continuamente el listado del programa en la pantalla. Siempre ajusta la porción del programa que aparece en la pantalla de forma que la línea del momento quede visualizada. Este proceso de ajuste puede resultar bastante lento cuando uno escribe un programa tan largo. Para acelerar el proceso, puede poner el ordenador en modalidad rápida. Sencillamente, introduzca la palabra clave FAST como mandato inmediato. Si lo hace, verá que el ordenador invierte mucho menos tiempo ajustando el listado del programa. Cuando haya escrito el programa, y antes de pasarlo, retorne el ordenador al modo lento, entrando la palabra clave SLOW de nuevo como mandato inmediato.

Compruebe todas las líneas del programa para asegurarse de haberlas escrito correctamente. Luego introduzca el mandato de RUN para pasar el programa. La Figura 3.13. muestra la pantalla tal como aparece inicialmente, cuando lo pone en marcha por primera vez. En el centro de la pantalla hay un sólo pixel, y en la parte superior aparecen unas breves instrucciones. Usted puede trasladar este pixel hacia cual-

quier parte de la pantalla, en cualquiera de ocho direcciones: arriba, abajo, izquierda o derecha; y en las direcciones diagonales descritas como NO, NE, SE, SO. La palabra de la parte superior izquierda de la pantalla siempre describe la dirección actual. (En el comienzo del programa la dirección es "UP" —"arriba"—). Se puede cambiar la dirección apretando una de las teclas de números, desde 1 a 8. Para las direcciones diagonales, apriete las teclas de 1 a 4. Se puede leer las direcciones en los caracteres gráficos de las teclas en cuestión:

1: NO
2: NE
3: SE
4: SO

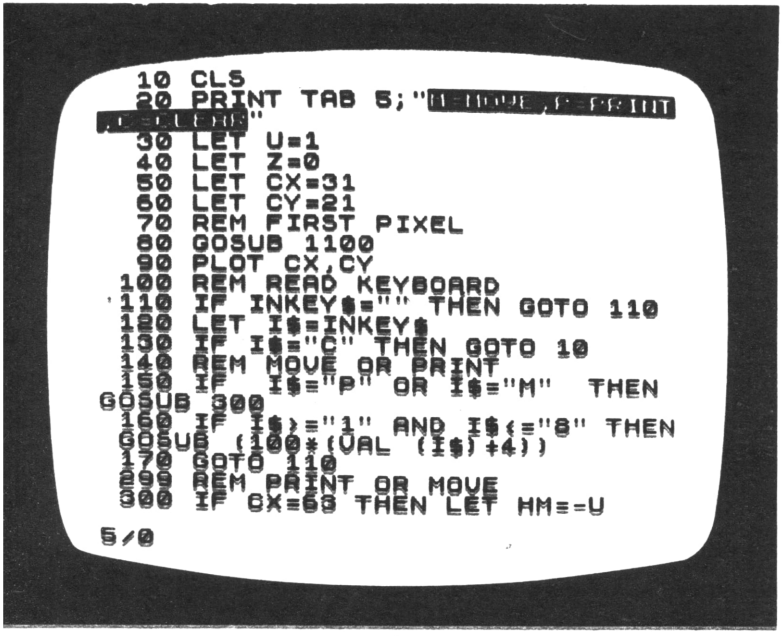

Figura 3.10: El programa de dibujo

Las otras direcciones están indicadas por las flechas que se ven en las teclas de la 5 a la 8:

5: izquierda
6: abajo
7: arriba
8: derecha

Pruebe de apretar todas estas ocho teclas, una tras otra. Cada vez que apriete una, la palabra de la parte superior izquierda de la pantalla cambiará.

Cuando haya elegido la dirección en que quiera mover, podrá trasladar el pixel apretando una de dos teclas. La tecla [M] mueve, simplemente, el pixel. Apriete la tecla una vez para moverlo un punto en la dirección indicada, o tenga

```
310 IF CX=U THEN LET HM=U
320 IF CY=U THEN LET VM=U
330 IF CY=40 THEN LET VM=-U
340 IF I$="M" THEN UNPLOT CX,CY
350 LET CX=CX+HM
360 LET CY=CY+VM
370 PLOT CX,CY
380 RETURN
499 REM DIRECTION SUBROUTINES
500 LET HM=-U
510 LET VM=U
520 PRINT AT Z,Z;"NW"
530 RETURN
600 LET HM=U
610 LET VM=U
620 PRINT AT Z,Z;"NE"
630 RETURN
700 LET HM=U
710 LET VM=U
720 PRINT AT Z,Z;"SE"
730 RETURN
800 LET HM=-U

5/0
```

Figura 3.11.: El programa de dibujos (cont.)

el dedo fijo y apretado sobre la tecla [M] para obtener un movimiento continuo.

La tecla [P] (para "print") deja un rastro de pixeles detrás cuando el pixel se mueve. También ahora puede escoger entre pulsar la tecla para lograr un movimiento cada vez o puede tenerla apretada continuamente para cierto número de movimientos. Tómese ahora un momento para experimentar con ambas teclas en diferentes direcciones.

Puede borrar cualquier pixel de la pantalla, moviéndola hacia atrás sobre ella. Aprieta la tecla de dirección acertada, hacia el pixel que quiera borrar, y luego apriete la tecla [M].

Si quiere empezar de nuevo con una pantalla limpia, apriete la tecla [C] para borrar ("clear") el dibujo que esté haciendo.

Cuando el pixel en movimiento llega a uno de los costa-

```
 810 LET VM=-U
 820 PRINT AT Z,Z;"SU    "
 830 RETURN
 900 LET HM=-U
 910 LET VM=Z
 920 PRINT AT Z,Z;"LEFT  "
 930 RETURN
1000 LET HM=Z
1010 LET VM=-U
1020 PRINT AT Z,Z;"DOWN  "
1030 RETURN
1100 LET HM=Z
1110 LET VM=U
1120 PRINT AT Z,Z;"UP    "
1130 RETURN
1200 LET HM=U
1210 LET VM=Z
1220 PRINT AT Z,Z;"RIGHT"
1230 RETURN


0/0
```

Figura 3.12.: El programa de dibujos (cont.)

dos de la pantalla, será "reflejado" automáticamente hacia el interior. Si se mueve en dirección diagonal, y el ángulo de reflexión será 90°. Si se mueve para arriba, para abajo, hacia la derecha o hacia la izquierda, será reflejado en dirección opuesta.

Descubrirá usted que cuanto más tiempo juegue con este programa, mejores serán los resultados. Un programa que podrá utilizar para dibujar casi todo lo que quiera en la pantalla. Por supuesto, la limitación principal consiste en la resolución, determinada por el tamaño del pixel. La Figura 3.14. muestra un dibujo sencillo trazado con este programa.

Interior del programa

Pulse la tecla BREAK para interrumpir el programa;

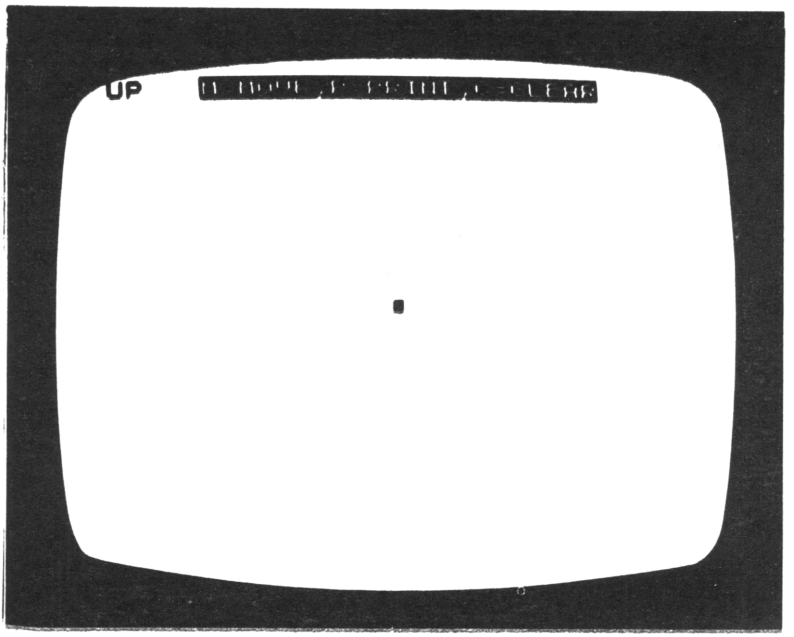

Figura 3.13.: Iniciando el programa de dibujos

luego pulse ENTER para listar la línea del programa a la pantalla. Este programa está compuesto de una sección controladora principal y de cierto número de subrutinas. Primero veremos cómo funciona la sección controladora principal junto con las subrutinas; luego observaremos detalladamente algunas líneas del programa.

Este programa demuestra el tremendo poder de las subrutinas. Las tareas principales que debe efectuar están organizadas en sus propias subrutinas individuales. El resultado es que el programa se entiende fácilmente y se corrige o revisa con la misma facilidad. En cuanto usted conozca la tarea de cada subrutina, conocerá exactamente a dónde debe mirar si quiere cambiar algún detalle del programa.

En pocas palabras, he ahí cómo está organizado el programa. Ocho subrutinas de su última parte están encarga-

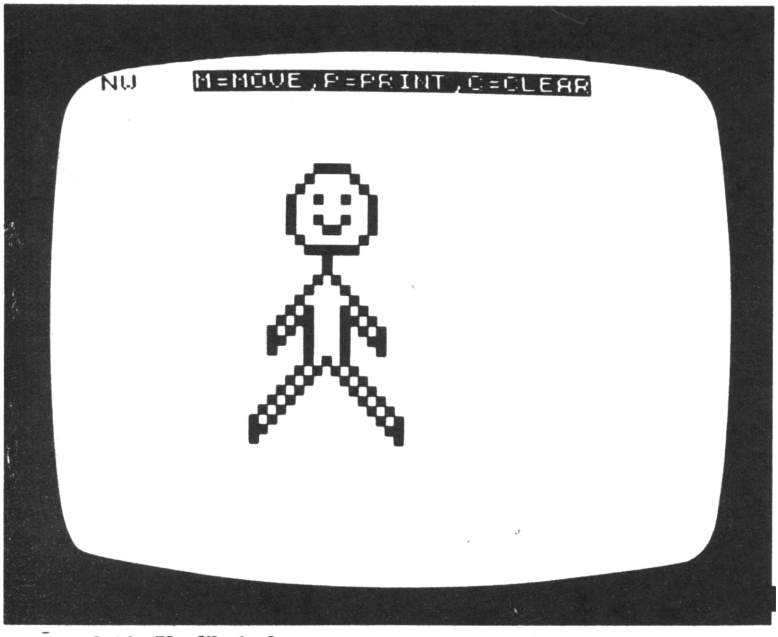

Figura 3.14.: Un dibujo de muestra

das de la tarea de cambiar la dirección del movimiento. Cada vez que usted aprieta una de las teclas de dirección, convoca una de las subrutinas. Estas subrutinas controlan los valores de dos variables, llamadas HM (abreviación, inglesa, de "movimiento horizontal") y VM (significando "movimiento vertical"). A su vez, estas dos variables determinan la dirección que sigue el pixel cuando se aprieta la tecla [P] o la tecla [M]. Las variables HM y VM siempre tienen uno de estos tres valores: 1, 0 ó —1. Además de controlar estas variables, las subrutinas de dirección imprimen la dirección actual en el ángulo superior izquierdo de la pantalla. Cada una de estas subrutinas sólo tiene cuatro líneas de longitud, y cada una empieza con un número de línea múltiplo de 100 (500, 600, 700...). Este es un hecho muy importante para la declaración GOSUB que llama a las subrutinas, como veremos muy pronto.

La subrutina que mueve el pixel está localizada en la línea 300. Es más larga y complicada que las subrutinas de dirección. Primero debe comprobar si la posición actual del pixel en movimiento se encuentra en uno de los cuatro bordes de la pantalla. Si se encuentra, la subrutina debe ajustar una de las direcciones HM o VM para invertir la dirección del movimiento. Luego la subrutina debe determinar si el pixel se mueve nada más, o si deja un rastro detrás (es decir, si uno está apretando la tecla [M] o la tecla [P]). Si el pixel se mueve nada más, la posición anterior en la pantalla debe borrarse (con el mandato de UNPLOT). Finalmente, esta subrutina determina la nueva dirección del pixel, añadiendo HM y VM a la dirección actual y traza (PLOTs) la posición nueva.

Las líneas desde la 110 a la 170 leen el teclado y llaman a las subrutinas correctas, dependiendo de la tecla que se haya apretado. Estas líneas forman un bucle infinito, como se puede ver en la línea 170:

```
170 GOTO 110
```

El resultado es que el ordenador se pasa mucho tiempo aguardando sencillamente que uno apriete una tecla del teclado. En este programa, a diferencia de los que usted ha visto hasta ahora, la tecla de INPUT no se emplea para leer lo escrito en el teclado. En su lugar se emplea la función programadora llamada INKEY$ (situada debajo de la tecla [B] del teclado). Veremos cómo difiere INKEY$ de INPUT cuando hablemos de estas líneas en detalle.

Finalmente, como hemos retrocedido a través del programa, las líneas de la 10 a la 90 son lo que podríamos llamar la sección de "iniciación" del programa. La línea 10 despeja la pantalla (con el mandato CLS). La línea 20 imprime la línea de instrucción de vídeo inverso en la cima de la pantalla, utilizando la función TAB para adelantar los cinco espacios del mensaje desde el borde izquierdo de la pantalla. Las líneas de la 30 a la 60 definen (o, podríamos decir, *inician*) varias variables muy importantes; muy bien podríamos empezar nuestra discusión detallada del programa con estas variables.

Las líneas de la 30 a la 40 definen las variables U (de "unidad") y Z (de "zero", equivalente inglés del castellano "cero"):

```
30 LET U = 1
40 LET Z = 0
```

La razón de la existencia de estas variables es una historia más bien larga. Tiene más que ver con la naturaleza del ordenador de usted que con este programa particular. Como sabe, el ordenador tiene una cantidad limitada de memoria. Este programa está diseñado para encajar dentro de —y operar con — los 2K de memoria originarios del ordenador. Cuando uno empieza a meterse en programas relativamente largos como éste —especialmente si emplean mucho la pantalla del televisor al ejecutarlos— empieza también a buscar maneras de conservar espacio de memoria. No olvide que el ordenador ha de utilizar su memoria para varias tareas

diferentes, cuando se está pasando un programa. Parte de la memoria se reserva para almacenar las instrucciones del programa propiamente dichas, otra parte se requiere para tener constancia de la información que se envía a la pantalla, y todavía se necesita más memoria para otras tareas menos visibles. Una manera obvia de conservar memoria consiste, pues, en utilizar la menor cantidad posible para el programa propiamente dicho. En el programa de dibujos ahí es donde intervienen las variables U y Z.

El ordenador necesita mucha menos memoria para almacenar un carácter que para almacenar un número. El ordenador posee un sistema especial de almacenar números (que, por si usted quiere investigar algo más, se llama *binario de punto flotante*) ideado para guardarlos lo más exactamente posible en una cantidad de memoria relativamente pequeña. Naturalmente, la exactitud constituye un elemento importante cuando uno se pone a emplear el ordenador para cálculos matemáticos. Este sistema ocupa muchísima más memoria para cada número literal de la que necesita para un carácter. De manera que cada vez que en el listado del programa aparece un número literal, usted debería preguntarse si existe la posibilidad de reemplazar dicho número por una letra.

En el programa de dibujos, los números 0 y 1 se utilizan muchas veces. Sabiendo esto, podemos recurrir a una treta limpia y sencilla para ahorrar un montón de espacio de la memoria. Podemos asignar los valores de 0 y 1 a dos variables en el comienzo del programa, y luego utilizarlas a todo lo largo del programa para representar dichos valores 0 y 1. Esto es, exactamente, lo que hacen las líneas 30 y 40. Como resultado, cuando usted lea este programa, siempre que vea las variables llamadas Z y U ha de pensar en los valores "constantes" 0 y 1. Esto quizás le dificulte un poco el descifrar el programa, pero al final vale la pena. Al fin y al cabo, usted posee una capacidad de memoria mucho mayor que el ordenador.

Las otras dos variables definidas cerca del comienzo del

programa no necesitarán una explicación tan larga, ni mucho menos. Las variables CX y CY representan la posición horizontal y la posición vertical del momento, respectivamente. (Si posee usted alguna cultura matemática, sabrá que las letras X e Y son empleadas tradicionalmente como nombres de variables para las situaciones horizontal y vertical en un gráfico). Las líneas 50 y 60 inicializan estas dos variables:

```
50 LET CX 31
60 LET CY 21
```

Esto significa que el pixel que usted ve en el centro, aproximadamente, de la pantalla cuando empieza el programa está en la dirección 31,21.

La línea 70 es la primera de varias líneas REM que aparecen en este programa. La palabra clave REM es la abreviación de "remark" (que en castellano significaría "observación", "comentario"...) (Está situada encima de la tecla [E] de su teclado). Las líneas REM no originan ninguna acción del ordenador, el cual, durante el pase del programa, las ignora por completo. La utilidad de las líneas REM está en la oportunidad que le ofrecen a usted de "documentar" su programa dentro del propio listado. Después de la palabra clave REM, usted puede escribir lo que quiera. Habitualmente, escribirá unas cuantas palabras cortas —del lenguaje que usted emplee, no del BASIC— describiendo lo que sucede en determinado punto del programa. Luego, cada vez que mire el listado del programa, las líneas REM estarán allí para ayudarle a comprender lo que hace el programa.

Desgraciadamente, las líneas REM ocupan espacio en la memoria del ordenador. Por esta razón, debe ser parco su empleo. Además, si ve, mientras elabora un programa, que le va a faltar memoria de ordenador, siempre puede proceder a borrar (o condensar) las líneas REM que haya escrito. Pero en general descubrirá que un comentario bien situado y bien redactado, en forma de línea REM, puede ahorrarle un sin-

fín de tiempo cuando trate de revisar, corregir o, sencillamente, comprender un programa.

La sección de inicialización del programa tiene dos líneas más. La línea 80 inicializa la dirección del movimiento llamando una de las subrutinas de dirección:

```
80 GOSUB 1100
```

La subrutina de la línea 1100 es la que establece la dirección hacia "UP" ("arriba").

