Enciclopedia práctica de

INFORMATICA

1

ALGAR

Enciclopedia práctica de

INFORMATICA

Enciclopedia práctica de

INFORMATICA

VOLUMEN 1

ALGAR

 ISBN (Obra completa): 84-7685-012-3 - ISBN (Tomo I): 84-7685-013-1 - Redactores y colaboradores: G. Jefferson, R. Ford, S. Tarditti, F. Martín, F. Blasco - Para la edición inglesa: R. Pawson (editor), D. Tebbut (consultant editor), C. Cooper (executive editor), D. Whelan (art editor), Bunch Partworks Ltd. (proyecto y realización) - Realización gráfica: Luis F. Balaguer - Fotocomposición: Tecfa, S. A. (Barcelona) - Impresión y encuadernación: Gráficas Ramón Sopena, S. A. - Provenza, 95 - 08029 Barcelona - Depósito Legal: B. 16.085-1990 - Impreso en España - Printed in Spain.

ÍNDICE

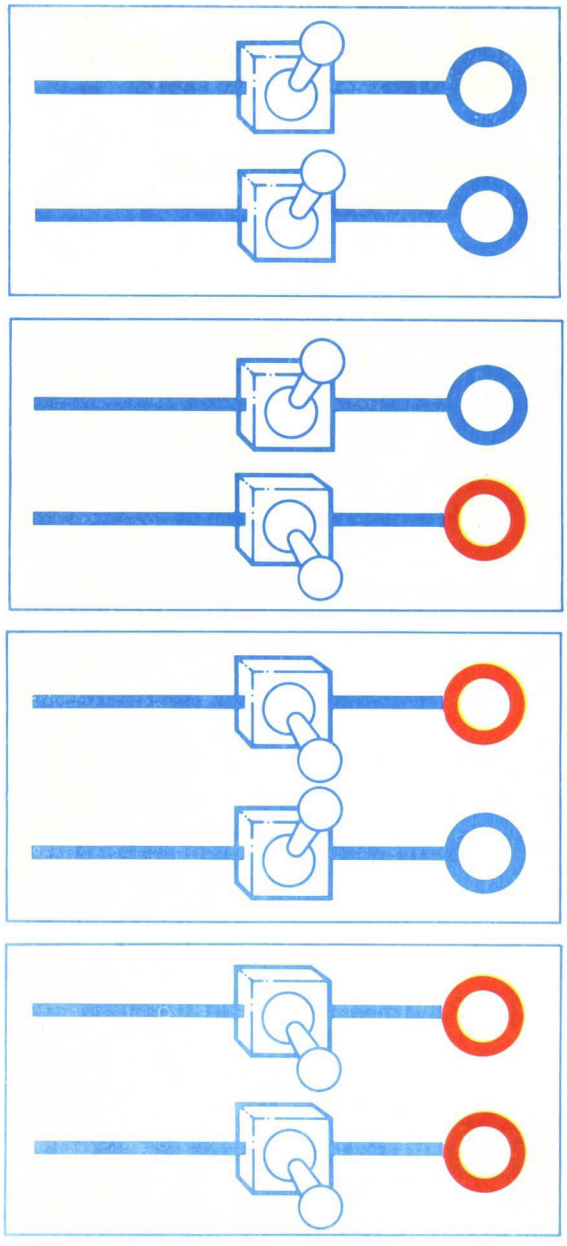
¿Qué es un ordenador?

¿Cómo "piensan" los ordenadores y cuánto "saben"? Las respuestas son esenciales para llegar a entenderlos

La pregunta "¿Qué es un ordenador?" no puede responderse con la misma sencillez que, por ejemplo, "¿Qué es un televisor?" o "¿Qué es una lavadora?". La razón de esta dificultad es que, a diferencia de estos dos electrodomésticos, el ordenador no posee una única función exclusiva. Los ordenadores digitales son una nueva generación de máquinas capaces de ejecutar una serie casi infinita de funciones según el programa que su propietario les suministre.

La idea de programabilidad, es decir, la capacidad de programación, no resulta del todo extraña en el hogar moderno; muchos electrodomésticos, como las lavadoras automáticas, llevan incorporados diversos programas que permiten utilizarlos de maneras diferentes. Sin embargo, en el caso de los ordenadores, es toda la función de la máquina la que, en cuestión de minutos, puede modificarse mediante la introducción de un nuevo programa: el ordenador se convierte en procesador de textos, en máquina para juegos o en un contable que cuida del estado de su cuenta bancaria.