Finalmente, la línea 90 traza el pixel inicial:

```
90 PLOT CX,CY
```

Este es el pixel que está en el centro de la pantalla al empezar a pasar el programa.

La sección de iniciación del programa se ejecuta cuando el programa empieza, y de nuevo siempre que uno aprieta la tecla [C] para despejar la pantalla. La línea 130 emplea una declaración GOTO para "reinicializar" el programa para un carácter input de "C":

```
130 IF I$ = "C" THEN GOTO 10
```

La línea 110 lee el teclado, utilizando el mandato INKEY$. Cada vez que se ejecuta INKEY$, el ordenador "escudriña" el teclado una vez para ver si usted está apretando una tecla. Si usted está apretando exactamente una tecla, una sola, INKEY$ devuelve el carácter representado por aquella tecla: una letra, entre la "A" y la "Z", o un dígito, de "0" a "9". Advierta que INKEY$ trata los dígitos como caracteres, no como números. He ahí una distinción importante. Significa que el ordenador *no* está almacenando estos dígitos particulares como números que hay que utilizar para cálculos (es decir, *no* bajo la forma de "punto flotante binario") sino más bien como simples caracteres, lo mismo que cualesquiera otros del teclado.

Como queremos que el ordenador preste atención cons-

tante a las teclas apretadas en el teclado, hemos de escribir el mandato de INKEY$ dentro de un bucle:

```
110 IF INKEY$ = "" THEN GOTO 110
```

¡La declaración GOTO, después del THEN, envía el control del programa nuevamente al comienzo de la misma línea! Cada línea, parafraseada, dice: "Si INKEY$ no devuelve ningún carácter (significando que no se está apretando ninguna tecla en el teclado) retorna al comienzo de la línea y efectúa nuevamente el INKEY$". Mientras usted no apriete una tecla, el ordenador estará realizando la línea 110 una y otra vez. Pero tan pronto como apriete una tecla, la declaración verdadero-o-falso:

```
INKEY$ = ""
```

se vuelve falsa, y el control del programa pasa a la línea siguiente.

Como usted ha pasado ya este programa, sabe que INKEY$ es completamente distinto de INPUT por la acción que determina. INPUT imprime cada carácter que usted escribe en el ángulo inferior izquierdo de la pantalla. INPUT también espera que usted apriete la tecla de ENTER antes de que almacene realmente sus datos de input en una variable. INKEY$ no hace ninguna de estas cosas; sencillamente, lee un solo carácter del teclado y devuelve este carácter al programa.

En cuanto INKEY$ ha retornado un carácter, la línea 120 almacena el carácter en la variable en serie I$:

```
120 LET I$ = INKEY$
```

Entonces, las líneas 130, 150 y 160 comprueban el valor de I$ —en tres declaraciones IF diferentes— para decidir qué debe hacer a continuación. Hemos visto ya que la línea 130

envía el control del programa hasta el comienzo, para despejar la pantalla.

Las líneas 150 y 160 ilustran una característica de la declaración IF que todavía no hemos examinado: el empleo de las palabras AND (Y) y OR (O) en declaraciones de verdadero-o-falso. Estas dos palabras le permiten a usted comprobar más de una igualdad o desigualdad en una sola declaración IF.

La línea 150, que comprueba las teclas P o M, contiene la palabra OR:

```
150 IF I$ = "P" OR I$ = "M" THEN GOSUB 300
```

En otras palabras, si la variable I$ contiene el carácter P *o* el carácter M, entonces la línea 150 llama a la subrutina de la línea 300. Una declaración conteniendo la palabra OR es verdadera si *uno* de los elementos es cierto, o lo son *ambos*.

La línea 160 es la más difícil del programa. Compruebe si usted está apretando una de las teclas de dirección, de la "1" a la "8":

```
160 IF I$ > = "1" AND I$ < = "8" THEN
                GOSUB [100 * (VAL (I$) 4)]
```

Una declaración conteniendo la palabra AND es cierta si *ambos* elementos lo son. Así, esta línea llama a una subrutina solamente si el valor I$ está entre el carácter "1" y el carácter "8". La declaración GOSUB de la línea 160 es diferente de otras declaraciones GOSUB en cuanto que *calcula* el número de línea de la subrutina que convoca. Veremos que esto da a la línea 160 la facultad de convocar una de ocho subrutinas diferentes, característica importantísima, por cierto. La clave de esta característica radica en la expresión que sigue a la palabra GOSUB:

```
[100* (VAL (I$) + 4)]
```

Hay algunas cosas acerca de esta expresión que usted comprenderá mejor después de haber leído el Capítulo 4. (En realidad, la línea 160 debería darle mucho que pensar. No se apure demasiado si no lo entiende inmediatamente). En pocas palabras esta expresión produce un múltiplo de 100 entre 500 y 1.200; y usted recordará que éstos son los números de línea de las subrutinas de dirección. Los números de línea están calculados en tres pasos:

1. La función VAL convierte el valor de I$ de un carácter (entre "1" y "8") en un número (entre 1 y 8).
2. La expresión añade 4 a este número, produciendo un número comprendido entre 5 y 12.
3. El resultado del paso 2 se multiplica entonces por 100, dando un número situado entre 500 y 1.200.

Lo que importa comprender es que se puede escribir una sola declaración GOSUB que llamará a una de varias subrutinas diferentes. La declaración puede "decidir" qué subrutina convocar, basándose en un cálculo que incluye una variable.

Para resumir, las líneas 130, 150 y 160 pueden originar la convocación de una subrutina, pero solamente si usted está apretando una de las teclas que tienen significado para este programa: C, M, P, o de 1 a 8. Si aprieta alguna otra tecla, ninguna declaración IF producirá la llamada de una subrutina, y el programa pasará, sencillamente, a la línea 170:

```
170 GOTO 110
```

Así pues, de una manera o de otra, el programa siempre retorna a la declaración INKEY$ de la línea 110.

Ahora veamos brevemente las subrutinas, empezando con la de dirección. Todas estas subrutinas cortas tienen el mismo formato. Cada una contiene dos declaraciones LET que asignan valores a las variables HM y VM. En la subrutina que mueve el pixel, los valores de estas dos variables se

usan para ajustar la localización actual en la dirección correcta. Para ver cómo funciona esto, miremos la subrutina de dirección "NW" de la línea 500. Esta subrutina coloca el movimiento horizontal a −1 y el vertical a 1:

```
500 LET HM = -1
510 LET VM = 1
```

Puede usted ver que substrayendo 1 de la parte horizontal de una dirección de pixel, moveremos este pixel hacia la izquierda (es decir, hacia el "oeste"). Del mismo modo, añadiendo 1 a la parte vertical de la dirección, moveremos el pixel para arriba (o sea, hacia el "norte").

Cada una de las subrutinas de dirección termina imprimiendo la dirección nueva en el ángulo superior izquierdo de la pantalla (en la dirección 0,0):

```
520 PRINT AT Z,Z; "NW"
```

En las líneas 350 y 360 la subrutina de movimiento del pixel, la dirección actual (CX,CY) se cambia por los valores de HM y VM:

```
350 LET CX = CX + HM
360 LET CY = CY + VM
```

La línea 350 se puede parafrasear como "añade el valor de HM al valor de CX, y almacena el resultado en la variable CX". El valor antiguo de CX se ha perdido, lo mismo que el de CY de la línea 360.

Una vez calculada la dirección nueva, se puede colocar el pixel en la pantalla:

```
370 PLOT CX,CY
```

La subrutina de la línea 300 tiene dos características más que debería estudiar usted. Las líneas que van de la 300 a la

330 comprueban la dirección actual para ver si el pixel ha llegado a uno de los cuatro bordes de la pantalla. Si ha llegado, es preciso cambiar el valor HM, o el valor VM a fin de introducir nuevamente el pixel en el área de la pantalla. Por ejemplo, si la dirección horizontal llega a 63 (el lado derecho de la pantalla) es preciso cambiar HM, la variable del movimiento horizontal a —1 con objeto de que el pixel avance hacia la izquierda:

```
300 IF CX = 63 THEN LET HM = —U
```

Finalmente, en la línea 340, la subrutina de movimiento del pixel comprueba y ve si debería borrar el pixel actual antes de situar el nuevo. Si usted aprieta la tecla [M], UNPLOT borra el pixel de CX,CY:

```
340 IF I$ = "M" THEN UNPLOT CX, CY
```

Si usted apretase la tecla [P], el mandato UNPLOT no tendría lugar, y el pixel actual continuaría en la pantalla junto con el siguiente.

Estudiando cuidadosamente este programa, puede usted perfeccionar su comprensión de varias declaraciones BASIC, incluidas la LET, la IF y la GOSUB. Por añadidura, este programa le presenta varias palabras nuevas del vocabulario BASIC: REM, INKEY$, TAB, AND, OR y VAL. Si todo esto le resulta excesivo para asimilarlo en una sola lectura, debe retornar a este programa más tarde, quizás después de haber leído los capítulos finales de este libro.

RESUMEN

En este capítulo hemos recorrido mucho terreno. La lista siguiente le ayudará a recordar cada uno de los mandatos BASIC que aprendió y a revisar qué efecto producen:

- Definir variables y darles valores: LET, INPUT.

Leer información procedente del teclado: INPUT, INKEY$.

- Exhibir información en la pantalla: PRINT, AT, TAB, CLS.

Visualizar (exhibir) gráficos en la pantalla: PLOT, UNPLOT.

Repetir instrucciones, crear bucles: FOR, TO, STEP, NEXT, GOTO.

Controlar el orden de la actuación: GOTO, GOSUB, RETURN.

Tomar decisiones: IF, AND, OR, THEN.

Documentar su programa: REM.

CAPITULO 4
TOME CINCO:
LOS NUMEROS EN SU ORDENADOR

INTRODUCCION

En este capítulo, usted aprenderá a utilizar el ordenador para efectuar cálculos numéricos. El capítulo empieza con una discusión de las operaciones aritméticas, seguida de una breve inspección de algunas de las funciones aritméticas más corrientes. Verá usted algunos ejemplos de todas estas características en acción.

También encontrará programas importantes y útiles presentados en este capítulo. El primer programa convierte el ordenador de usted en una "supercalculadora", completa, con operaciones, funciones y memoria. El segundo programa, el más largo de este libro, le permitirá crear gráficos de barras para cualquier clase de datos numéricos. Aprenderá a utilizar ambos programas, y tendrá ocasión de ampliar sus conocimientos sobre la programación BASIC estudiando la estructura y la lógica de los programas. En particular, aprenderá algo sobre *matrices,* en el segundo programa. Una matriz u ordenación (en inglés *array*) es una estructura esencial de la programación en BASIC que permite almacenar muchos artículos bajo un solo nombre de variable.

CALCULOS EN EL ORDENADOR

Las operaciones aritméticas que su ordenador puede realizar son: adición, sustracción, multiplicación, división y exponenciación. Los símbolos usados para estas operaciones son:

+ adición
— sustracción
* multiplicación
/ división
** exponenciación

Una manera de instruir al ordenador para que efectúe un cálculo es una declaración de PRINT. El ordenador empezará evaluando el cálculo, y luego visualizará los resultados en la pantalla. Para ver cómo funciona esto, introduzca, como inmediatos, cada uno de los mandatos siguientes:

```
PRINT 15, 193 + 6.5
PRINT 1235—872
PRINT 18 * 97
PRINT 183/5
PRINT 9 ** 3
```

Cada vez que introduzca uno de estos mandatos, verá el resultado del cálculo casi inmediatamente, visualizado en la cima de la pantalla. El último de estos cinco mandatos PRINT es un ejemplo de exponenciación. La expresión "9**3" significa 9 elevado a la tercera potencia, o sea 9 al cubo. El resultado, como puede usted ver, es 9x9x9, o 729. (Símbolo "**" es un carácter de desplazamiento localizado en la tecla [H]).

También pueden incluirse múltiples operaciones aritméticas en una sola declaración. Por ejemplo, la declaración siguiente origina la suma de dos productos:

```
PRINT 9 * 8 + 5 * 3
```

El número exhibido en la pantalla es 87, suma de 72 y 15. Advierta que el ordenador ha calculado primero las dos multiplicaciones, y luego ha sumado los dos productos. He ahí un punto esencial. Cuando el ordenador calcula varias operaciones lo hace en un orden fijo. Y uno ha de tener en

cuenta este orden de operaciones, si no quiere correr el riesgo de conseguir unos resultados falsos.

El orden de las operaciones es como sigue:

1. Exponenciación, de izquierda a derecha.
2. Multiplicación y división, de izquierda a derecha.
3. Adición y sustracción, de izquierda a derecha.

Para ensayar este orden fijo, introduzca la línea siguiente:

```
PRINT 1 + 2 ** 3 * 4—5
```

¿Habría predicho usted el resultado acertado, que es 28? He aquí cómo se evalúa la declaración aritmética: Primero la exponenciación, 2**3, que da 8; luego el producto, 8*4, que da 32, finalmente, la adición de 1 y la sustracción de 5, dando un resultado final de 28.

Evidentemente, si usted hubiese introducido esta línea esperando que el ordenador efectuase las operaciones de izquierda a derecha —sin tener en cuenta para nada la naturaleza de las mismas— habría quedado muy sorprendido. Realizando cada operación en una secuencia de izquierda a derecha, el resultado sería 103. (Pruébelo: 1 más 2 es 3; 3 al cubo es 27; 27 multiplicado por 4 es 108; 108 menos 5 es 103). Es evidente, uno tiene que pensar cuidadosamente en el orden de operaciones del ordenador siempre que formula un cálculo aritmético.

Sin embargo, a menudo querrá hacer cálculos que no se adapten adecuadamente a este orden fijo. Por fortuna, el BASIC le da una manera de superar el orden normal y establecer el que le convenga a usted. Para ello, debe emplear paréntesis en las expresiones aritméticas.

Cuando parte de una expresión está encerrada entre paréntesis, aquella parte se calcula primero. Por ejemplo, entre estas dos declaraciones, por turno:

```
PRINT 1 + 2 * 3
PRINT (1 + 2) * 3
```

En la primera declaración, la multiplicación se realiza primero, y luego la adición, dando un resultado de 7. En la segunda declaración, el ordenador calcula primero la suma encerrada en el paréntesis, y luego la multiplica por 3, y el resultado es 9.

También se pueden escribir declaraciones con niveles de paréntesis. (A los paréntesis encerrados dentro de otro paréntesis los ingleses los llaman, a veces, *nested,* que significaría *anidados.* La traductora del presente libro propondría, para el castellano, la denominación de paréntesis *interiores,* la cual permitiría distinguir todos los niveles sensibles, porque permitiría la denominación de *interior segundo, interior tercero,* etc.)(1) El ordenador se ocupa primero del paréntesis más *interior,* y luego sigue hasta el más exterior. Por lo tanto, la declaración siguiente, con paréntesis, se calcularía de izquierda a derecha:

```
PRINT (((1 + 2) ** 3) * 4) —5
```

El resultado de esta declaración, según vimos antes, será 103. Fíjese en un detalle importante sobre los paréntesis interiores. A cada símbolo de apertura de paréntesis "(" le corresponde su pareja de cierre de paréntesis ")". Si por descuido escribe usted una expresión en que los símbolos de paréntesis no vayan por parejas —uno de cierre para cada uno de apertura— el ordenador se negará a realizar el cálculo.

Hemos visto anteriormente que las expresiones aritméticas pueden contener variables; una expresión será válida siempre que las variables que contenga estén definidas ya. Esto vale para las declaraciones LET, para las PRINT, y para la mayoría de las declaraciones del BASIC en las que se acude a valores numéricos. A fin de que vea por sí mismo que esto es cierto, introduzca la siguiente serie de instrucciones como mandatos inmediatos:

(1) Nota de la traductora

```
LET A = 1
LET B = A + 2
LET C = B ** 3
LET D = C * 4
LET E D—5
PRINT A, B, C, D, E
```

Cada declaración LET de esta secuencia utiliza el valor de la variable definida en la declaración anterior. El mandato PRINT visualiza luego los valores de todas las variables. Observe que la variable final, E, tiene el valor de 103, resultado de la expresión (((1 + 2)**3) *4) —5. Las otras variables contienen los valores de los cálculos intermedios.

La declaración PRINT anterior ilustra un rasgo adicional del BASIC del cual no habíamos hablado todavía: el uso de *comas* para separar elementos de visualización. Una coma le dice al ordenador que deje, entre un elemento y otro, una anchura equivalente a la mitad de la de la pantalla. De modo que el resultado de esta declaración PRINT es la visualización siguiente:

```
1               3
27              108
103
```

En una declaración PRINT, la diferencia entre un punto y coma y una coma es muy importante. Como recordará usted, el punto y coma manda al ordenador que sitúe los elementos uno después de otro.

Aún debería usted fijarse en otro detalle referente a los cálculos: Para el ordenador, la división por cero no está definida; al escribir una instrucción que originase una división por cero suscitaría un mensaje de error. Sabiendo esto, usted no escribiría, adrede, una declaración tal como:

```
LET E = 1/0
```

Pero es posible que a menudo surjan problemas en los

que usted escriba declaraciones de división que contengan variables en el denominador; por ejemplo:

```
LET E = 1/D
```

Si en algún caso esta variable D equivaliese a cero, al ejecutar esta declaración LET, el ordenador le daría a usted un mensaje de error. Esto puede constituir un problema singularmente difícil en un programa en el que usted no sepa con certeza, por adelantado, qué valores tomará D. Una manera de soslayar el problema consiste en escribir la declaración LET como una cláusula de una declaración IFG:

```
IF D<>0 THEN LET E = 1/D
```

Esta declaración asegura que la instrucción LET sólo será ejecutada si la variable D no contiene un valor de cero.

Ha visto usted que el teclado de su ordenador le ofrece varias funciones matemáticas útiles. Puede escribirlas conmutando el teclado a la modalidad de función. Exploremos el empleo de estas funciones en cálculos.

FUNCIONES ARITMETICAS

Veremos las funciones siguientes:

- ABS (para valor absoluto; situada debajo de la tecla [G]);
- INT (para valor entero (integral); situada debajo de la tecla [R]);
- SGN (para signo; situada debajo de la tecla [F]);
- SQR (para raíz cuadrada; situada debajo de la tecla [H]).