¿Cómo realiza el ordenador tantos trabajos diferentes? Lo sabremos en profundidad a medida que avance el curso, pero analicemos ahora someramente algunos de los principios que entran en juego.



Conmutando en números
Para representar los números, los ordenadores utilizan circuitos eléctricos. Los circuitos consisten básicamente en interruptores. Cada interruptor puede estar en una de dos posiciones: encendido o apagado ("on" u "off"). Con dos interruptores se consiguen cuatro combinaciones de "on" y "off". Los ordenadores emplean un sistema similar a éste para representar los números. "Off/off" es cero, "off/on" es uno, "on/off" es dos y "on/on" es tres. La combinación de grupos compuestos por más de dos elementos permite la representación de mayor cantidad de números. Los ordenadores pueden procesar cifras muy elevadas y realizar complicadas operaciones matemáticas con gran rapidez, valiéndose de miles de interruptores microscópicos

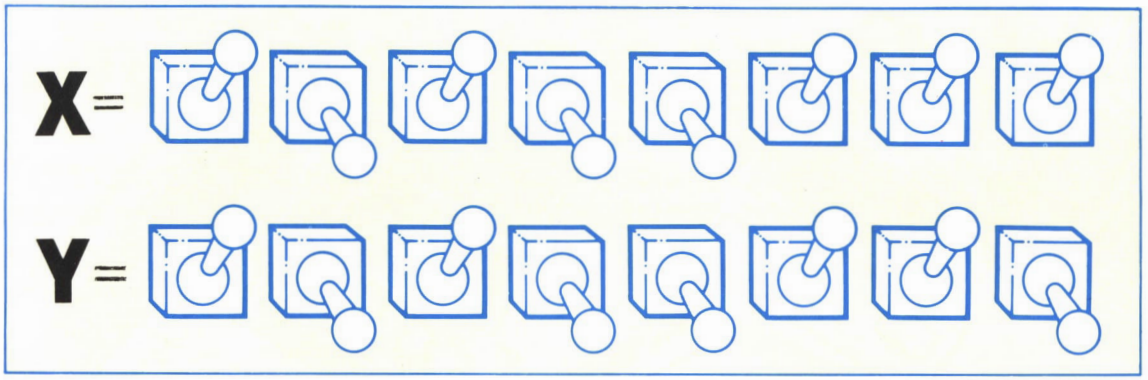
En cierto sentido, un ordenador no es más que una caja repleta de diminutos interruptores eléctricos que pueden conectarse entre sí de distintas maneras. Ésta no es, sin embargo, la mejor forma de comenzar si se desea comprender lo que un ordenador puede *hacer*; a este nivel no somos nosotros los interesados en comprender los ordenadores, sino quienes los diseñan y construyen. El ordenador moderno es una máquina extraordinariamente compleja; gracias al sorprendente desarrollo en el campo de la microelectrónica (el famoso *chip* de silicio, o circuito integrado de silicio), incluso hasta el más pequeño de los ordenadores personales puede contener 250 000 de estos pequeños interruptores. Todos estos interruptores pueden estar "encendidos" o "apagados". Cualquier apostador de quinielas sabe que el número de combinaciones posibles entre interruptores "encendidos" y "apagados" es enorme. Por otra parte, el ordenador que usted adquiera llevará un programa incorporado con carácter permanente que será como un disfraz para esta complejidad que resulta asombrosa para la mente de cualquier persona, y que permitirá que usted "hable" con la máquina mediante algunos vocablos abreviados pero fácilmente reconocibles.

Al utilizar un ordenador por primera vez, mucha gente se sorprende porque, una vez encendido, descubren que la máquina no sabe nada que les resulte útil. Por extraño que parezca, muchas personas siguen pensando, aún hoy, que el ordenador es un "cerebro electrónico" del cual se espera que lo sepa todo. ¿Es necesario que el ordenador sepa cuál es la capital de Afganistán? ¿O cuál es la altura del Kilimanjaro? En realidad, lejos de saber todas estas cosas, el chip de silicio, que conforma el "cerebro" de un ordenador, no sabe ni siquiera el alfabeto y menos aún algo de aritmética. Lo único que el ordenador sabe es varios centenares de combinaciones numéricas, y todo lo que se le quiera enseñar ha de ser indefectiblemente traducido a esos números. Los pequeños interruptores que hemos mencionado pueden recordar números; cada patrón de interruptores ON y OFF ("encendido" y "apagado", respectivamente) representa un número de acuerdo al sistema numérico binario, que sólo utiliza las cifras "0" y "1". El hecho de que el ordenador pueda recordar o, mejor dicho, *almacenar* información, es de vital importancia para la forma en que éste trabaja; la memoria electrónica de un Sinclair Spectrum retiene una información equivalente al contenido en palabras de seis páginas de este fascículo.

Además de almacenar números en su memoria, el ordenador puede hacer cosas *con* esos números: puede sumarlos y restarlos, compararlos entre sí y trasladarlos dentro de su memoria. Todo lo que la máquina puede hacer se deriva de estas sencillas operaciones. Supongamos que deseamos almacenar textos en el ordenador. Inventemos un código, de manera que a cada letra del alfabeto le corresponda un número determinado: el ordenador podrá entonces almacenar palabras en forma de números y combinarlas entre sí. ¿Que deseamos jugar a la rana? Dibujemos la silueta de una rana y tracemos sobre ella una cuadrícula, de modo que a cada pequeño cuadrado resultante pueda asignársele un número... Por supuesto, no es necesario que usted mismo se invente los códigos, pues este trabajo ya ha sido realizado por los fabricantes y los diseñadores, que lo han estructurado en forma de programas para ordenadores.

¿Qué es un programa? Es una lista de instrucciones que se le proporciona al ordenador para que realice operaciones simples (sumar, comparar, etc.) en un orden especial. Pero, ¿en qué consisten estas instrucciones y cómo llegan hasta el ordenador? ¡En realidad, se trata tan sólo de otros números que también están almacenados en la memoria del ordenador! Esto parece colocarnos ante una paradoja similar a la del huevo y la gallina. El ordenador no puede hacer nada si no cuenta con un programa que le diga lo que debe hacer; cada vez que usted pulsa la letra "A" en el teclado, un programa contenido en el ordenador debe explorar el teclado, descubrir cuál es la tecla que ha pulsado y luego decirle al ordenador cuál es el código numérico que le corresponde a esa letra. Pero en un principio, cuando se diseñaron los primeros ordenadores, este programa de exploración del teclado no existía; alguien debía colocar los números correctos directamente en la memoria del teclado, mediante instrumentos especiales, sólo para conseguir que ésta comprendiera las letras mecanografiadas y que estas letras quedaran reflejadas en una pantalla de televisión.

Pero una vez realizados estos primeros programas básicos, todo resulta mucho más sencillo. Ahora usted



Un código para letras y números

Un grupo de ocho interruptores permite 256 combinaciones distintas de "on" y "off". Esta cantidad resulta más que suficiente para un código individual (que no emplee más que unos y ceros) para cada una de las letras, números y signos especiales que se incluyen en el teclado de un ordenador similar al de una máquina de escribir. La ilustración muestra cómo están representadas las letras X e Y dentro de un ordenador que utiliza el código ASCII

puede introducir nuevos números en la memoria del ordenador mecanografiándolos en el teclado. Este proceso se denomina programación en código de lenguaje máquina, y más adelante nos ocuparemos de él con mayor amplitud. Como la programación en código de lenguaje máquina es bastante difícil y aburrida, algunos programadores ingeniosos han elaborado programas (en código de lenguaje máquina) que traducen los vocablos ingleses como PRINT, BEEP, LOAD y LIST (impresión, emisión de señal acústica, carga y listado de un programa) a instrucciones en código de lenguaje máquina que el ordenador puede utilizar. Todos los ordenadores personales, a excepción de los más sofisticados, llevan incorporado este tipo de programa, gracias a lo cual uno puede programarlos mediante un lenguaje sencillo para ordenadores en lugar de hacerlo mediante series de números. Este lenguaje se denomina BASIC. Recuerde que cada vez que utilice el BASIC lo que está trabajando para usted allí, en el interior del ordenador, es el producto de horas y horas de trabajo por parte de los programadores.