Se trata de unas funciones que muy probablemente emplee usted algún día, aunque no tenga aficiones matemáticas. El ordenador también le ofrece series de funciones, para

usos especiales. Entre ellos están las funciones trigonométricas (SIN, COS, TAN, ARCSIN, ARCCOS, ARCTAN) y las funciones exponenciales (EXP, LN). Si estas funciones le interesan, puede documentarse sobre ellas leyendo el Apéndice A.

La función del valor absoluto, ABS, suministra el valor *positivo* de cualquier número. En otras palabras, si un número es negativo, ABS lo hace positivo. Si un número es positivo, ABS no lo cambia. Introduzca los dos mandatos siguientes para ver cómo funciona ABS:

```
PRINT ABS —45
PRINT ABS 45
```

El resultado de ambos mandatos es 45. Advierta que el signo menos de la primera declaración expresa sencillamente el signo del número 45, y *no* una operación entre dos números diferentes.

La función INT elimina la parte fraccionaria (si la hay) de un número decimal. Por ejemplo, ensaye estos mandatos:

```
PRINT INT 19.34
PRINT INT 19.81
```

El resultado de ambas declaraciones es 19. Fíjese en que INT no redondea el valor; simplemente *trunca* el número, dejando solamente la parte entera. Más adelante de este mismo capítulo veremos la manera de utilizar INT para crear una función redondeadora.

La función SGN siempre produce uno de tres valores posibles, dependiendo del signo del número que se escriba después de ella:

—1 si el número es negativo
0 si el número es cero
1 si el número es positivo.

La declaración:

```
PRINT SGN —45, SGN O, SGN 45
```

produce la secuencia —1 0 y 1.

Finalmente SQR da la raíz cuadrada de un número positivo. Por ejemplo:

```
PRINT SQR 81
```

produce 9. Si usted escribe un número negativo después de SQR, el ordenador le dará un mensaje de error; la raíz cuadrada de un número negativo no está definida.

Las funciones pueden formar parte de declaraciones que incluyan más de una operación aritmética; pero, repitámoslo, deberá usted prestar mucha atención al orden de las operaciones. Las funciones se evalúan antes que ninguna de las cinco operaciones aritméticas (a menos que dichas operaciones estén encerradas dentro de paréntesis). Por ejemplo, entre la declaración:

```
PRINT 10 * SQR 81 + 19
```

El resultado es 109. El ordenador empieza buscando la raíz cuadrada de 81, que es 9; luego multiplica este 9 por 10, dando 90, y finalmente añade 19, dando un resultado final de 109. Compare esto con la declaración:

```
PRINT 10 * SQR (81 + 19)
```

Esta declaración da 100. El ordenador empieza realizando la adición encerrada entre paréntesis, luego saca la raíz cuadrada, y luego ejecuta la multiplicación.

Finalmente, unas funciones pueden trabajar sobre los resultados de otras funciones. Dicho de otro modo, uno puede utilizar funciones en expresiones complicadas tales como:

SQR INT 9.73

Esta expresión dice: "busca la raíz cuadrada del valor entero de 9.73". Si usted entra esta expresión en su ordenador como parte de una declaración PRINT, verá que el resultado es 3.

En la sección siguiente, aprenderemos algo más sobre el empleo de la declaración INPUT para cálculos, y pasaremos un programa corto que nos ayudará a experimentar con esta característica.

CALCULOS CON LA DECLARACION INPUT

Escriba NEW para limpiar cualesquiera líneas de la memoria de su ordenador; luego introduzca las tres líneas siguientes de programa:

```
10 INPUT V
20 PRINT V
30 GOTO 10
```

Superficialmente esto no parece un programa útil. La línea 10 lee un valor numérico procedente del teclado, y almacena el valor en la variable V. La línea 20 exhibe el valor de V en la pantalla. La línea 30 devuelve el control del programa a la línea 10, formando un bucle infinito. Cuando usted pase el programa, el ordenador aguardará que entre un número mediante el teclado, y luego visualizará este número en la pantalla del televisor.

Como veremos, el ordenador sabe leer realmente expresiones aritméticas procedentes del teclado durante el pase del programa. He ahí una característica singular y muy útil del BASIC del T/S 1000. En vez de limitarse a escribir un número en el teclado, se puede escribir cualquier expresión aritmética válida, incluyendo cualesquiera de las cinco operaciones aritméticas, y cualquier función numérica. El ordenador realizará inmediatamente los cálculos especificados en

la expresión y luego almacenará el *resultado* de dichos cálculos en la variable de INPUT, que en este caso será la V.

Pase el programa y ensáyelo ahora. Empiece introduciendo un número sencillo, digamos el 10; el número aparecerá en la cima de la pantalla. Y podrá ver que el programa funciona.

Luego introduzca una expresión. Ensaye ésta:

```
10 * SQR 49 + 16/8
```

El número 72 aparecerá bien visible en la cima de la pantalla. El ordenador ha evaluado la expresión, almacenando el resultado, 72, en la variable V.

Ahora introduzca la expresión:

```
V/9
```

El ordenador responderá visualizando el número 8 en la pantalla. La expresión V/9 significaba: "divide el valor actual de la variable V por 9". Dado que V contenía el valor 72, procedente de los cálculos anteriores, el resultado de la división es 8.

Resumiendo, pues, hemos visto que el ordenador aceptará cualquier expresión numérica válida en respuesta a una declaración numérica de INPUT. Tal expresión hasta puede incluir variables, siempre que tales variables hayan sido definidas ya.

Ahora estamos preparados para contemplar el programa que convierte al ordenador de usted en una supercalculadora.

CONSTRUCCION DE UNA SUPERCALCULADORA

El programa de supercalculadora aparece en la Figura 4.1. Escríbalo cuidadosamente en su ordenador y luego pásselo. Verá el mensaje:

en vídeo inverso, y la pista L, entre comillas, indicando que el ordenador está esperando un input en cadena.

Usted puede utilizar este programa para encontrar la respuesta a cualquier problema de cálculo. La expresión que escriba en el ordenador puede incluir:

- las cinco operaciones aritméticas;
- cualesquiera de las funciones matemáticas del teclado;
- expresiones complejas, con paréntesis interiores y funciones también interiores;
- dos variables: M1 y M2, representando la función de memoria del programa. (Hablaremos de esta particularidad muy en breve).

En otras palabras, usted puede introducir cualquier cálculo que el ordenador haya de aceptar como expresión numérica válida. El programa tiene la forma de bucle infinito; y usted puede utilizarlo para efectuar tantos cálculos como quiera.

Probémoslo. Escriba la expresión siguiente en el teclado:

```
1 + INT (SQR (2 * 3) * 4)
```

Apriete la tecla de ENTER y espere la acción. La pantalla se queda en blanco por un período muy breve. Luego, cuando retorna la información, la pantalla debería tener el aspecto de la Figura 4.2. Estúdiela cuidadosamente para descubrir todo lo que tenga que decirle.

Primero, aparece en el centro de la pantalla el "eco" de la expresión que usted introdujo en el ordenador. Esto tiene gran importancia... ¿Cuántas veces ha realizado usted una serie de cálculos con una calculadora de bolsillo, y luego, al ver el resultado, se ha preguntado nerviosamente si había apretado las teclas adecuadas? Las equivocaciones en el input son, probablemente, la causa más corriente de resultados incorrectos cuando usamos máquinas para hacer nues-

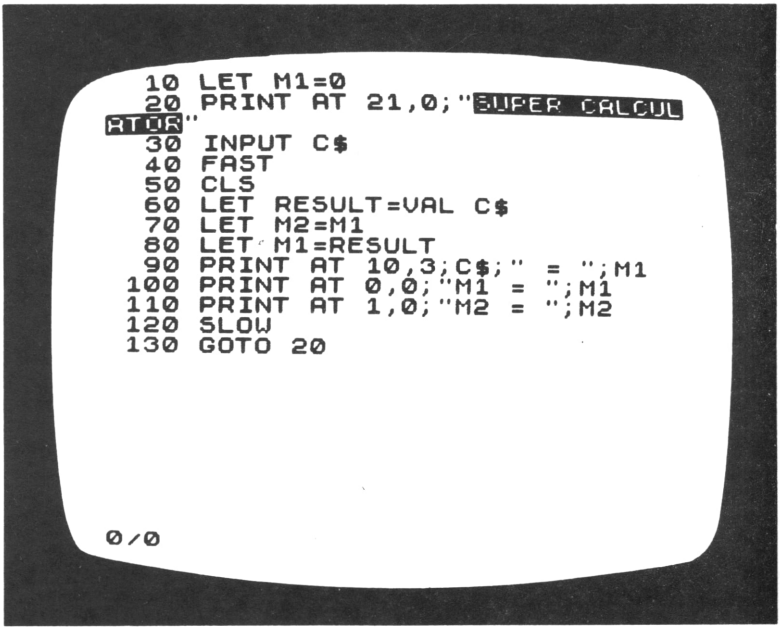

Figura 4.1: El programa de supercalculadora

tros cálculos. De manera que lo primero que le dice este programa, incluso antes de contestar su problema, es la ex presión exacta que usted ha introducido en el ordenador. Usted puede estudiar este "eco" a su placer para asegurarse de que realmente representa el cálculo que quería efectuar. Después del eco, el ordenador le da la respuesta al problema. En este caso, la respuesta es 10.

Además, este programa establece para usted dos variables de memoria. Las llamamos M1 y M2; después del primer cálculo, sus valores del momento se le visualizan en ángulo superior izquierdo de la pantalla. M1 es el resultado de los cálculos más recientes que el ordenador ha realizado. M2 es el resultado del cálculo anterior, o sea, el antepenúltimo. Como puede ver en la pantalla, los valores actuales de M1 y M2 son:

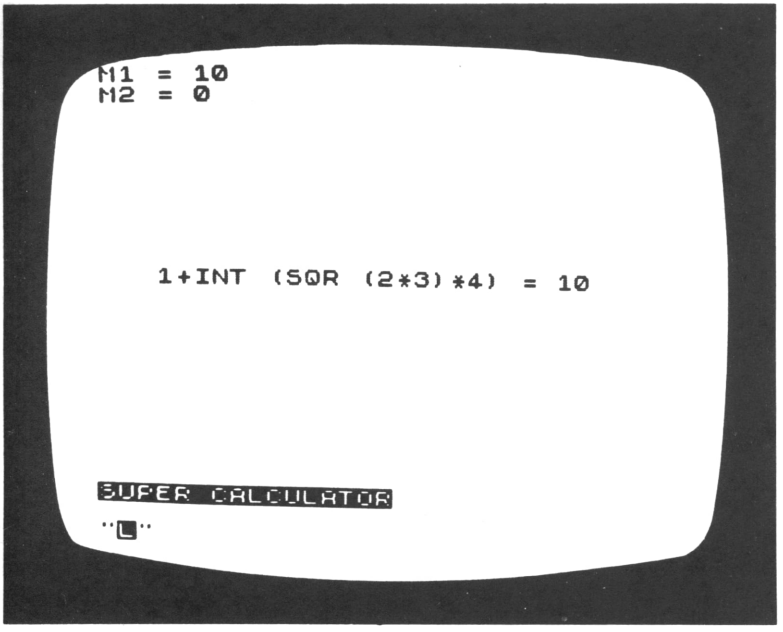

Figura 4.2: Una respuesta del programa supercalculador

M1 = 10
M2 = 0

El valor de M1 es el resultado del cálculo visualizado actualmente en el centro de la pantalla. Dado que éste es el primer cálculo que usted hizo durante este pase del programa, M2 sigue siendo igual a cero. (Tanto M1 como M2 tienen el valor de cero al empezar el programa).

Ahora escriba la segunda expresión para que el programa la evalúe:

(18 * * 2)/5

La respuesta, tal como se ve en la Figura 4.3., es 64.8. Vea qué les ha pasado a las dos variables de la memoria:

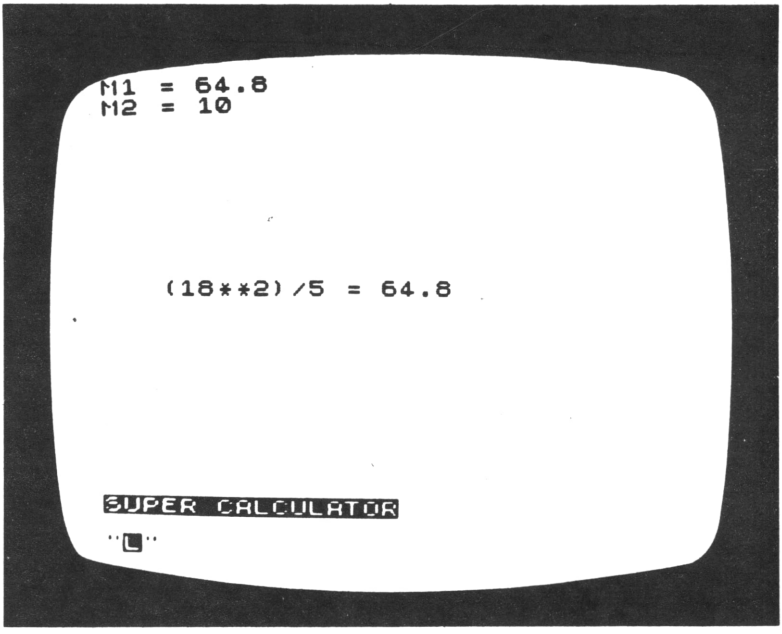

Figura 4.3: Un segundo cálculo

M1 64.8
M2 10

Ahora el resultado más reciente se presenta como M1. El resultado del primer cálculo, 10, se presenta como M2. El propósito de estas dos variables de memoria es éste: uno puede usarlas en expresiones que introduzca en el ordenador. Por ejemplo, entre la siguiente expresión como tercer cálculo:

M1/M2

Esto significa: "divide el valor actual de M1 por el valor actual de M2". El ordenador le dará la respuesta:

M1/M2 = 6.48

En muchas ocasiones, el tener acceso a estas dos variables de la memoria le será muy útil, cuando tenga que efectuar una larga serie de cálculos. Sus valores actuales serán almacenados y visualizados en cada paso.

Siga practicando con este programa hasta que comprenda todas sus particularidades. Es probable que descubra muy pronto que el pase del programa se interrumpe, y aparece el mensaje de error, si usted escribe una expresión que el ordenador no puede evaluar en un número. Esto puede suceder de una variedad de maneras; cualquiera de los errores siguientes puede originar la terminación del programa:

- introducir una expresión que contenga un número impar de signos de paréntesis;
- incluir una variable indefinida en su expresión (es decir, cualquier variable que no sea M1 ó M2).
- solicitar un cálculo que resulte en una división por cero;
- utilizar una función que no origine un número.

Si usted incurre en una de estas equivocaciones, puede reanudar, naturalmente el programa introduciendo, sencillamente, el mandato de RUN. Sin embargo, la desventaja de hacerlo así es que perderá los valores actuales de las dos variables de memoria, M1 y M2. (Según recordará, el mandato de RUN, elimina todas las variables del momento antes de empezar el programa). Una manera mejor de reanudar el programa consiste, pues, en introducir el mandato inmediato:

GOTO 100

Cuando usted haga esto, los valores actuales, inalterados, de M1 y M2 aparecerán en la pantalla, y el programa estará listo para el cálculo siguiente. Veremos exactamente por qué actúa este mandato GOTO cuando examinemos la estructu-

ra interna del programa. En general, siempre se puede dar nuevo comienzo a un programa utilizando una declaración GOTO en lugar de RUN. El resultado es que los valores actuales de cualesquiera variables en la memoria del ordenador se conservarán.

Ahora echemos una mirada al programa en sí. Uno puede visualizar el programa en la pantalla introduciendo un mandato de STOP para terminar el pase del programa, y luego apretar nuevamente la tecla de ENTER. O puede referirse de nuevo a la Figura 4.1., donde aparece el listado del programa.

DENTRO DEL PROGRAMA SUPERCALCULADOR

Las tres primeras líneas del programa son directas. La línea 10 inicia la variable de memoria M1 a cero: la línea 20 exhibe el título del programa en la pantalla, y la línea 30 lee el input procedente del teclado. Adviertan que el input está almacenado en la variable en cadena C$:

```
30 INPUT C$
```

La expresión todavía *no* está evaluada como un cálculo, como lo estaría si la variable fuese numérica. Por lo que el ordenador sabe en este punto, C$ contiene una serie ordinaria de caracteres.

La línea 40 pone al ordenador en el modo rápido (fast):

```
40 FAST
```

Hemos visto ya el mandato de FAST en acción un par de veces en programas anteriores de este libro, pero nunca hemos examinado exactamente qué hace. En el modo rápido, el ordenador deja de enviar información a la pantalla del televisor durante el rato en que está ejecutando cálculos. Como resultado, tales cálculos se llevan a cabo mucho más aprisa. Cuando pase usted un programa, podrá decir

que está en modo rápido si en alguno de sus puntos la pantalla queda en blanco. En el programa supercalculador, recordará, la pantalla se queda en blanco justo después de haber introducido usted las expresiones que quiere calcular. El ordenador está dedicando sus recursos a calcular, y, en consecuencia, la tarea se hace rápidamente.

Como un experimento, usted puede probar de borrar la línea 40 y luego pasar el programa para ver qué diferencia resulta. Encontrará que los cálculos tardan mucho más en quedar terminados. (Asegúrese de reponer la línea 40 en el programa antes de continuar).

La línea 50 limpia la pantalla con el mandato CLS. La línea 60 es la verdadera declaración de trabajo de este programa:

```
60 LET RESULT = VAL C$
```

Como hemos visto en el Capítulo 3, la función VAL busca el equivalente numérico de una cadena. Por supuesto, para que VAL funcione siquiera, la cadena debe estar compuesta de caracteres que se puedan convertir en números; por este motivo, las expresiones que usted escriba en este programa han de ser, todas, expresiones numéricas válidas. Cuando uno se equivoca y escribe una expresión que no vale, el mensaje de error que contiene es:

```
C/60
```

El 60 significa que el programa termina en la línea 60. Como puede leer usted en el Apéndice B, el código de error C significa que usted ha dado una expresión no válida a la función VAL.

Si todo va bien, pues, la declaración LET de la línea 60 almacena el resultado calculado de la expresión numérica en la variable numérica RESULTADO. Las líneas 70 y 80 resitúan los valores de las variables de memoria M1 y M2.