Con lenguajes de ordenador como el BASIC resulta sencillo escribir programas útiles o tan sólo de entretenimiento, mientras uno permanece ajeno a la frenética y compleja actividad que se ha suscitado en el interior del ordenador sólo para detectar que se ha tecleado la letra A. Por ejemplo, es muy fácil escribir un programa que almacene los nombres de las capitales de todos los países del mundo para que, al ser requerido el ordenador con la pregunta "¿Cuál es la capital de Afganistán?", proporcione como respuesta "Kabul". En otras palabras, el cerebro electrónico sólo puede saber aquello que se le ha comunicado con anterioridad: no puede descubrir cosas por sí mismo.

Si las cosas son así, ¿por qué, entonces, los ordenadores son tan útiles? Porque pueden almacenar enormes cantidades de información y manipularla mucho mejor que las personas. Y, por supuesto, no siempre tiene que ser *usted* quien coloque en él la información por primera vez. Puede comprar un programa, escrito por otra persona, que tenga almacenados los nombres de todas las capitales; en este caso el ordenador actuará como libro de referencia electrónico. Otra posibilidad es comprar un programa que trabaje con la información que usted le proporcione a través del teclado: un programa de "tratamiento de textos" con el que podrá mecanografiar, corregir y hacer nuevas copias de, por ejemplo, sus cartas y documentos, o bien un programa de "base de datos" con el cual podrá confeccionar el catálogo de una biblioteca y que en unos segundos le responderá a preguntas como "¿Qué libros de George Bernard Shaw poseo que hayan sido publicados antes de 1926?"

El hecho de que esa pobre máquina sorda que es el ordenador comprenda sólo números, en la práctica no es un signo de debilidad, sino más bien de fortaleza. Si los ordenadores trataran realmente con los objetos que nos interesan, pongamos por caso con palabras o con colores, serían muchísimo más complicados y se necesitaría una clase diferente de ordenador para cada tipo de trabajo que se realizara. De todas maneras, ¿cómo haría usted para almacenar VERDE en la memoria de un ordenador? Una vez se ha captado el principio de que el ordenador no necesita "comprender" aquello con lo que está tratando en la forma que una persona precisa comprenderlo, se entenderá cómo es que un solo tipo de ordenador puede ocuparse prácticamente de todo. Lo único que se requiere es que un programador sea capaz de describir el problema de

¿POR QUÉ SOFTWARE?

El micro es un maestro nato

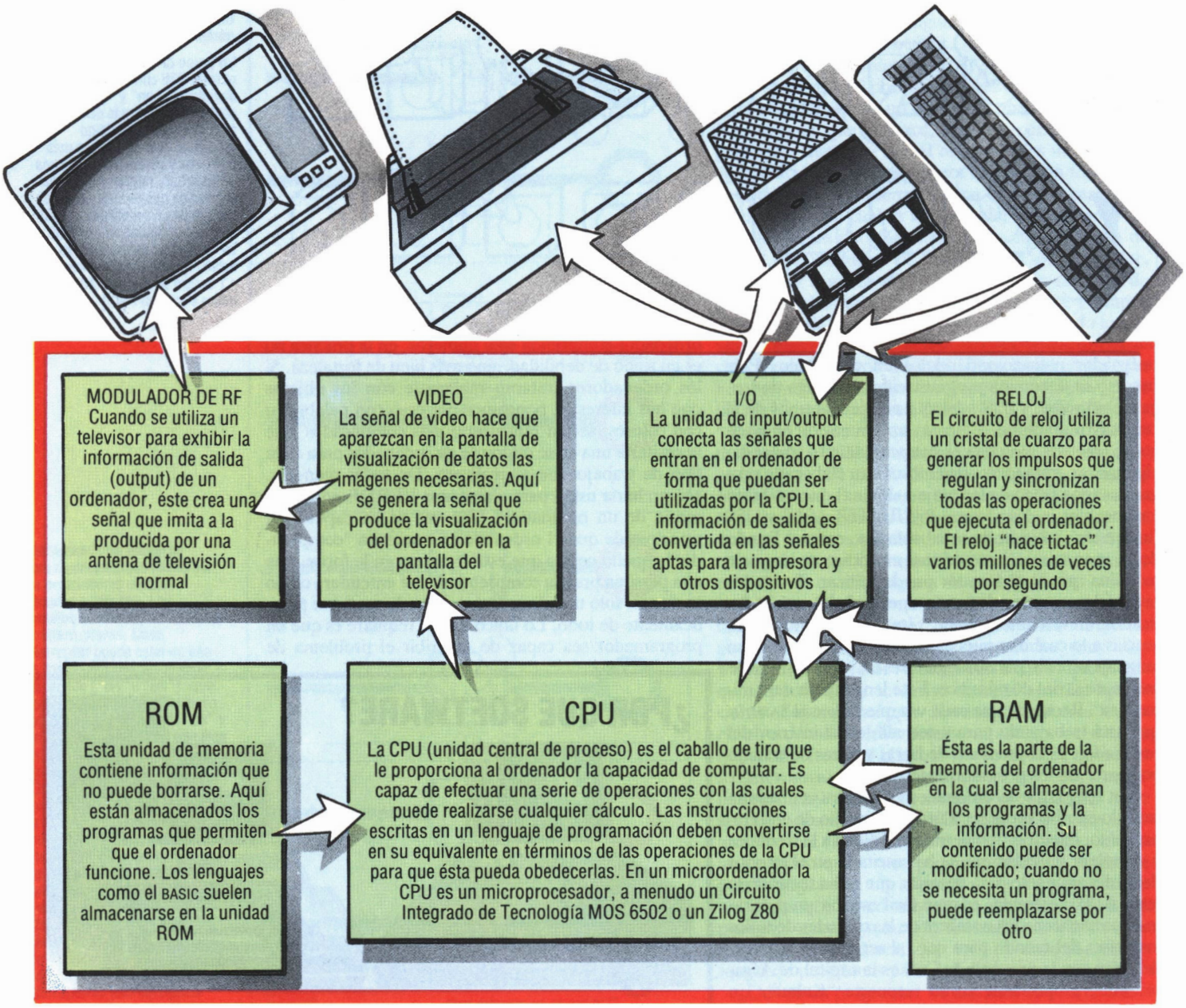
Útil para cartas y estados de cuentas

Fue inventado con fines científicos

Y también para jugar y divertirse

Un ordenador es una máquina versátil y puede desempeñar diversas funciones. El software centraliza su poder. Una misma máquina puede ser utilizada por un hombre de negocios con un software de gestión, por un tecnólogo con un software estadístico, o con fines de entretenimiento si se carga con un software de juegos. El software es lo que determina la actividad del ordenador

Tony Lodge



Lo que sucede en el interior

Para instalar un sistema completo de ordenador y dejarlo a punto para ser utilizado, es necesario conectar entre sí diversas unidades. Los chips de silicio, responsables de la existencia del ordenador personal, están alojados en el interior de la caja, por lo general debajo del teclado. Quite la tapa y se encontrará con estos componentes principales

forma que éste pueda reducirse a una serie de números. Por ejemplo, si deseamos que un ordenador produzca música, no esperaremos hacer flotar en su interior los sonidos reales; en cambio, será necesario describir cada nota de la escala musical mediante un número que sea proporcional a su altura o su frecuencia. Podemos hacer que el ordenador envíe las señales eléctricas que utiliza para representar los números a un amplificador en lugar de a una pantalla de televisión, con lo cual se podrá oír los resultados. ¿Cómo hacer que un disparo de misil atraviese la pantalla dirigiéndose hacia los "invasores del espacio"? Simplemente, desplazando algunos números, que representan la forma de un misil, de un lugar a otro en la parte de la memoria del ordenador que actúa como "mapa" de la pantalla del televisor. Tanto a las imágenes como al movimiento, al color y al sonido puede asignárseles un código numérico adecuado que permita que el ordenador los manipule y que, mediante un "transmisor", como un televisor o un amplificador, lo vuelva a convertir en señales que tengan algún significado.

Intentaremos, pues, responder a la pregunta que nos formulamos al comienzo: "¿Qué es un ordenador?" Es una máquina que almacena señales electrónicas que representan números. Algunos de estos números son instrucciones que le dicen al ordenador lo que debe hacer con los otros números. El ordenador seguirá estas instrucciones con total exactitud, sin cansarse y sin equivocarse (aunque reproducirá fielmente nuestros propios errores de programación), a un promedio de muchos miles de operaciones por segundo. El resultado final de estas infatigables manipulaciones no es otro que más números. Éstos son "traducidos" en la información que deseamos, de forma que nos resulten comprensibles. Es la actividad de los programadores humanos la que determina la utilidad del ordenador, explotando su destreza con los números y haciéndole desempeñar tareas que tengan algún significado para nosotros, partiendo de diversas formas de información y transformándolas según un proceso que, sin el ordenador, resultaría sumamente largo, tedioso y complicado.