Primero, la M2 recibe el resultado del cálculo anterior, que entonces está almacenado en M1:

```
70 LET M2 = M1
```

Luego M1 recibe el resultado del último cálculo:

```
80 LET M1 = RESULTADO
```

Las tres líneas siguientes preparan la pantalla. La línea 90 exhibe la expresión original (todavía almacenada en la variable en serie C$) y la respuesta (almacenada en M1):

```
90 PRINT AT 10,3;C$;" = ";M1
```

Las líneas 100 y 110 exhiben los valores de memoria en el ángulo superior izquierdo de la pantalla:

```
100 PRINT AT 0,0; "M1 = ";M1
110 PRINT AT 1,0;"M2 = ";M2
```

Por esta razón puede usted utilizar el mandato:

```
GOTO 100
```

para reanudar el programa en el caso de una terminación de error.

Finalmente, la línea 120 retorna el ordenador al modo SLOW (lento) y la línea 130 envía el control del programa nuevamente a la línea 20, formando el bucle infinito.

Antes de llegar al final de este capítulo, pase otra vez el programa supercalculador para el último experimento. Escriba la expresión:

```
5 * * 20
```

en el teclado y apriete la tecla de ENTER. Es probable que

usted haya adivinado por adelantado que la expresión "5 elevado a la vigésima potencia" (o "5 elevado al exponente 20") daría un número muy grande, pero la visualización en la pantalla del televisor acaso le sorprenda de veras. He aquí tal como se verá:

5 * * 20 = 9,5367432E + 13

La respuesta está expresada en lo que llamamos *notación científica.* El ordenador pasa automáticamente a esta notación para números muy largos, como éste. La notación no es tan difícil de leer como quizás le parezca a usted al principio, si no está familiarizado con ella. La letra E situada hacia el final de la respuesta es la inicial de "exponente" y significa precisamente "exponente"; así pues, la expresión E 13 significa 10 elevado al exponente 13, o sea:

10.000.000.000.000.

De modo que la expresión completa dada en notación científica significa:

9,5367432 x 10.000.000.000.000

ó

95.367.432.000.000

Si usted se para a pensar que el ordenador puede producir números mucho mayores que éste, quizás empiece a comprender el motivo de utilizar la notación científica. Es simplemente una manera más económica de visualizar números grandes en la pantalla.

Si le interesa profundizar más en esta cuestión, puede emplear el programa supercalculador para contestar cierto número de cuestiones que se le pueden ocurrir. Con un

poco de ensayo, puede descubrir muchas cosas sobre cómo maneja los números su ordenador:

- A medida que los números crecen más y más, ¿cuándo empieza el ordenador a mostrarlos en notación científica?
- ¿Cuál es el número mayor que puede manejar su ordenador?
- ¿Con qué exactitud visualiza el ordenador los números muy grandes? (Pruebe de escribir 5 * * 15. Usted sabe que cualquier potencia de 5 ha de terminar con la cifra 5. ¿Termina de verdad en 5 la respuesta del ordenador?)

CREAR GRAFICOS DE BARRAS

El programa siguiente que examinaremos está ideado para producir diagramas de barras. Un gráfico de barras proporciona una manera conveniente y a veces realmente espectacular de comparar series de números. Cada número está representado por una barra; cuanto mayor el número, más larga la barra. Cuando usted contempla un gráfico de barras, puede ver a la primera mirada la importancia relativa de cada categoría representada en el gráfico. Por esta razón, quizás quiera utilizar el programa de gráficos para representar información importante siempre que le interesen más los valores *relativos* que los verdaderos datos.

Dado que el diagrama de barras se ocupa de listas de datos, ha llegado el momento de que se entere bien de la estructura de datos en BASIC diseñada para manejar tales listas. A esta estructura la llaman *ordenación* (o serie, o conjunto, y en inglés *array*). Veremos ordenaciones, y la manera de definirlas, antes de seguir adelante, hacia nuestro programa.

Ordenaciones

Actualmente usted ya sabe que una simple variable puede contener, exactamente, un solo valor cada vez. Si usted asigna un valor nuevo a una variable —empleando LET o INPUT— el valor antiguo se perderá para siempre. La simple variable es una manera conveniente de almacenar cualquier número o cualquier clase de datos que no estén relacionados entre sí de ninguna manera especial excepto en cuanto que todos son necesarios para el programa.

A menudo, sin embargo, cuando usted use el ordenador para procesar datos, tendrá listas o tablas de datos emparentados que le gustaría almacenar en la memoria del ordenador. Pueden intervenir docenas —y hasta centenares— de datos diversos, de manera que resultaría muy incómodo el definir una variable separada para cada clase.

El BASIC proporciona estructuras especiales de datos, llamadas *ordenaciones (arrays)* que son un elemento perfecto para almacenar grandes cantidades de datos. Se dice que las ordenaciones tienen *dimensiones*. Una ordenación unidimensional se emplea para almacenar una *lista de datos*. Una ordenación bidimensional puede almacenar una *tabla* de datos, dispuestos en filas y columnas. En su T/S 1000 es probable que usted sólo muy raras veces necesite emplear ordenaciones que tengan más de dos dimensiones, aunque el BASIC permite ordenaciones multidimensionales. El programa de gráficos de barras utiliza ordenaciones unidimensionales, de manera que limitaremos nuestro examen al almacenamiento de *listas* de datos.

Suponga que tiene usted una lista de 15 números que quiere introducir en la memoria del ordenador. Usted necesitará diseñar un programa que ejecute tres tareas:

1. leer los números procedentes del teclado durante el pase del programa;
2. realizar algunos cálculos con los números;

3. visualizar nuevamente los números en la pantalla, junto con los resultados de los cálculos.

Usted puede empezar definiendo una ordenación que contenga a la vez los 15 números. Digamos que a esta ordenación decide llamarla N (por *números*). He aquí cómo define la ordenación en la primera línea de su programa:

```
10 DIM N (15)
```

La palabra clave DIM, situada sobre la tecla [D] significa dimensión. Una declaración DIM especifica las cuatro características esenciales de una ordenación:

1. el nombre de la ordenación
2. el tipo de datos que contendrá (numéricos o en serie)
3. el número de dimensiones que tiene
4. el número de datos singulares permitidos en cada dimensión. (A esta última característica se la denomina *longitud*).

Aquí están las características de la ordenación definida antes, en la línea 10:

1. su nombre es N;
2. es una ordenación numérica (puesto que su nombre no termina en el carácter $);
3. tiene una sola dimensión (puesto que hay un solo número especificado dentro del paréntesis);
4. la longitud de la ordenación es 15.

Cuando el ordenador llega a esta línea de su programa, reserva automáticamente espacio en su memoria para 15 números bajo el nombre de ordenación N.

Con la ordenación definida, usted necesita una manera de identificar cada uno de sus 15 elementos. En cuanto usted ha asignado números a la ordenación, acaso necesite

acceder al séptimo número de la lista, por ejemplo. En tal caso, necesitará una manera adecuada de decirle exactamente al ordenador qué elemento le interesa.

El sistema para identificar los elementos de una ordenación es muy sencillo. Usted escribe el nombre de la ordenación, luego, entre paréntesis, el número de elementos que quiere identificar. Por ejemplo, el 7º elemento de la ordenación N se llama:

```
N(7)
```

Del mismo modo, los cinco elementos primeros son:

```
N(1)
N(2)
N(3)
N(4)
N(5)
```

Al número que está entre paréntesis se le llama *índice* dentro de la ordenación. Usted puede escribir declaraciones como:

```
LET N(7) = 162
INPUT N(5)
PRINT N(2)
```

La primera de estas declaraciones almacena el número 162 como 7º elemento de la ordenación N. La segunda declaración leerá un número del teclado, que se almacenará en el 5º elemento de N. La tercera declaración exhibe el valor actual del 2º elemento de N en la pantalla.

Usted también puede escribir un nombre de variable entre paréntesis después del de la ordenación; por ejemplo:

```
N(I)
```

Entonces la elección de elementos depende del valor del índice I. Si I es igual a 5, N(I) se refiere al 5º elemento de N.

Sabiendo esto, usted estará en condiciones de usar bucles FOR en los que la variable de control de los bucles se emplea también como el índice dentro de la ordenación. Por ejemplo:

```
10 DIM N(15)
20 FOR I = 1 TO 15
30 INPUT N(I)
40 NEXT I
```

En este programa, la variable de control, I, es al mismo tiempo el índice dentro de la ordenación N. Con estas sencillas líneas, puede instruir al ordenador para leer quince números en el teclado y almacenarlos todos bajo el nombre N de la ordenación. Ahora usted puede empezar a ver qué formidable colección de herramientas tenemos en las ordenaciones y bucles FOR.

Para ver este poder en acción, escriba las líneas de la 10 a la 40 anteriores, en el ordenador. Añada las tres líneas siguientes, que exhiben los quince valores de N en la pantalla:

```
50 FOR I = 1 TO 15
60 PRINT N(I)
70 NEXT I
```

Pase el programa. Entre quince números cualesquiera en el ordenador, y cuando haya terminado aparecerán los quince, a la vez, en la pantalla. Lo que importa recordar es que los quince números están almacenados en la memoria del ordenador, y se puede acceder a ellos y utilizarlos para cualquier finalidad que le convenga a usted. Toda operación que pueda ser ejecutada en una serie de elementos se puede incorporar en un bucle FOR/NEXT.

El ordenador también le permite a usted el definir ordenaciones en serie, pero el método es algo más complicado que en el caso de las ordenaciones numéricas. Nuevamente, contemplaremos una ordenación unidimensional. En la de-

claración DIM, después de haber definido usted la dimensión y su longitud, debe añadir otro número dentro del paréntesis. Este número representa la longitud de cada elemento de la ordenación en serie (o cadena) el número de caracteres que contiene. Todos los elementos de una ordenación en serie tienen la misma longitud predefinida. Miremos un ejemplo.

Supongamos que usted debe almacenar una lista de quince programas en la memoria de su ordenador para emplearlas en un programa. La palabra más larga contiene ocho caracteres. Usted puede definir una ordenación en cadena para esta lista de la manera siguiente:

```
10 DIM W$(15,8)
```

El nombre de la ordenación es W$; contendrá 15 series, de 8 caracteres de longitud cada una.

Usted puede realizar ensayos con esta ordenación introduciendo por el teclado el siguiente programa corto:

```
10 DIM W$(15,8)
20 FOR I = 1 TO 15
30 INPUT W$(I)
40 PRINT W$(I);
50 NEXT I
```

Advierta, ante todo, que usted puede referirse a elementos de esta ordenación en cadena de la misma manera utilizada en la ordenación numérica. La palabra 7ª de la ordenación W$ se llama:

```
W$(7)
```

y la palabra Iª es:

```
W$(I)
```

Ahora pase el programa, y observe qué hace. Primero entre una palabra que tenga ocho caracteres de longitud, exactamente:

CAMARADA

La palabra se almacenará en W$(1) y, gracias a la línea 40, se visualizará, al mismo tiempo, en la cima de la pantalla. Ahora entre dos palabras, que se almacenarán en W$(2) y W$(3), respectivamente. Escoja palabras de menos de ocho caracteres de longitud:

VIVO
PAN

La primera aparecerá en la pantalla inmediatamente después de la palabra CAMARADA. (Advierta que la línea 40 termina en un punto y coma, lo cual evita que se cambie de línea). No obstante, la palabra PAN aparece cuatro espacios más allá de la palabra VIVO. Usted verá en la pantalla la línea siguiente:

CAMARADAVIVO PAN

¿Por qué hay cuatro espacios de separación entre VIVO y PAN? Porque usted definió los elementos de la ordenación W$ como compuestos de ocho caracteres cada uno. Dado que VIVO sólo contiene cuatro caracteres, el ordenador llena W$(2) con espacios a fin de dar la longitud exacta de ocho.

Ahora pruebe de introducir una palabra que tenga *más de ocho* caracteres:

INTERESANTE

y verá aparecer tan sólo los ocho caracteres primeros, INTERESA, en la pantalla. La palabra entera no encaja en un

elemento de W$, con lo cual los tres caracteres últimos quedan, simplemente, eliminados.

Ahora que ha tenido una breve introducción a ordenaciones así numéricas como en cadena, está preparado ya para ver el programa de gráfico de barras. Este programa emplea dos ordenaciones diferentes —definidas en dos declaraciones DIM— para almacenar toda la información destinada a crear un gráfico de barras.

EL PROGRAMA DEL GRAFICO DE BARRAS

El listado del gráfico de barras aparece en las Figuras que van desde la 4.4. hasta la 4.7. Cuando esté escribiendo un programa largo como éste en el teclado del ordenador, resulta una idea excelente el guardarlo en cassette por fases. Cuando haya escrito usted alrededor de una cuarta parte del programa, guárdelo. Luego siga escribiendo hasta llegar, poco más o menos, a la mitad del programa. Cuando ejecute la segunda operación de ahorro, puede grabarla sobre la misma parte de cinta de cassette en la que grabó la primera parte, dado que esta primera parte ya no la necesitará más. Continúe guardando por estadios, de esta manera, hasta que haya escrito y guardado el programa entero. Este proceso le protege de una pérdida accidental de su trabajo. Lo más que puede perder en cualquier punto, mientras esté escribiendo, es algo que haya introducido desde la operación anterior de ahorro.

Cuando haya introducido el programa entero, compruébelo cuidadosamente para asegurarse de no haber cometido errores. Luego pase el programa.

Este programa produce un gráfico de barras para cualquier lista de números con un máximo de quince elementos de longitud. La Figura 4.8. muestra un ejemplo del resultado final del programa. Este gráfico de barras particular compara gastos en comestibles, artículo por artículo, para el mes de Diciembre. Si estos datos fuesen los auténticos de su casa de usted, dispondría de una representación gráfica

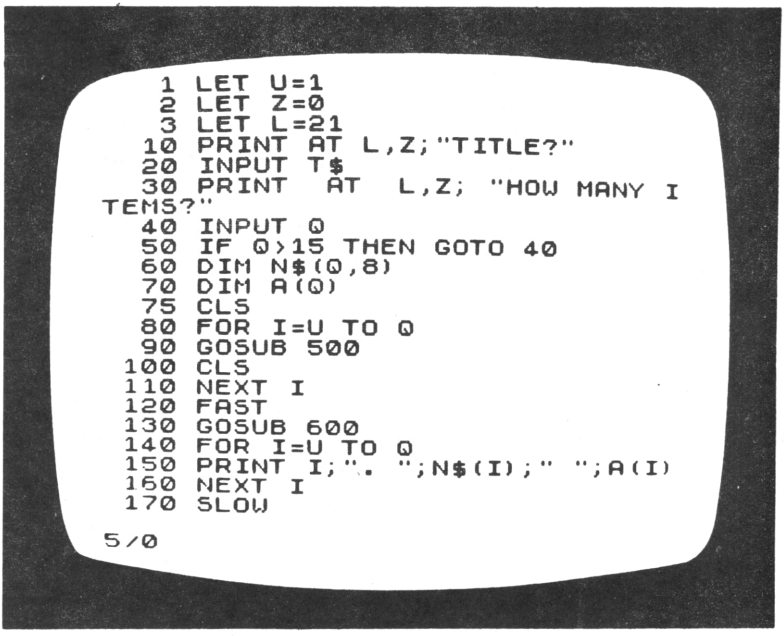

Figura 4.4: El programa de gráfico de barras

"fácil de leer" de sus gastos del mes. (Si se está rascando la cabeza y preguntándose *qué haría* usted de un diagrama de barras, deje de inquietarse. Este programa se puede utilizar para *cualquier* colección de datos, relativos a cualquier asunto. En el principio del programa, usted suministra el título del gráfico de barras. También suministra el nombre de cada categoría, junto con los datos numéricos).

Este programa le guía a usted a través de tres fases distintas de actividad a fin de producir el gráfico de barras: Primero debe introducir (input) todos los datos: incluyendo título, nombres de elementos y datos numéricos. El programa proporciona indicaciones claras para explicarle a usted exactamente qué clase de información introducir en cada punto de la fase de input. La segunda fase consiste en corregir los errores de input. Esta fase empieza con un

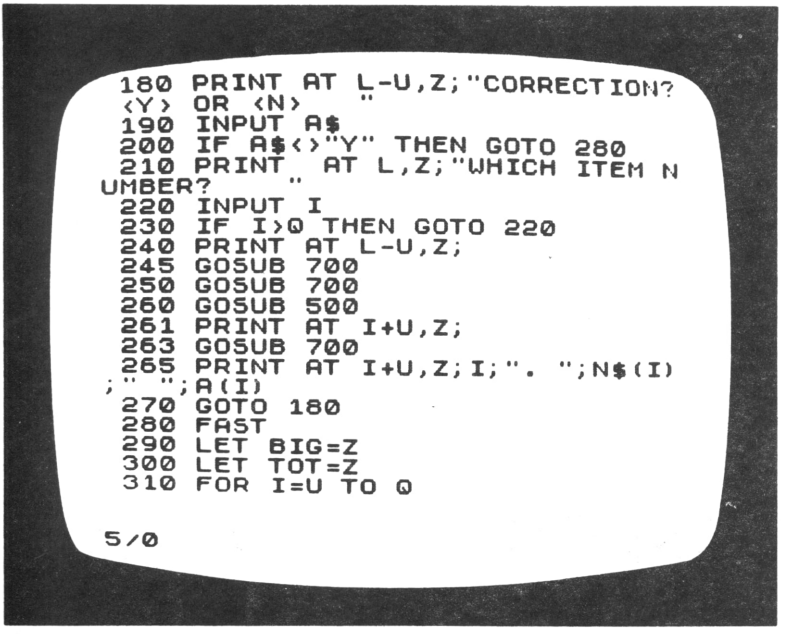

Figura 4.5: El programa de gráfico de barras (cont.)

"eco" (podríamos decir, en castellano, una "reproducción") completo del programa. en forma de tabla, de todos los datos que usted ha introducido en el ordenador. Usted puede estudiar esta información a placer, y tendrá la oportunidad de corregir todos los errores que encuentre en los datos de input. Finalmente, en la tercera fase, verá el diagrama de barras producido por los datos introducidos. Examinemos cada una de estas fases; introduciremos los gastos de comestibles en este pase de muestra a través del programa.

En el comienzo del programa, como ha visto usted ya en su propia pantalla, el programa le invita a entrar el título del gráfico (o diagrama) de barras:

¿TITULO?

Usted puede entrar cualquier título mientras su longitud no sobrepase los 32 caracteres. Entre el título que ve en la cima del gráfico de barras de la Figura 4.8. Siguiendo esta indicación, verá una segunda pregunta en la pantalla:

¿CUANTAS VECES?