Un sistema completo

Tanto para comunicarse en dos direcciones con su ordenador como para almacenar programas e incluso para algunos juegos, usted necesitará un hardware extra

Impresora

La impresora es imprescindible cuando se necesitan copias en papel de los programas o resultados impresos. Existen varias clases de impresoras, cuyo precio varía según la rapidez del proceso de impresión y la calidad de las copias

Unidades de disco

Al igual que las cassettes, las unidades o dispositivos de disco almacenan programas. En lugar de una cassette, se utiliza un "disco flexible". Las unidades de disco son mucho más caras que las grabadoras, pero trabajan mucho más rápido y almacenan mayor cantidad de información

Grabadora

Una grabadora de cassettes normal constituye una forma barata de guardar programas. El programa queda almacenado en la memoria del ordenador mientras éste lo está utilizando. Al cortarse el suministro energético, el contenido de la memoria desaparece. Antes de que esto ocurra, el programa puede grabarse en cinta magnetofónica; cuando se lo vuelva a necesitar, bastará con reproducírselo al ordenador

Televisión

Un aparato de televisión normal permite visualizar los mensajes del ordenador. Y, cuando usted escribe un programa, todo lo que vaya tecleando aparecerá reflejado en la pantalla. El monitor situado detrás del televisor está diseñado para proporcionar imágenes de mayor definición y calidad

Mando de palanca

De aspecto similar a los mandos de que disponen algunos juegos recreativos, su utilización depende del tipo de juego para el cual se emplee. Por ejemplo, puede controlar tanto una nave espacial como los caracteres de un laberinto. Algunos mandos de palanca poseen un "relleno" de diez o más botones (dispuestos como los de una calculadora), cuya utilización depende, asimismo, del juego de que se trate

Mando de bola

Este mando también se utiliza para los juegos. Haciendo rotar la bola se mueven por la pantalla los elementos del juego. Permite un ajuste de posición más preciso y más rápido que el mando de palanca y resulta más cómodo de utilizar. Está dotado de botones para tareas como disparar "láseres", etc.

El ordenador

El ordenador es el corazón de todo el sistema, aunque para poder comunicarse con el usuario requiere algunos "extras". Posee un teclado similar al de una máquina de escribir pero con algunas teclas adicionales. Cuenta con algunas clavijas bipolares (situadas, por lo general, en la parte posterior del ordenador) para conectarlo con otras máquinas, como con la grabadora o la unidad de disco, con el televisor, etc.

Mark Watkinson

La revolución mundial

La revolución de los ordenadores se extiende por toda la Tierra y está cambiando los hábitos de la sociedad. Es el mundo del mañana, pero se vive hoy

Ordenador
Unidad de salida
Unidad de entrada
Unidad de entrada o salida
Interface

La revolución de los ordenadores coincidió con la carrera espacial y con el viaje que llevó al primer hombre a la Luna. Los millones de dólares que se invirtieron en este esfuerzo tuvieron como consecuencia directa la concentración de las mejores mentes científicas y la apertura de una brecha en cuanto a técnicas de fabricación. El viaje a la Luna hizo posible algo casi imposible. Los efectos inmediatos abarcan desde la creación de nuevos materiales de cerámica, plásticos y adhesivos hasta la microminiaturización de un increíble potencial informático.

Cuando un "ordenador" se identificaba con bastidores llenos de circuitos, también implicaba miles de componentes electrónicos unidos entre sí mediante cables individuales. Los costos de fabricación eran enormes. Ahora esos mismos circuitos pueden introducirse en chips de silicio lo suficientemente pequeños, como para caber en una carcasa del tamaño de un teclado.

Los chips albergados en el interior de ordenadores como el Dragon o el Spectrum no sólo son pequeños, sino que pueden fabricarse en serie a un costo casi insignificante. El silicio, la materia prima a partir de la cual se producen los chips, es uno de los elementos más comunes de la Tierra. Los granos de arena están compuestos en su mayor parte por silicio.

Ahora que el ordenador ha conquistado un sitio en los hogares y en las oficinas de todo el mundo desarrollado, tenemos la oportunidad de constituirnos en privilegiados testigos del comienzo de la segunda revolución industrial. La primera revolución industrial sustituyó la mano de obra del trabajador por una maquinaria motorizada. La revolución del ordenador supondrá un ahorro del tiempo de trabajo del personal especializado y reemplazará a los obreros de las fábricas por robots controlados por ordenador.

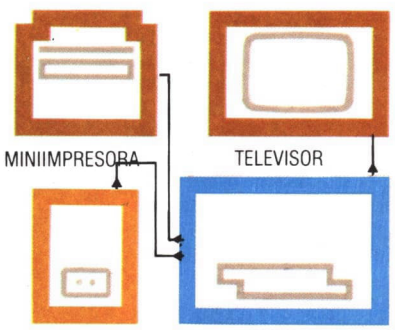
La incidencia a largo plazo de esta revolución en nuestras vidas no está ni mucho menos esclarecida. Lo único que puede suponerse con toda certeza es que los patrones de trabajo y ocio sufrirán modificaciones y que éstas se producirán con rapidez. Los robots, extensiones mecánicas de los ordenadores, están reemplazando a los obreros. Éstos se enfrentan a la disyuntiva de especializarse en algo o quedarse sin trabajo.

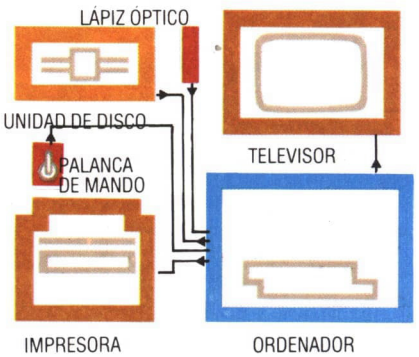
Actividades tan tradicionales como las de tipógrafo y cajista, e incluso la de maestro, podrían ser sustituidas con la actual tecnología. Y dentro de poco acudir a la consulta del médico de cabecera podrá consistir en una entrevista con un terminal de ordenador.

La consecuencia social de la primera revolución industrial fue el desplazamiento de millones de personas desde las zonas rurales y la aparición de la polución industrial (y los beneficios materiales) que caracterizan al mundo occidental. Nos hallamos al borde de una revolución tan dramática como aquélla, en la que el motor de la sociedad será el ordenador. Una revolución en la que la clave para la supervivencia estará en comprender al ordenador y saber utilizarlo.

Personal

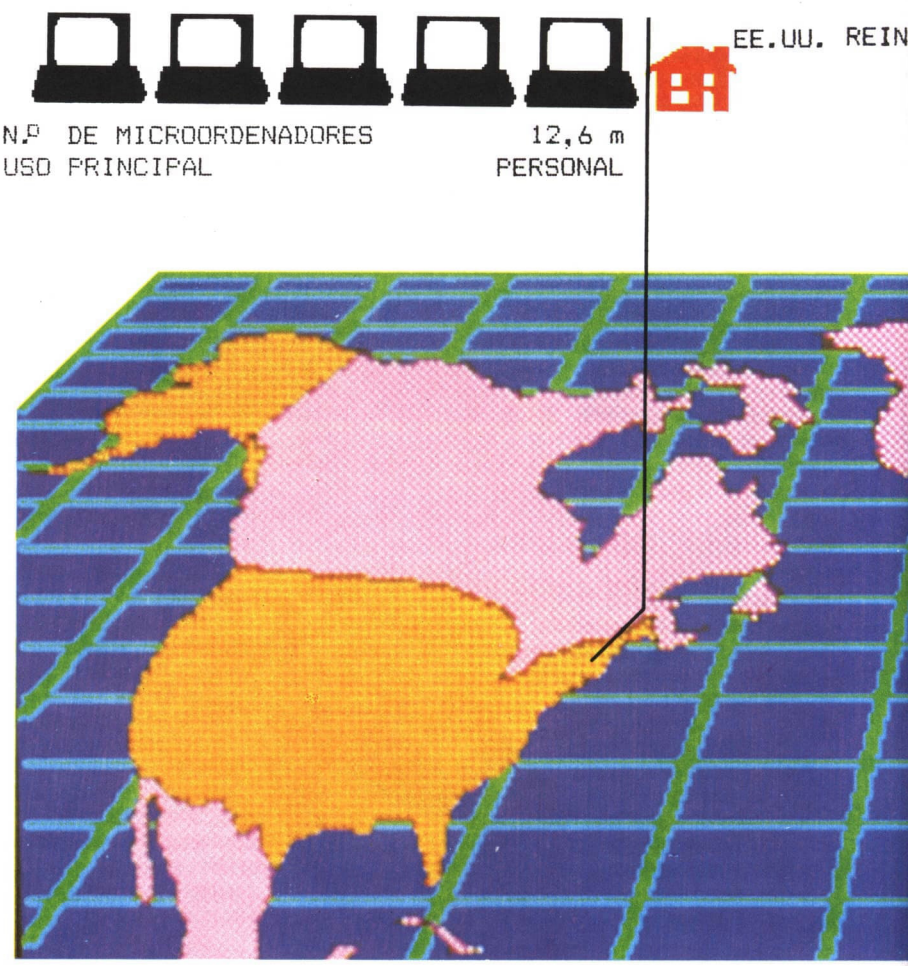
Un sistema personal económico puede montarse por poco dinero. Es su pasaporte para el futuro