Usted debe introducir el número de elementos que habrá en su programa de gráficos (hasta 15 artículos). En este gráfico de gastos en la tienda hay 15 artículos; usted debería escribir el número 15 y después apretar ENTER.

A continuación, el programa le pedirá que introduzca el nombre, y el valor numérico de cada uno de los 15 artí-

```
 320 IF A(I)>BIG THEN LET BIG=A(
I)
 330 LET TOT=TOT+A(I)
 340 NEXT I
 350 LET AVE=TOT/Q
 360 LET FAC=L/BIG
 370 CLS
 380 GOSUB 600
 390 FOR I=U TO Q
 400 PRINT N$(I);">";
 410 FOR J=U TO INT (A(I)*FAC+.5
)
 420 PRINT "*";
 430 NEXT J
 440 PRINT
 450 NEXT I
 455 PRINT
 460 PRINT "TOTAL =";TOT;" AVERAG
E =";INT (AVE*100+.5)/100
 470 SLOW
 480 STOP
 500 PRINT AT L-U,Z;"ITEM ";I;

5/0
```

Figura 4.6: El programa de gráfico de barras (cont.)

culos de su gráfico. La invitación para el primer artículo tendrá este aspecto:

ARTICULO 1 = > NOMBRE

Usted debería introducir la palabra CARNE, primer artículo comestible. En cuanto lo haya introducido, la invitación cambiará a:

ARTICULO 1 = > NOMBRE: CARNE
CANTIDAD:

Ahora usted debe introducir el importe de lo gastado en carne durante el mes. Introduzca el número 27.53.

El programa continuará preguntándole el nombre y el

```
 510 PRINT "=> NAME: ";
 520 INPUT N$(I)
 530 PRINT N$(I)
 540 PRINT "AMOUNT:"
 550 INPUT A(I)
 560 PRINT AT L,Z;"          "
 570 RETURN
 600 PRINT  TAB ((32-LEN T$)/2);
T$
 610 PRINT
 620 RETURN
 700 PRINT "
           "
 710 RETURN


0/0
```

Figura 4.7: El programa gráfico de barras (cont.)

importe de cada artículo hasta que usted haya introducido los 15. La Figura 4.9. muestra los artículos que debería introducir después de CARNE.

Cuando haya introducido el último elemento, la pantalla se quedará en blanco un instante. Luego toda la información que muestra la Figura 4.9. aparecerá en la pantalla. Esto es un "eco" de todos los datos de input que usted suministró al programa. En el fondo de la pantalla, justo después de la tabla de datos, aparece la pregunta:

¿CORRECCION? < Y > OR < N >

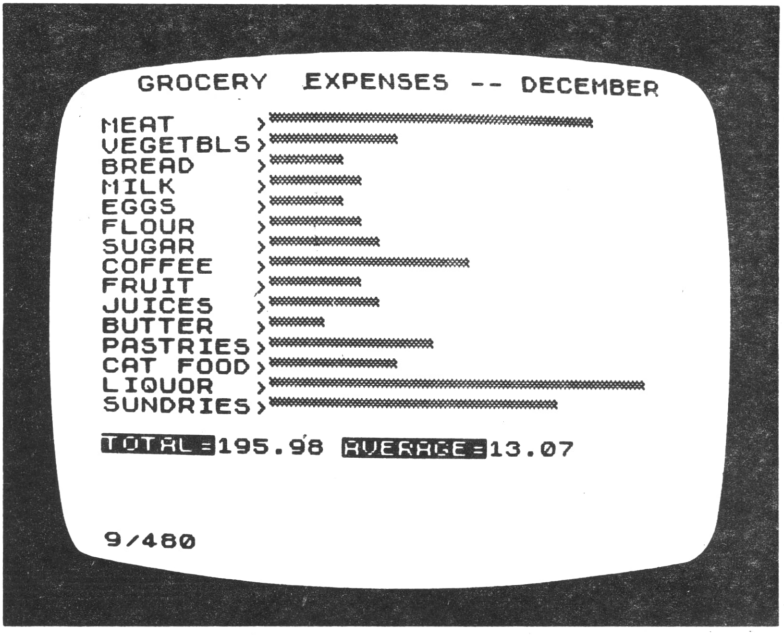

Figura 4.8: Pase de Muestra del programa de gráfico de barras y gastos de alimentación

Esta es la ocasión que se le presenta para examinar los datos en busca de los errores que pudiera cometer al escribir durante la fase de input. Si encuentra uno, sólo necesita escribir Y (para "yes" — sí—) a fin de indicar que tiene que proceder a una corrección.

En respuesta, el programa preguntará:

¿QUE NUMERO DE ARTICULO?

En este punto, usted debe escribir el número del artículo que desea cambiar. Para ver cómo funciona el proceso de corrección, cambiemos el importe que aparece para el artículo 9, FRUTA. Introduzca el número 9. El programa le invitará entonces a introducir el nombre y el número del artículo. (Es preciso introducir ambas piezas de información, incluso aunque uno sólo quiera cambiar una de ellas). Introduzca FRUTA para el nombre, y el nuevo valor 7.15 para el importe. La corrección se hará en la nueva visualización, de modo que usted pueda ver inmediatamente los resultados de su corrección. Puede proceder a tantas correcciones como quiera; la fase de corrección continúa hasta que usted introduce la letra N (primera de "no") en respuesta a la invitación a corregir.

Tan pronto como deje la fase de corrección del programa, la pantalla volverá a quedar en blanco unos momentos, y luego aparecerá en ella el gráfico de barras. Eche otra mirada a la Figura 4.8., el gráfico de barras de los gastos en comestibles. Advierta que, además del gráfico, el programa le da la cantidad total gastada durante el mes, y el promedio de gastos por artículo.

La Figura 4.10 muestra un segundo ejemplo de gráfico de barras. Esta vez el título es "Suministros de oficina-Gastos, 1982" y las barras representan los gastos mensuales en suministros de oficina, desde Enero a Diciembre. El gasto total para el año resulta de 7.439'86 dólares, y el promedio mensual de gastos es de 619'99 dólares.

En la sección siguiente de este capítulo examinaremos

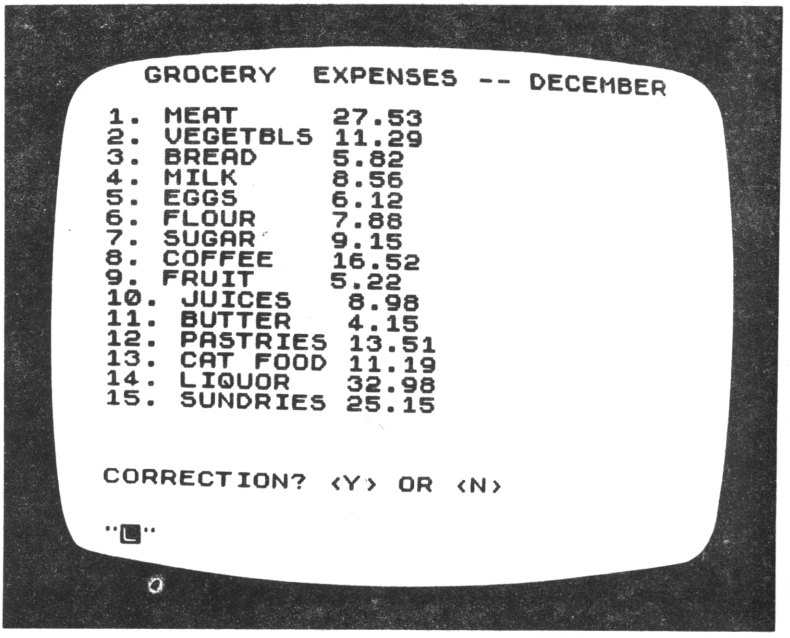

Figura 4.9: Los datos de alimentos

la estructura de este programa. Antes de leerla, usted quizá quiera pasar el programa nuevamente, introduciendo algunos datos reales suyos propios (de su vida doméstica, comercial o escolar) y producir un diagrama de barras que tenga significado para usted.

DENTRO DEL PROGRAMA DEL GRAFICO DE BARRAS

Una vez más, mientras lea esta descripción, puede escoger entre consultar nuevamente las figuras desde la 4.4. hasta la 4.7., o listar el programa, sección por sección, en su propia pantalla.

Este programa se aproxima más al punto de agotar la memoria disponible de su ordenador (2K solamente; no el módulo de memoria adicional). Por esta razón, en el comienzo

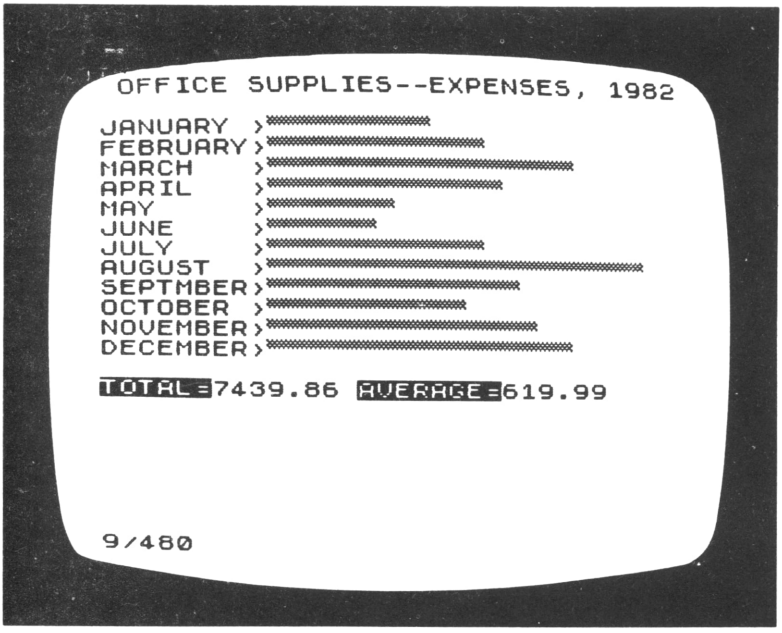

Figura 4.10: Segundo ejemplo gráfico de barras: gastos de oficina

del programa se definen tres variables que actúan como "constantes" a todo lo largo del mismo. Tal como vimos en el Capítulo 3, este es un método para ahorrar espacio de memoria substituyendo los valores numéricos literales por nombres de variables de una sola letra:

```
1 LET U = 1
2 LET Z = 0
3 LET L = 21
```

La variable L se usa como una dirección para colocar invitaciones en la pantalla.

Como otra manera de conservar memoria, no se ha colocado ninguna línea REM en el programa. Lo cual es una desgracia en un programa tan largo, pero a veces tendrá

usted que decidirse por este recurso. Para compensar la falta de comentarios en el listado, aquí tiene un esquema del programa, sección por sección:

• *Sección de iniciación* (líneas 1 a 70).

Esta sección define las tres variables "constantes"; lee el título y el número de elementos suministrados por el teclado; y dimensiona dos ordenaciones, N$ y A, para almacenar los datos del gráfico de barras.

• *Sección de input* (líneas 75 a 110).

Esta sección controla el input de todos los datos del gráfico de barras. Invoca la subrutina de la línea 500 una vez por cada elemento. A su vez, la subrutina de la línea 500 exhibe las invitaciones en la pantalla y lee el input procedente del teclado.

• *Sección de eco* (líneas 120 a 170).

Esta sección "reproduce" (hace el "eco") todos los datos de input de la pantalla en una tabla completa. Llama a la subrutina de la línea 600 para visualizar el título en la cima de la pantalla.

• *Sección de corrección* (líneas 180 a 270).

Esta sección controla el diálogo de corrección. Siempre que usted corrige una cosa, esta sección almacena los datos nuevos en los elementos apropiados de N$ y A, y visualiza estos datos nuevos en la pantalla. La sección de corrección invoca dos subrutinas: la subrutina de la 500. que suministra las invitaciones de input y lee datos del teclado, y la subrutina de la línea 700, que despeja una sola línea de la pantalla a fin de que se pueda visualizar allí información nueva.

• *Sección de cálculos intermedios* (líneas 280 a 360).

Esta sección calcula cuatro valores para los datos numéricos almacenados en la ordenación A. BIG es el dato mayor de la ordenación. TOT es la suma total de todos los valores de la ordenación. .AVE (abreviación de "average") es el promedio de los valores de la ordenación. El valor FAC es un factor de conversión que emplearemos para hacer cada barra del gráfico proporcional en longitud al número que represente.

• *La sección del gráfico de barras* (líneas 370 a 480).

Esta sección empieza con una llamada a la subrutina de la línea 600, que visualiza el título del gráfico de barras. Luego dibuja cada barra, calculando su longitud aproximada mediante el empleo del valor del factor FAC de conversión. Finalmente, esta sección visualiza los valores totales y de promedio de los datos.

• *Las subrutinas*

— en la línea 500: la subrutina de input;
— en la línea 600: la subrutina del título;
— en la línea 700: la subrutina de despeja-línea.

Empleando el esquema anterior como guía, usted debería estar en condiciones de leer el programa entero, y comprender el significado y el objetivo de la mayoría de líneas. Sin embargo, la mayoría de éstas quizás todavía requieran alguna explicación. Las notas siguientes clarificarán varias de las líneas más difíciles del programa:

Las líneas 60 y 70 contienen las declaraciones DIM. La línea 60 define la ordenación en cadena, N$, para almacenar los nombres de los artículos del gráfico de barras, y la línea 70 define las ordenaciones numéricas, A, para el importe de cada artículo:

```
60 DIM N$ (Q,8)
70 DIM A(Q)
```

La *longitud* de cada una de estas ordenaciones queda especificada por la variable Q. Durante el pase del programa, usted introduce (input) un valor para esta variable en respuesta a la pregunta: "¿CUANTOS ARTICULOS?":

```
30 PRINT AT L, Z; "¿CUANTOS ARTICULOS?"
40 INPUT Q
```

Si, por ejemplo, usted input el número 12 para la cantidad de artículos en el gráfico de barras, entonces la longitud de cada ordenación se fijará en 12, como si las líneas 60 y 70 estuvieran redactadas de la manera siguiente:

```
60 DIM N$(12,8)
70 DIM A(12)
```

Así pues, mediante el empleo de la variable Q, usted sabe que el espacio de memoria reservado para las dos ordenaciones será exactamente el que se necesite, ni más ni menos.

A estas dos ordenaciones se les asignan valores en el bucle FOR localizado desde la línea 80 hasta la 110:

```
80 FOR I = 1 TO Q
90 GOSUB 500
100 CLS
110 NEXT I
```

La línea FOR transforma la variable de control, I, desde 1 hasta el valor de Q. La subrutina de la línea 500 emplea esta variable de control para especificar el número correcto de artículos en el mensaje que visualiza en la pantalla:

```
500 INPUT N$(I)
.
.
.
550 INPUT A(I)
```

Estas líneas de input, y las de invitación (prompt) que las preceden, están aisladas en su propia subrutina porque se necesitan en dos momentos diferentes del programa: primero en la sección inicial de input; luego en la sección de corrección.

Otro bucle FOR, en las líneas de la 140 a la 160, visualiza toda la información de ambas ordenaciones en la pantalla:

```
140 FOR I = U TO Q
150 PRINT I; ".";N$(I); "";A(I)
160 NEXT I
```

La línea 150 usa la variable de control I tres veces: primero para visualizar el número del artículo, luego para acceder al valor exacto de N$, para el nombre del artículo; y finalmente para visualizar el importe del artículo, partiendo de la ordenación A.

Las líneas 310 a 340 determinan los valores mayores de la ordenación A (BIG), y la suma de todos los valores (TOT). Las dos variables empleadas para estos cálculos se inician primero a cero:

```
290 LET BIG = Z
300 LET TOT = Z
```

Luego otro bucle FOR se mueve a través de la ordenación entera desde 1 a Q:

```
310 FOR I = U TO Q
320 IF A(I) > BIG THEN LET BIG = A(I)
330 LET TOT = TOT + A(I)
340 NEXT I
```

La línea 320 compara cada valor de la ordenación con el valor actual de BIG. Cada vez que encuentra un valor mayor que BIG, ese valor se convierte en el nuevo BIG. La línea 330 acumula un total móvil de todos los valores.

El valor de promedio es el total dividido por el número de valores de Q:

```
350 LET AVE = TOT/Q
```

El factor de conversión proporcional se basa en el valor más elevado de la ordenación, BIG:

```
360 LET FAC = L/BIG
```

Como recordará usted, la variable L, es un valor "constante", igual a 21. De este modo la barra más larga del gráfico tendrá 21 caracteres de longitud. Como veremos pronto, el multiplicar cada valor de la ordenación A por el factor de conversión FAC dará un número comprendido entre cero y 21; este número, a su vez, determinará la longitud de la barra correspondiente del gráfico.

El gráfico de barras en sí queda visualizado en la pantalla por las líneas de la 390 a la 450. Estas líneas son dignas de un examen cuidadoso. Le muestran a usted un ejemplo de un bucle FOR *dentro* de otro bucle FOR. A tal disposición la llamamos a veces un *bucle anidado* (o acaso pudiera llamarse también un "bucle interior"). Hay dos reglas esenciales que usted debe tener en cuenta para el diseño de bucles anidados:

1. El bucle interior debe utilizar una variable de control diferente de la del bucle exterior.
2. Las declaraciones FOR y NEXT del bucle interior deben estar localizadas *ambas* dentro del bucle exterior.

Tenga presentes estas normas mientras examina las líneas que crean el gráfico de barras.

El bucle exterior, que empieza en la línea 390, controla el número del artículo, incrementando la variable, I, de control desde 1 a Q:

```
390 FOR I = U TO Q
```

Primero, el nombre del artículo se visualiza:

```
400 PRINT N$ (I); " > ";
```

Entonces el bucle interior crea una barra de longitud proporcional exacta para el importe, A(I):

```
410 FOR J = U TO INT (A(I) * FAC + .5)
```

La expresión A(I) * FAC resultará, como hemos visto, en un número situado entre 0 y 21. Añadiendo .5 y tomando el valor entero del resultado, podemos redondear este número hasta el entero más próximo:

```
INT (A(I) * FAC + .5)
```

De esta manera el bucle interior incrementa su variable de control, J, desde I hasta la longitud adecuada de la barra para representar el valor A(I). Cada vez que se incrementa J, la línea 42 aumenta la longitud de la barra en un carácter. (El carácter de los gráficos, es decir, el "bloque de construcción" de las barras, aparece en la tecla [S].

Para resumir, usted puede ver que por cada repetición del bucle exterior, el bucle interior entra en acción para crear una barra del diagrama de barras. Estas son, probablemente, las líneas que más importa que usted entienda en este programa. Si le cuesta trabajo entender cómo funcionan, quizás la siguiente analogía le ayude a captar el concepto de bucle anidado (o interior).

Piense en el funcionamiento de las tres manecillas de un reloj. Si usted define una vuelta alrededor de la esfera como

un "bucle", podrá ver tres niveles de bucles en el reloj. El bucle más interior es la manecilla de los segundos, que repite su acción 60 veces por cada bucle completo de la manecilla de los minutos. A su vez, la manecilla de los minutos, da 12 vueltas a la esfera por cada bucle de la manecilla de las horas. Así pues, hay tres niveles de repetición; el bucle más interno despliega la mayor actividad, y el más extremo controla la actividad de los otros dos. La acción de los bucles anidados en nuestro programa es similar. El bucle más externo, en la línea 390 controla la actividad del bucle interno, de la línea 410.

RESUMEN

Su ordenador le ofrece a usted cinco operaciones aritméticas (+, —, *,/ ,* *) y un surtido muy útil de funciones matemáticas. Usted puede combinar estas operaciones y funciones en expresiones complejas a fin de realizar cálculos. Al escribir tales expresiones, debe darse cuenta siempre del orden de operaciones establecido por el ordenador. Para saltarse el orden —o simplemente para clarificar una expresión y hacerla más legible— usted puede incluir paréntesis en expresiones aritméticas.

La modalidad rápida del ordenador puede resultar útil cuando usted quiere que la máquina se concentre en efectuar cálculos rápidos, tal como hemos visto en el programa supercalculador.

La *ordenación* es una estructura importante del BASIC que le permite a usted almacenar muchos valores bajo un solo nombre de variable. La ordenación es una herramienta ideal para utilizarla dentro de bucles FOR; la variable de control de un bucle FOR se puede utilizar como un *índice* dentro de la ordenación.

El programa del gráfico de barras presentado en este capítulo brinda un buen ejemplo del poder de los bucles FOR anidados (o interiores) así como varios ejemplos del empleo eficaz de ordenaciones dentro de bucles FOR.

CAPITULO 5
PALABRAS, PALABRAS, PALABRAS: SERIES Y FUNCIONES EN SERIE EN SU ORDENADOR

INTRODUCCION

En este capítulo final, examinaremos con más detalle las series (o cadenas) y las funciones y operaciones que se pueden efectuar en ellas. Como usted sabe, una serie es un elemento de datos que consiste en uno o más caracteres. Usted ya posee una experiencia considerable en series y variables en serie, gracias a los programas ofrecidos en los capítulos anteriores; pero todavía quedan varias funciones importantes sobre las cuales debe aprender algo.

Empezaremos el presente capítulo echando una ojeada al código de caracteres usado por el T/S 1000. Verá la manera de utilizar las funciones CODE (código) y CHR$ para pasar y repasar entre los números de código y los caracteres que representan. Mientras, examinará y pasará un par de programas cortos, aprenderá también el significado de la concatenación, y ensayará la utilísima función LEN.

Luego, preparándose para el último programa de este libro, descubrirá qué sucede cuando usted "corta una rebanada" de una serie (o cadena) y por qué es posible que quiera efectuar tal operación. El último programa, llamado de adopción de decisiones, es un elemento divertido que convierte al ordenador en el consejero y asesor privado de usted. De cualquier forma que lo corte en rebanadas, a este programa no se le ocurrirá arrastrarlo a usted, encadenado.

EL CODIGO DE CARACTERES DEL T/S 1000

El ordenador emplea su propio código para almacenar

—y distinguir— los caracteres del teclado. En algunas situaciones de programación, puede ser útil que usted conozca este código.

En él, a cada carácter se le asigna un número entre 0 y 255. En este contexto, el ordenador considera que cada elemento del teclado es un *carácter:* incluso las palabras clave y otras que pueden aparecer en el teclado. Las Figuras de 5.1. a 5.3. le muestran dicho código. En verdad, no hay motivo alguno para que éste se aprenda de memoria ninguna parte de este código; pero debería tomarse unos minutos para familiarizarse con él y tomar nota del emplazamiento general de las diversas categorías de caracteres. Por ejemplo, puede ver que los caracteres gráficos están emplazados en dos partes separadas del código (1 a 10, y

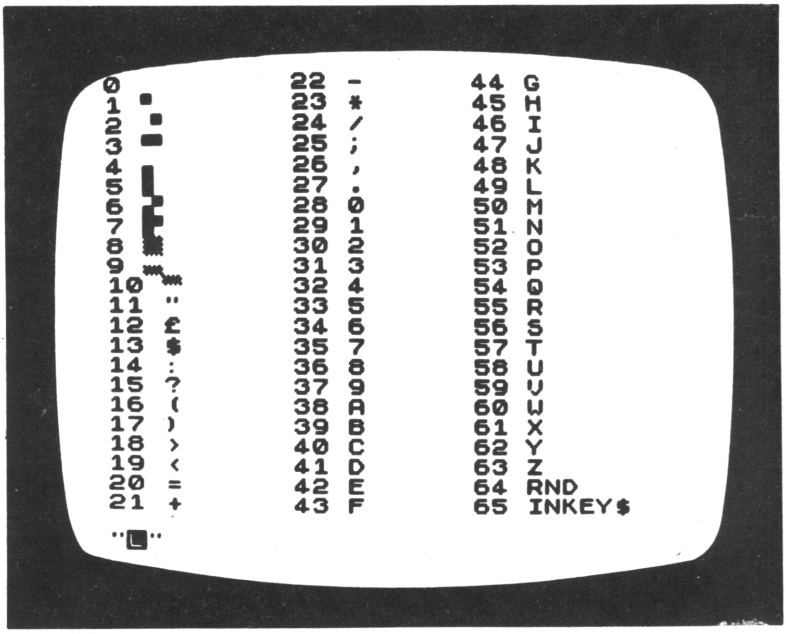

Figura 5.1: El código de caracteres

128 a 138); los dígitos y las letras están localizados cerca del comienzo del código (28 a 63); y los caracteres en vídeo inverso aparecen, aproximadamente, en la mitad del código (139 a 191). También puede hallar todas las palabras clave, funciones, operadores y otros símbolos del código.

Quizá se fije en que falta un gran trozo del código (de la 67 hasta la 127). Sencillamente, la mayoría de estos números que faltan no se utilizan; a unos cuantos los empleamos como 'teclas de control", tales como la tecla de dirección y la tecla DELETE, que no exhiben caracteres en la pantalla.

La función CHR$ suministra el carácter representado por un número de código. Por supuesto, el número ha de estar entre 0 y 255; CHR$ retorna el carácter, la palabra clave, la función o el símbolo correspondientes.

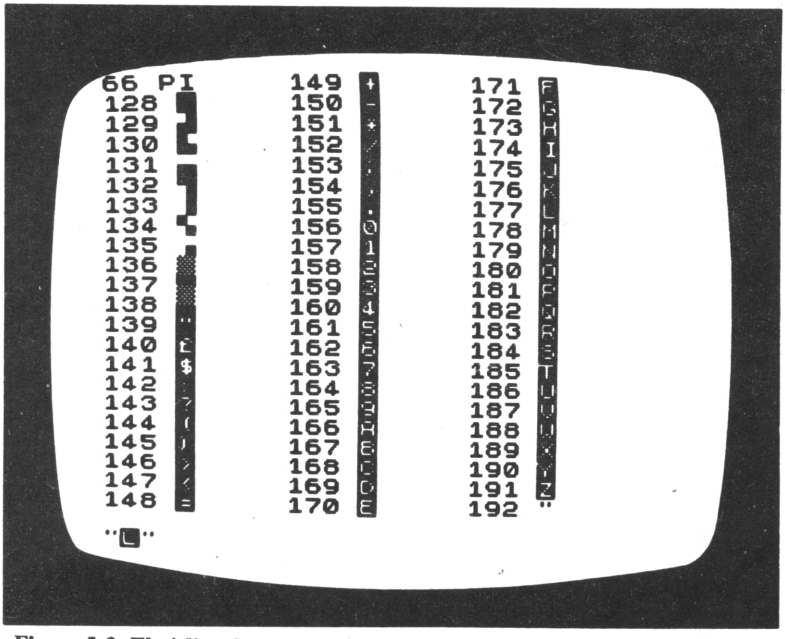

Figura 5.2: El código de caracteres (cont.)

El siguiente programa de seis líneas lee un número de código del teclado y luego exhibe el carácter en la pantalla, para que usted lo vea:

```
10 PRINT AT 21,0; "¿QUE NUMERO DE CODIGO?"
20 INPUT C
30 IF C > 255 THEN GOTO 20
40 CLS
50 PRINT AT 8,8; "CODE";C;"=";CHR$ C
60 GOTO 10
```

La línea 30 ensaya el número que usted escribe para asegurarse de que se trata de un número de código válido. La línea 50 utiliza la función CHR$ para visualizar el carácter.

Inversamente, la función CODE suministra el número de código de un carácter. Acaso usted recuerde CODE y CHR$ del Capítulo 2; el primer programa que introdujo usted en el ordenador empleaba ambas funciones para transformar una serie de letras en su equivalente de vídeo inverso. El programa es meramente un ejercicio; en realidad no hace nada útil. Pero usted puede aprender varias técnicas interesantes para manejar series, si se toma el tiempo suficiente para estudiarlo con cuidado y atención.

Algún ejemplo de output de este programa aparece en la Figura 5.5. Como usted puede ver, el programa le invita repetidamente a introducir una serie de caracteres en el ordenador. Después de cada input de una serie, el programa visualiza esta serie en vídeo inverso. El proceso de crear una serie (o cadena) en vídeo inverso lo ejecutan las líneas comprendidas entre la 50 y la 80, ambas inclusive, y constituye la parte interesante de este programa.

La variable en cadena W$ contiene la palabra de input. La variable R$ contendrá la videoinversión de la palabra.. En la línea 50, a la R$ se la inicializa como una *serie nula,* es decir, una serie o cadena que no contiene ningún carácter.

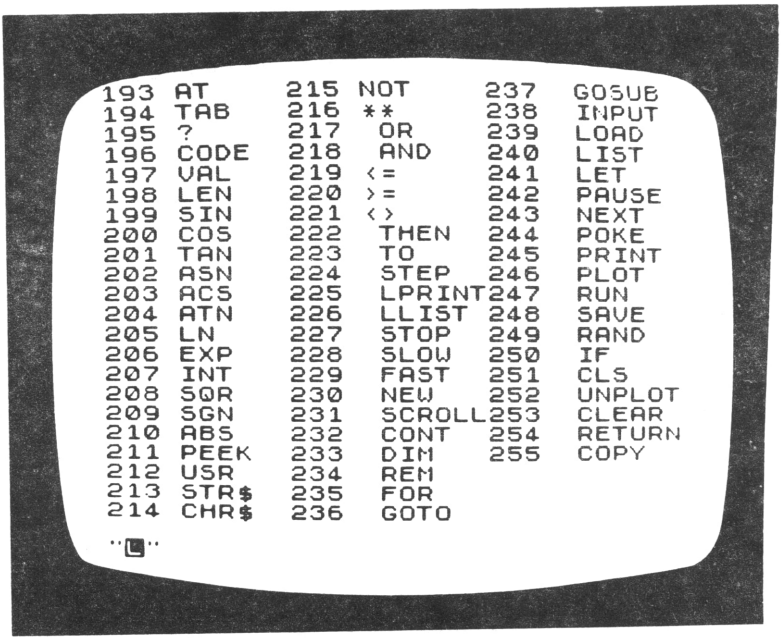

Figura 5.3: El código de caracteres (cont.)

```
50 LET R$ = " "
```

Luego, el bucle FOR de las líneas 60 hasta 80 encuentra el equivalente en vídeo inverso de cada carácter de W$, y añade el carácter al final de R$. La declaración FOR (línea 60) emplea la función LEN para determinar la longitud, en caracteres, de la cadena de input W$. Como puede usted ver, la variable de control, I, se incrementa desde 1 hasta LEN W$:

```
60 FOR I = 1 TO LEN W$
```

De esta manera maneja el bucle cada uno de los caracteres de W$, uno sólo cada vez. El BASIC le permitirá acceder a *un carácter* de una serie poniendo un número que repre-

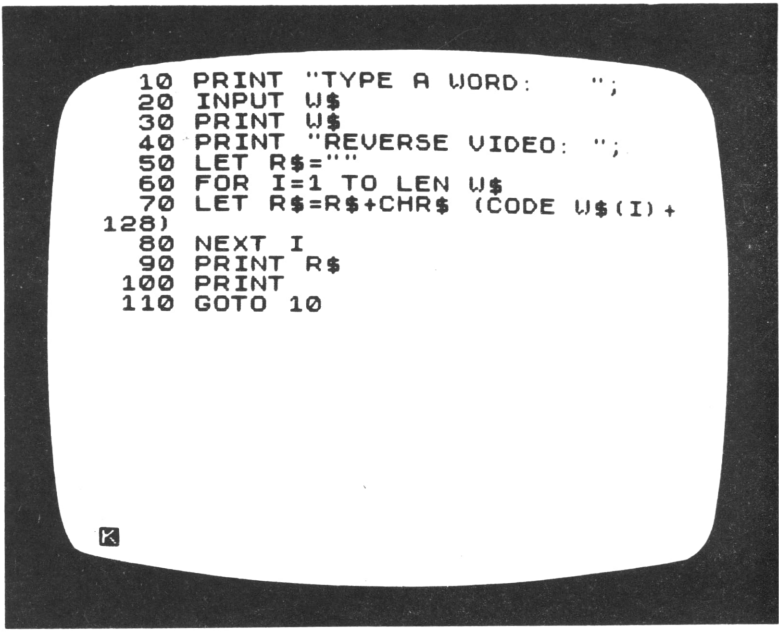

Figura 5.4: El programa de Video inverso

sente la situación del carácter, entre paréntesis, después del nombre de la variable en cadena. Así, W$(I) representa el carácter WS situado en la posición I.

Por ejemplo, si usted escribe una serie de tres caracteres en el teclado del ordenador, el bucle FOR incrementará I de 1 a 3. Los tres caracteres de la serie son W$(1), W$(2) y W$(3).

Ahora veamos la expresión, de la línea 70, que convierte un carácter en su equivalente de vídeo inverso:

CHR$(CODE W$(I) + 128)

Dentro del paréntesis, la función CODE suministra el número de código del carácter W$(1). Mire una vez más el código de caracteres de las Figuras 5.1., 5.2. y 5.3. Si su cadena de input se compone de signos de puntuación, dígitos o

letras, el número de código de cada carácter de la cadena se encontrará en algún punto situado entre 11 y 63. Si añade 128 a cualquiera de estos números de código, verá que el código resultante representa el equivalente en vídeo inverso del carácter. Pruebe un ejemplo. El código de carácter de la letra D es 41. Añadiendo 128 a 41 tenemos 169; no cabe duda, el código del carácter D en vídeo inverso es 169.

De modo que ahora usted puede ver exactamente el fruto de la expresión de conversión en la línea 70. Empieza añadiendo 128 al número de código del carácter en cuestión:

```
(CODE W$(I) + 128)
```

y luego utiliza la función CHR$ para encontrar el carácter representado por el código nuevo:

```
CHR$(CODE W$(I) + 128)
```

Una vez determinado el nuevo carácter, se añade a la cadena de vídeo inverso que se está acumulando en la variable R$. El combinar dos cadenas de este modo se llama *concatenación*. El proceso es indicado por el símbolo más , como en la línea 70:

```
70 LET R$ = R$ + CHR$(CODE W$(I) + 128)
```

Esta declaración LET dice: "Añade el carácter nuevo al final del valor de la cadena actual almacenado en R$; luego almacena la nueva cadena, que tiene un carácter más que antes, nuevamente en R$.

La conversión a vídeo inverso es completa cuando I equivale a LEN W$; la acción del bucle FOR ha terminado, y la línea 90 exhibe R$ en la pantalla:

```
90 PRINT R$
```

Para resumir, las líneas desde 50 a 80 de este programa,

pueden enseñarle varios puntos importantes sobre cadenas (o series) y su manejo:

- Usted puede iniciar una variable en cadena como una *variable nula,* significando una cadena que no contiene ningún carácter (línea 50).
- La función LEN suministra la longitud, en caracteres, de una cadena (línea 60).
- Usted puede acceder a un solo carácter de una cadena especificando la posición del carácter, entre paréntesis (línea 70).
- Los códigos de los caracteres en vídeo inverso tienen todos números que superan en 128 unidades a los de los caracteres regulares (línea 70).
- Se puede concatenar dos cadenas (es decir, combinarlas) utilizando el símbolo " " (línea 70).

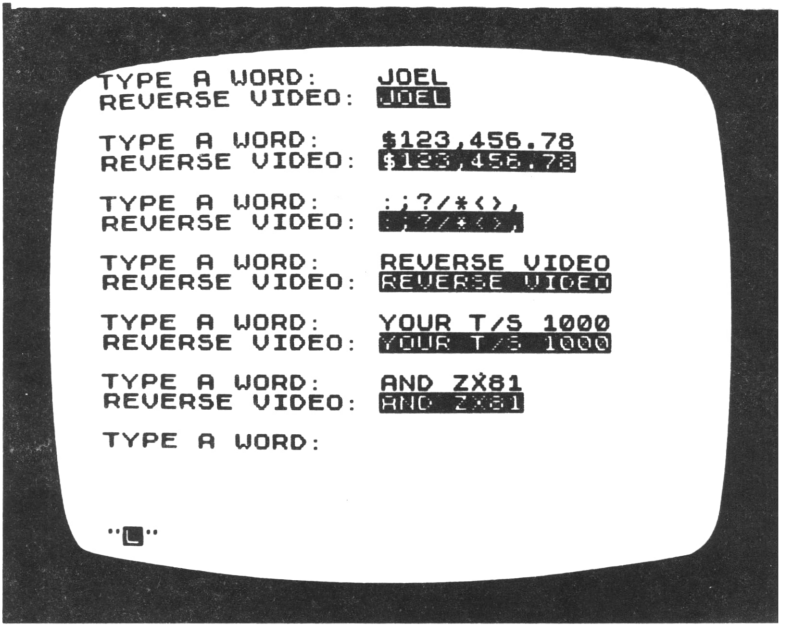

Figura 5.5: Output de muestra del programa de vídeo inverso

En la sección siguiente verá usted que el ordenador también le proporciona una manera de acceder a varios caracteres consecutivos de una cadena, en un proceso llamado *rebanado* (en inglés *slicing*).

REBANAR (SLICING) UNA CADENA

A menudo es conveniente poder identificar una *subcadena* de una cadena más larga, por diversos motivos. Uno quizá quiera examinar una cadena para ver si contiene cierta secuencia de caracteres, o acaso quiera cambiar los valores de ciertas posiciones en el interior de una cadena. En el programa de adopción de decisiones, al final de este capítulo, querremos, simplemente, acceder a porciones relativamente pequeñas de una cadena larga, a fin de visualizarlas. La técnica del *rebanado* le permite efectuar todas estas cosas sencilla y eficientemente.

Haremos pruebas de rebanado en una serie de mandatos inmediatos. Asegúrese de escribir el mandato en el teclado de su ordenador de manera que pueda ver realmente el resultado del rebanado.

Empiece definiendo la cadena W$, como sigue:

```
LET W$ = "FUNDAMENTAL"
```

Ahora introduzca el mandato:

```
PRINT W$
```

para asegurarse de que la palabra FUNDAMENTAL esté almacenada realmente en la memoria bajo el nombre de variable W$.

La sintaxis para el rebanado (slicing) utiliza la palabra TO, entre paréntesis, después del nombre de la variable en cadena. En general, la expresión:

```
S$ (M TO N)
```

identifica la porción de cadena S$ situada en las posiciones de M hasta N. M y N pueden ser números literales, variables numéricas. A veces designamos la porción rebanada de la cadena como una *subcadena* (o *subserie*) *(substring).*

Como primer ensayo de rebanado, escriba este mandato:

```
PRINT W$ (5 TO 8)
```

Antes de apretar la tecla ENTER, pruebe de predecir cuál será el fruto de este mandato, contando las posiciones de la palabra FUNDAMENTAL. El mandato debería visualizar la palabra AMEN en la cima de la pantalla.

Usted puede omitir el primer número de posición, o el último, en la cláusula TO. Si omite el primero, la rebanada empezará con el primer carácter de la cadena:

```
PRINT W$(TO 3)
```

Este mandato da la palabra (las tres letras) FUN. Si omite el último número, la rebanada se extiende hasta el final de la cadena. Por ejemplo:

```
PRINT W$(6 TO)
```

Al introducir este mandato, usted debería ver la palabra MENTAL en la pantalla.

También se puede *revisar* una rebanada de una cadena. Para ello, usted debe especificar la rebanada que desea revisar en una declaración LET, como sigue:

```
LET W$(TO 5) = "INCRE"
```

Escriba este mandato y luego introduzca:

```
PRINT W$
```

para ver qué le ha pasado a su cadena. Debería ver la pala-

bra INCREMENTAL. Los cinco caracteres primeros de la cadena han sido reemplazados por otros nuevos.

El programa adoptador de decisiones utiliza la técnica del rebanado (slicing) para elegir una respuesta —de una larga cadena de respuestas— para una pregunta que usted le formulará. El desilusionador secreto de tal programa es el siguiente: Elige las respuestas al azar, por completo. Así pues, antes de examinar el programa, hemos de dirigir una rápida mirada a los números al azar de su ordenador.

NUMEROS AL AZAR

La función RND, situada debajo de la tecla [T], devuelve un *número al azar* entre 0 y 1, y devuelve números diferentes cada vez que usted la emplea. Para ver RND en acción, pase el siguiente programa de dos líneas:

```
10 PRINT RND
20 GOTO 10
```

Una columna de números al azar aparecerá en la pantalla; el programa continuará hasta que no quede espacio en la columna. En principio, los números al azar se eligen de una manera completamente impredictible. En realidad, el ordenador emplea una fórmula matemática para *computar* los números suministrados por RND. Esto significa que los números no son verdaderamente al azar; sin embargo, la fórmula está ideada para suministrar números que siempre *parezcan* al azar. Para los programas que escribirá usted en su ordenador, este azar *aparente* bastará y sobrará siempre, probablemente.

En ordenadores personales pequeños tales como el T/S 1000, el empleo más frecuente de la función al azar suele consistir en simular acontecimientos de tal índole en programas de juegos. Por ejemplo, usted puede usar RND para simular una echada de dados, o una mano de naipes barajados al azar.

Para crear estas simulaciones, necesitamos una manera de convertir el *campo* de los números de azar generado por RND. Los números al azar entre 0 y 1 quizás no sean muy útiles; pero si podemos convertirlos en enteros al azar entre 1 y 6, por ejemplo, entonces podremos utilizarlos en ciertas clases de juegos.

La fórmula para efectuar tal conversión es relativamente sencilla. Si usted tiene un número al azar mayor que 0 y menor que 1 y lo multiplica por 6, tendrá entonces un número al azar comprendido entre 0 y 6:

```
RND * 6
```

Si usted trunca este número, tomando la parte entera (por medio de la función INT) le quedará un número entero, al azar, que puede ir de 0 a 5:

```
INT (RND * 6)
```

Finalmente, adicionando 1 al resultado, produce un entero al azar comprendido entre ¿ y 6, ambos inclusive:

```
INT (RND * 6) + 1
```

Para ver esta fórmula en acción, introduzca y pase el programa corto que mostramos en la Figura 5.6. Este programa crea un juego de adivinanza muy elemental. El ordenador empieza exhibiendo el siguiente mensaje en la pantalla:

```
YO ESTOY PENSANDO UN NUMERO
DE 1 A 6. ¿SABE ADIVINARLO?
```

Usted introduce una suposición en forma de un entero comprendido entre 1 y 6. Luego el ordenador le dice si la suposición ha sido acertada o errónea. Por ejemplo, si usted supone 4, verá un mensaje tal como:

BIEN. EL NUMERO ERA 4

o

LO SIENTO. EL NUMERO ERA 3.

Aunque este juego es muy sencillo, le enseña cómo puede utilizarse el RND para producir resultados imprevisibles mediante el ordenador. Advierta que la línea 5 del programa produce el número al azar, usando la fórmula que hemos elaborado antes:

```
5 LET R = INT (RND * 6) + 1
```

Luego verá una fórmula parecida en el programa de adopción de decisiones.

EL QUE TOMA DECISIONES

Usted puede utilizar el programa que toma decisiones como un juego de salón, para divertir a sus amigos con la maravillosa capacidad del ordenador para predecir el futuro, resolver dilemas personales, o tomar decisiones rápidas en asuntos de negocios. Para hacer funcionar el programa, escriba mediante el teclado una pregunta de "sí o no"; el ordenador se parará a meditar unos instantes, y luego exhibirá su "cuidadosa" respuesta en la pantalla.

Por supuesto, el oscuro secreto que ahora usted comparte con su ordenador es el de que las respuestas han sido elegidas al azar, y no tienen relación alguna con la naturaleza de las preguntas. Pero mantengámoslo en secreto.

La Figura 5.7. nos ofrece una pantalla de muestra de este programa. Después de haber introducido usted una pregunta, el ordenador la reproduce en la cima de la pantalla, y visualiza su respuesta dentro de la caja de decisiones. Luego el ordenador espera la siguiente pregunta

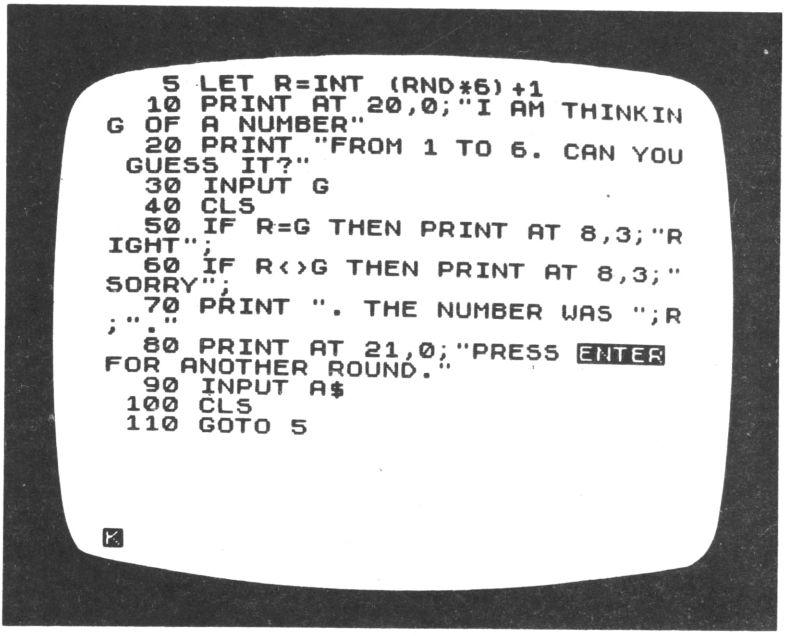

Figura 5.6: El programa del juego de adivinanzas

de usted, y puede pasarse el día respondiendo a sus preguntas, si usted lo quiere.

El listado del programa aparece en la Figura 5.8. Examinaremos las tres importantes técnicas ilustradas en este programa: el uso de RND para elegir al azar entre varias respuestas; el uso del rebanado (slicing) para acceder a la respuesta acertada, y finalmente, el uso de la función LEN para centrar una cadena horizontalmente en la pantalla.

Empecemos estudiando bien la línea 10. Esta define la "cadena de respuestas", A$. La cadena tiene, exactamente, 60 caracteres de longitud y está compuesta de seis secciones de 10 caracteres, representando las seis respuestas diferentes que el ordenador puede dar a sus preguntas: (Ponemos las equivalencias inglesas entre paréntesis para poder

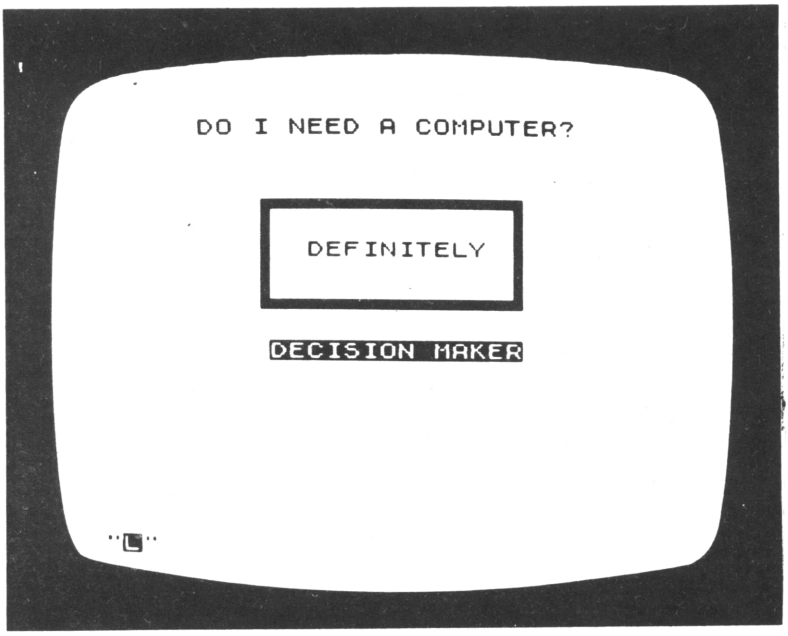

Figura 5.7: Pantalla de muestra del Adoptador de Decisiones

reproducir la figura 5.8. tal como en el original. El lector consultará las expresiones que no entienda).

SI	(YES)
NO	(NO
QUIZAS	(PERHAPS)
CIERTAMENTE	(DEFINITELY)
¿POR QUE NO?	WHY NOT?
PREGUNTE DE NUEVO	ASK AGAIN

Como las longitudes de las respuestas son distintas, están rodeadas de espacios en la cadena, los cuales las llenan hasta el total de 10 caracteres. Así, la primera respuesta empieza en A$(1) y va hasta A$(10); la segunda va desde A$(11) hasta A$(20); la tercera de A$(21) hasta A$(30); y

así sucesivamente. Veremos que esta regularidad es la clave para elegir respuestas de la cadena.

La línea 20 visualiza el título del programa. Las líneas de la 30 a la 100 dibujan la "caja de decisión" en la pantalla, usando dos bucles FOR y cuatro declaraciones PLOT.

La línea 110 lee la rpegunta de usted en el teclado y la destina a la variable en cadena Q$:

```
110 INPUT Q$
```

Las líneas 150 y 160 eligen la respuesta. La línea 150 utiliza el RND en una fórmula confeccionada específicamente para la conveniencia del proceso de rebanado:

```
150 LET R = 10 * INT (RND * 6) + 1
```

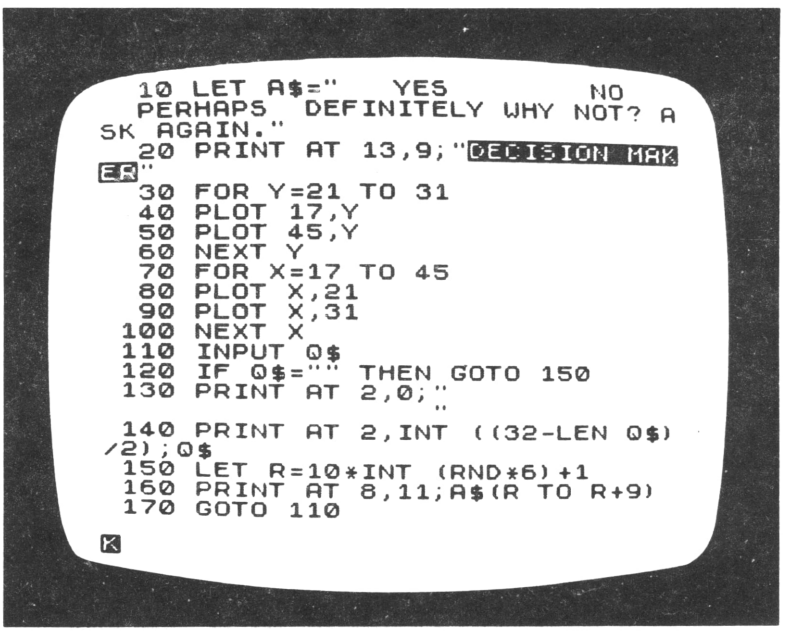

Figura 5.8: El programa de decisiones

Hemos visto que la expresión INT (RND * 6) produce un entero al azar desde 0 a 5. Multiplicar este resultado por 10 nos da un múltiplo de 10 desde 0 a 50. Si añadimos 1, tendremos uno de los seis números siguientes:

```
1 11 21 31 41 51
```

Estos números, tal como hemos visto, representan los respectivos puntos de partida de las seis respuestas diferentes contenidas en la cadena (o serie) A$. De este modo, la línea 150 elige uno de estos puntos de partida, al azar, y asigna su valor a la variable R.

Una vez determinado este punto de partida, el acceder a la respuesta es un simple procedimiento de rebanado. Se efectúa en la línea 160:

```
160 PRINT AT 8,11;A$(R TO R + 9)
```

Esta línea escoge la subserie de 10 caracteres de A$ desde la posición R a la posición R 9; luego visualiza esta subcadena en la pantalla, dentro de la caja de ordenaciones. Así, gracias a la variable R, la línea 160 exhibe en la pantalla una línea escogida al azar. Entonces la línea 170 empieza otra ronda, devolviendo el control del programa a la declaración INPUT:

```
170 GOTO 110
```

Una línea más de este programa merece cierto análisis. La línea 140, que visualiza la pregunta de usted en la cima de la pantalla, contiene una fórmula interesante para *centrar* horizontalmente la cadena Q$:

```
140 PRINT AT 2, INT((32—LENQ$)/2);Q$
```

(Si usted fuese muy observador, quizás se habría fijado en una fórmula similar en el programa de gráfico de barras

del Capítulo 4). La cláusula AT de esta declaración PRINT pone la serie en la fila 2; la posición de la columna se calcula en una fórmula basada en la longitud de la serie:

```
INT ((32—LEN Q$)/2)
```

Esta fórmula asume que la pregunta en cadena Q$ no tiene más de 32 caracteres de longitud. La función LEN suministra el número de caracteres en Q$. Dado que la pantalla tiene 32 columnas de anchura, la expresión (32—LEN Q$)/2 da la dirección de la columna donde debe empezar la cadena a fin de que quede centrada horizontalmente. Por ejemplo, digamos que usted introduce una cadena de pregunta de 20 caracteres de longitud; para centrar la cadena, usted necesitaría 6 espacios a ambos lados de la pregunta. Puede ver que la fórmula (32—20)/2 le da el punto de partida acertado.

RESUMEN

Su ordenador codifica cada carácter del teclado como un número entre 0 y 255. Para la mayoría de situaciones de programación usted puede ignorar este código; el ordenador lo emplea muy en secreto, y usted ni siquiera tiene que darse cuenta de su existencia. Pero para las ocasiones en que el código le puede ser útil, las funciones CODE y CHR$ le ayudarán en la conversión entre caracteres y números de código.

Concatenar significa combinar series (cadenas) para crear una serie nueva. El rebanado es el proceso de extraer o identificar una subserie. Hemos visto ejemplos de ambos procesos en los programas de este capítulo.

En este punto, usted debería estar en muy buenas relaciones con su ordenador, su funcionamiento y sus facultades computadoras. Está ya preparado para pasar cualquiera de los programas disponibles en el comercio para su ordenador, y para escribirlos por su cuenta. Posee un conocimiento

práctico del BASIC, lenguaje de programación rico y elocuente. En resumen, usted no es sólo el *propietario* de un ordenador, sino también el *usuario*.

APENDICE A
EL VOCABULARIO BASIC

Este apéndice define las palabras que aparecen en el teclado del T/S 1000, y, en muchos casos, da ejemplos de declaraciones BASIC que incluyen las palabras. El emplazamiento en el teclado de cada palabra se especifica junto con su categoría (palabra clave, función, o cambio).

ABS (función; tecla [G]). Proporciona el *valor absoluto* de un número.

AND (shift —cambio—; tecla [2]). Crea declaraciones compuestas "verdadero o falso" para decisiones IF. Por ejemplo:

```
IF I > 0 Y I < 100 THEN GOSUB 300
```

En este ejemplo, la declaración GOSUB después de THEN solamente se ejecutará si tanto la declaración I > 0 *como* I 100 son ciertas.

ARCCOS (función; tecla [S]). Retorna el arco coseno de un número entre —1 y 1. El resultado es en radianes. (1 radián 57.3 grados). ARCCOS se visualiza como ACS en la pantalla.

ARCSIN (función; tecla [A]). Retorna el arco seno de un número entre —1 y 1. El resultado se da en radianes. ARCSIN se visualiza como ASN en la pantalla.

ARCTAN (función; tecla [D]). Retorna el arco tangente en radianes. ARCTAN se visualiza como ATN en la pantalla.

AT (función; tecla [C]). Usada con la declaración PRINT para especificar un emplazamiento de visualización en

la pantalla de un carácter, cadena, o dígito. AT debe ir seguido de una dirección bajo la forma de *fila, columna,* donde *fila* es un número comprendido entre 0 y 21 y *columna* es un número comprendido entre 0 y 31. Por ejemplo, la declaración:

```
PRINT AT 8,10; "HELLO"
```

visualiza la palabra HELLO en la fila 8, empezando en la columna 10. (La dirección 0,0 representa el rincón superior izquierdo de la pantalla).

BREAK (tecla [SPACE]). Interrumpe el pase de un programa que está en marcha. BREAK no se puede emplear mientras el ordenador espera input del teclado.

CHR$ (función; tecla [U]). Suministra el carácter correspondiente a un número de código entre 0 y 225. Palabras clave, funciones, caracteres y símbolos están todos incluidos en el código de caracteres del ordenador.

CLEAR (palabra clave; tecla [X]). Despeja todas las variables y sus valores de la memoria; *no* despeja (no borra) líneas del programa.

CLS (palabra clave; tecla [V]). El mandato de "limpia pantalla" (clearscreen) borra toda información de la pantalla; una declaración PRINT subsiguiente visualizará información empezando en el ángulo superior izquierdo de la pantalla (posición 0,0).

CODE (función; tecla [I]). Suministra el número de código de un carácter dado. Los números de código van de O a 255.

CONT (palabra clave; tecla [C]). Continúa un pase de programa que se interrumpió por algún motivo.

COPY (palabra clave; tecla [Z]). Envía los contenidos actuales de la visualización de la pantalla a la impresora.

COS (función; tecla [W]). Suministra el coseno de un número que representa un ángulo. La función COS presupone que el ángulo está expresado en radianes. (1 grado .0175 radianes).

DELETE (shift —cambio—; tecla [0]). Borra el carácter, palabra clave o función inmediata al costado izquierdo de la pista. DELETE se puede usar siempre que se introduce una línea nueva en el ordenador, o cuando se corrige una línea completa.

DIM (palabra clave; tecla [D]). Define el nombre, tipo, dimensiones y longitud de una ordenación (array). Por ejemplo:

```
DIM N(10)
```

define N, ordenación numérica unidimensional de longitud 10. En el caso de ordenaciones en serie, todos los elementos de la serie deben tener la misma longitud, la cual debe especificarse en la declaración DIM. Por ejemplo:

```
DIM A$(10,5)
```

define A$, ordenación en cadena de una dimensión. A$ puede contener hasta 10 cadenas, cada una compuesta de 5 caracteres.

EDIT (shift; tecla [1]). Exhibe una copia de la línea actual en el fondo de la pantalla, preparada para la corrección. La línea actual está marcada por un carácter pequeño de vídeo inverso " " en el listado del programa.

ENERGIA, suministro de, 3-4, 8.

ENTER Introduce líneas de programa o información input en la memoria del ordenador.

EXP (función; tecla [X]).Suministra el "exponente natural" de un número. (Calcula un exponente de e, en el que e 2'7182818).

FAST (shift —cambio o conmutado—; tecla [F]). Pone al ordenador en la modalidad de cálculo rápido.

FOR (palabra clave; tecla [F]). Crea un bucle de repetición, motivando que las líneas que están dentro del bucle se repitan un determinado número de veces. (El final del

bucle está señalado por una declaración NEXT). Por ejemplo:

```
FOR I = 2 TO 20 STEP 2
```

La *variable de control* es I en esta declaración FOR. Durante la repetición del bucle, I aumentará de valor desde 2 hasta 20 en escalones de 2.

FUNCTION (shift; tecla [ENTER]). Pone el teclado en el modo de función; la pulsación siguiente se registrará como una tecla de función.

GOSUB (palabra clave; tecla [H]). Envía el control del programa a una subrutina. La primera línea de la subrutina se puede expresar como un número literal:

```
GOSUB 300
```

o una variable numérica:

```
GOSUB T
```

o hasta como una expresión numérica, para un GOSUB "calculado":

```
GOSUB 100 * (VAL I$ + 4)
```

GOTO (palabra clave; [G]). Envía el control del programa a una línea especificada. El número de línea se puede expresar como un valor numérico literal, como una variable numérica, o una expresión numérica. (Vea ejemplos en GOSUB).

GRAPHICS (shift; tecla [9]). Pone el teclado en la modalidad de gráficos; las pulsaciones subsiguientes quedarán registradas como caracteres gráficos o caracteres en vídeo inverso. La tecla de GRAPHICS también debe usarse para sacar el teclado de la modalidad de gráficos y pasarlo nuevamente a la de letras.

IF (palabra clave; tecla [U]). Introduce una declaración de decisión. Una declaración IF empieza con una declaración de "verdadero o falso", seguida de una cláusula THEN; por ejemplo:

```
IF D <> THEN LET Q = 1/D
```

Con esta declaración, el ordenador evalúa primero D < > 0 para verdadero o falso. Si la declaración es cierta, se ejecuta la de LET (después de THEN); si es falsa, el control del programa sigue hacia la línea siguiente.

INKEY$ (función; tecla [I]). Manda al ordenador que lea datos del teclado y los almacene en una variable especial. Por ejemplo:

```
INPUT S$
```

lee un input en serie procedente del teclado, y almacena el valor en la variable S$. INPUT hace que los datos de input hallen eco (aparezcan) en la pantalla al ser introducidos.

INT (función; tecla [R]). Proporciona el valor entero (o sea, la parte entera) de un número.

LEN (función; tecla [K]). Suministra la longitud, en caracteres, de una cadena.

LET (palabra clave; tecla [L]). Asigna un valor a una variable. Esta variable puede ser, o bien una nueva, que recibe su primer valor, o bien una antigua, que recibe un valor nuevo. A la izquierda del signo igual, la declaración LET debe contener un solo nombre de variable, en la parte de la derecha puede contener literales, variables o expresiones. Por ejemplo:

```
LET V = 5
LET A = B
LET X = (5 * Y)/2
```

Cualesquiera variables mencionadas a la derecha del signo igual deben ser definidas antes de ejecutarse la declaración LET. Esta declaración LET no cambia los valores de las variables situadas a la derecha del signo igual.

LIST (palabra clave; tecla [K]). Visualiza el programa actual en la pantalla. Si LIST va seguido de un número de línea, como en:

```
LIST 100
```

el listado del programa empieza en dicha línea.

LLIST (shift; tecla [G]). Envía el listado del programa a la impresora.

LN (función; tecla [Z]). Proporciona el logaritmo natural (base e) de un número positivo.

LOAD (palabra clave; tecla [J]). Recupera un programa guardado en una cinta de cassette. El mandato LOAD se puede usar en dos formas; con el nombre del programa:

```
LOAD "NAME"
```

o sin el nombre:

```
LOAD""
```

En ambas formas se precisan las comillas. En el segundo caso, el ordenador carga el primer programa que encuentra en la cinta de cassette.

LPRINT (shift; tecla [S]). Envía una línea de información a la impresora.

NEW (palabra clave; tecla [A]). Borra el programa actual de la memoria del ordenador.

NEXT (palabra clave; tecla [N]). Señala el final de un bucle FOR. La variable de control, definida en la declaración FOR, debe seguir a la palabra NEXT. Por ejemplo:

NEXT I.

NOT (función; tecla [N]). Modifica una declaración de "verdadero o falso" en una decisión IF; por ejemplo:

IF NOT D > 0 THEN GOTO 10

Esta declaración equivale a:

IF D < 0 THEN GOTO 10

OR (shift; tecla [W]). Crea declaraciones complicadas de "verdadero o falso" para decisiones IF. En el ejemplo siguiente, OR conecta dos igualdades:

IF X = 5 OR y = 3 THEN GOSUB 300

La declaración GOSUB expresada después de la palabra THEN se ejecutará si una de las dos igualdades es cierta, o lo son ambas.

PAUSE (palabra clave; tecla [M]). Manda al ordenador que haga una pausa en la ejecución del programa. La longitud de la pausa queda determinada por un entero que sigue a la palabra clave PAUSE. Por ejemplo:

PAUSE 50

origina una pausa de un segundo, aproximadamente y aumentando el número con múltiplos de 50 se incrementará la pausa en los correspondientes segundos adicionales. Si el número que sigue a PAUSE es mayor que 32766, el mandato significa "pausa eterna". Una pausa se puede interrumpir, y se reanuda el programa, apretando una tecla del teclado.

PEEK (función; tecla [0]). Devuelve los contenidos de un emplazamiento de memoria desde 0 hasta 65535.

PLOT (palabra clave; tecla [Q]). Visualiza un "pixel" en la

pantalla en una dirección determinada (un lugar determinado). Los elementos de la dirección PLOT están en orden inverso de los de la PRINT AT. La forma general del mandato PLOT es:

PLOT *h,v*

donde *h* es la dirección horizontal (de 0 a 63) y *v* es la dirección vertical (de 0 a 43). La dirección PLOT 0,0 está situada en el rincón inferior izquierdo de la pantalla.

POKE (palabra clave; tecla [0]). Escribe un valor en un emplazamiento especial de la memoria desde 0 hasta 65535. Toma la forma:

POKE *a,v*

donde *a* es la dirección del emplazamiento en la memoria y *v* es el valor. El valor absoluto de *v* no puede pasar de 255.

PRINT (palabra clave; tecla [P]). Envía una línea de información a la pantalla. Un punto y coma (;) separando elementos PRINT visualiza los elementos uno junto al otro, sin espacio de separación entre ellos. Una coma (,) entre elementos produce un desplazamiento (tab) adelante hasta el centro de la pantalla o hasta el comienzo de la próxima línea de la pantalla, hasta el que esté más cerca.

RAND (palabra clave; tecla [T]. Controla el punto de partida de una secuencia de números al azar producida por la función RND. Las declaraciones:

RAND

y

RAND 0

originan un punto de partida impredecible. La declaración:

RAND *n*

donde *n* es mayor que 0 y menor que 65536, origina un punto de partida específico y predecible.

REM (palabra clave; tecla [E]). Crea una línea de comentario ("remark") en el listado del programa. El ordenador ignora las líneas REM durante el pase del programa. Después de la palabra REM puede situarse cualquier clase de comentario.

RETURN (palabra clave; tecla [Y]. Indica el final de una subrutina. Envía el control del programa hacia la línea *siguiente* de la declaración GOSUB que originariamente llamó a la subrutina.

RND (función; tecla [T]). Produce un número "al azar" mayor que cero, o igual a cero, y menor que 1.

RUN (palabra clave; tecla [R]). Manda al ordenador que empiece ejecutando las líneas del programa que actualmente tiene en la memoria. (Empieza realizando un CLEAR de todas las variables de los pases de programas anteriores). Si RUN va seguido de una línea de programa, el ordenador empieza el pase del programa en la línea indicada.

SAVE (palabra clave; tecla [S]). Almacena un programa en una cinta de cassette. El mandato de SAVE requiere un nombre de programa, entre comillas:

SAVE "NAME"

SCROLL (palabra clave; tecla [B]). Hace ascender el espacio de una línea toda la información visualizada en la pantalla. La línea superior se pierde, y en el fondo de la pantalla aparece una línea en blanco.

SGN (función; tecla [F]). Proporciona el *signo* de cualquier número; origina un valor de —1, 0 ó 1.

SLOW (shift; tecla [D]). Retorna el ordenador a la modalidad de cálculo lento.

SPACE Sitúa un carácter de espacio en una línea de programa o en una cadena de input.

SQR (función; tecla [H]). Da la raíz cuadrada de un número no negativo.

STEP (shift; tecla [E]). En una declaración FOR, indica la cantidad de incremento (o decremento) para la variable de control. Por ejemplo:

```
FOR I 5 TO 25 STEP 5
```

En esta declaración la variable de control I aumentará en cinco unidades para cada repetición del bucle. Si la cláusula STEP falta en una declaración FOR, el valor del incremento por defecto es 1.

STOP (shift; tecla [A]). Interrumpe la ejecución del programa. La palabra STOP puede formar parte de una línea de programa, o puede ser introducida (input) por el teclado para terminar el pase de un programa.

STR$ (función; tecla [Y]). Proporciona la versión en cadena de un número. Por ejemplo:

```
LET S$ STR$ 1234
```

Esta declaración originará que el valor de cadena "1234" quede almacenado en la variable S$.

TAB (función; tecla [P]). Usada con la declaración PRINT para exhibir un mensaje que empiece en una columna especificada de la línea en curso. Por ejemplo:

```
PRINT TAB 15; "HELLO"
```

imprimirá HELLO empezando en la columna 15.

TAN (función; tecla [E]). Suministra la tangente de un número que representa un ángulo. TAN da por supuesto que el ángulo está en radianes (1 grado .01745 radianes).

THEN (shift; tecla [3]). Introduce el resultado de una declaración IF. A THEN le sigue siempre una palabra clave.

La declaración que sigue a THEN sólo se ejecuta si la declaración después de IF se evalúa como verdadera.

TO (shift; tecla [4]). Introduce el valor máximo de la variable de control en un bucle FOR. Por ejemplo:

```
FOR I 1 TO 10
```

TO también se utiliza para "rebanar" (slicing) cadenas. La siguiente declaración LET asigna valores desde la posición 2ª a la 4ª de la cadena S$:

```
LET S$(2 TO 4) = "ABC"
```

UNPLOT (palabra clave; tecla [W]). Borra un pixel de la pantalla en una dirección especificada. La forma general del mandato UNPLOT es:

UNPLOT *h,v*

donde *h* es la dirección (o emplazamiento) horizontal (de 0 a 63) y *v* es la dirección vertical (de 0 a 43). La dirección 0,0 de UNPLOT está situada en el rincón inferior izquierdo de la pantalla.

USR (función; tecla [L]). Llama a una rutina de lenguaje de máquina situada en una dirección de memoria especificada. La forma general de USR es:

USR *a*

donde *a* es la dirección en la memoria del comienzo de la rutina.

VAL (función; tecla [J]). Proporciona el valor numérico de una cadena. Los caracteres de la cadena pueden consistir en dígitos, funciones, cálculos o variables que forman una expresión numérica válida.

APENDICE B
CODIGOS DE ERROR DE T/S 1000

Al final de cada pase de programa, aparece un mensaje en el ángulo inferior izquierdo de la pantalla bajo la forma de:

m/n

donde *n* es el número de línea en el que se para la ejecución del programa, y *m* es uno de los siguientes códigos de error:

0 Sin ningún error; programa completado.
1 Discrepancia entre variables de control nombradas en declaraciones FOR y NEXT.
2 Uso de un nombre de variable indefinida
3 El índice de ordenación (array) está fuera del campo especificado en una declaración DIM.
4 El ordenador está sin memoria.
5 La pantalla del televisor se ha quedado sin espacio.
6 Un cálculo ha producido un número demasiado grande para que lo maneje el ordenador.
7 La declaración RETURN no está precedida por GOSUB.
8 Se usa INPUT como mandato inmediato.
9 El programa ha terminado en una declaración STOP.
A Empleo ilícito de una función (por ejemplo: SQR—1 ó LN 0).
B Dirección, número de código u otros enteros ilícitos en declaraciones tales como PRINT AT, PLOT, CHR$, etc.

C La expresión en cadena después de VAL no se puede evaluar y expresar con un número.
D Las teclas BREAK o STOP del teclado han interrumpido el programa.
E No usarla.
F Mandato de SAVE sin nombre de programa.

CURSO BASICO DE TECNICAS DE PROGRAMACION
M. D. Tagarro

UN CURSO BASICO DE PROGRAMACION EN CUALQUIER ORDENADOR.

Tras el análisis de una serie de problemas que podríamos considerar los más frecuentes, y con el conocimiento muy elemental del lenguaje de programación, tendremos la capacidad de adaptarnos en calidad de PROGRAMADOR, a cualquier instalación informática.

El ordenador es una máquina capaz de tener almacenado en su memoria, un algoritmo de resolución a un cierto tipo de problema (programa), captar los datos del problema mediante un medio de entrada, obtener la resolución del mismo en forma automática, y dar los resultados en un medio de salida, también automático.

Al ordenador se le proporciona la información a través de un PROGRAMA.

La eficacia o ineficacia del ordenador se encuentra en nuestra capacidad de utilizarlo correctamente, sabiendo plantear una solución al problema a resolver. Para ello, la solución que propongamos deberá responder a unos planteamientos lógicos, a una concreta metodología y pasarla a un lenguaje inteligible a la máquina (ordenador).

COMO PROGRAMAR UNA CONTABILIDAD BASICA.

TODO SOBRE SU TIMEX SINCLAIR 1000 Y SU ZX81

El presente libro le lleva a usted de la mano desde el comienzo y le explica en un lenguaje sencillo, cotidiano, la manera de utilizar los ordenadores TIMEX SINCLAIR 1000 y el ZX81 hasta el límite de sus posibilidades.

Usted aprenderá rápidamente a...

— conectar su televisor y su cassette al ordenador y hacerlos trabajar juntos
— utilizar el teclado para darle órdenes al ordenador
— escribir sus propios programas para gráficos, cálculos, juegos y otras cosas
— utilizar el fácil y expresivo lenguaje BASIC de programación
— emplear los programas, tan fáciles de pasar, incluidos en este libro para:
  — realizar toda clase de juegos didácticos, intructivos y de diversión.
  — convertir su ordenador en una calculadora imponderable
  — hacer diagramas de barras para ayudarse a calcular las finanzas domésticas
  — dibujar figuras en la pantalla de su televisor

El presente libro le ayudará a sacar el mejor partido de su ordenador.

cubierta OLIVE MILA
ESTUDIO DISEÑO GRAFICO