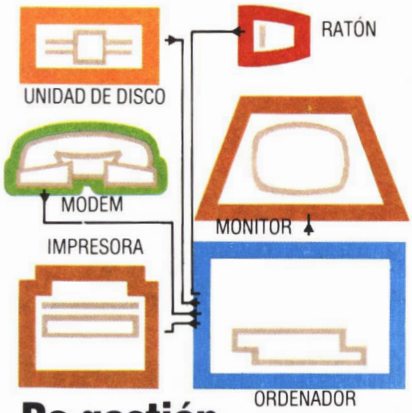




Juegos

Bastan unos pocos accesorios para acceder a su mundo. Los juegos no sólo entretienen, sino que enseñan los principios básicos de la informática



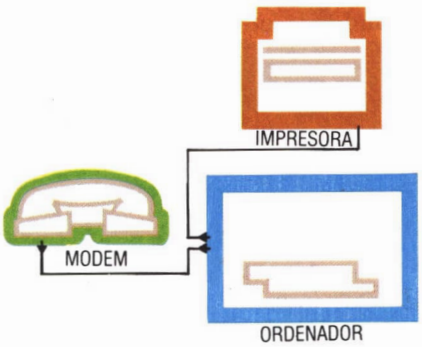


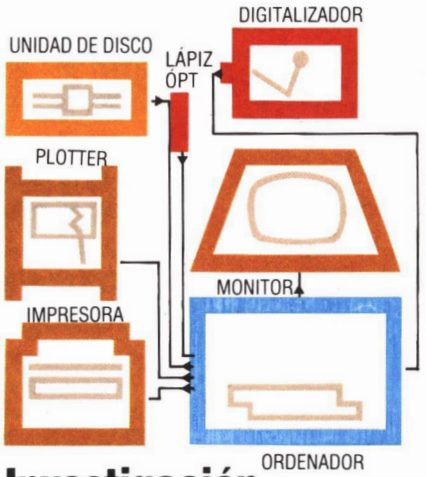
De gestión

Los pequeños sistemas de microordenador están ayudando (y reemplazando) a los empleados administrativos y a los hombres de negocios. Los modems hacen posible el acceso, a través de un teléfono convencional, a ordenadores grandes y a la imprescindible información de sus enormes bancos de datos. Los equipos de mecanógrafos se van convirtiendo en algo ya anticuado, a medida que el tratamiento de textos va ganando terreno. El software contable y por "hojas electrónicas" reduce los costos de la contabilidad y permite que las decisiones financieras se adopten en cuestión de minutos. Las unidades de disco sustituyen salas enteras repletas de archivadores, y la impresora se encarga de todo, desde imprimir cartas perfectas hasta proporcionar una tabulación instantánea de los coeficientes del día en la bolsa de valores. Y el empresario que no sepa digitar será auxiliado por el «ratón», dispositivo similar al mando de bola que permite introducir las órdenes en la máquina

Comunicación

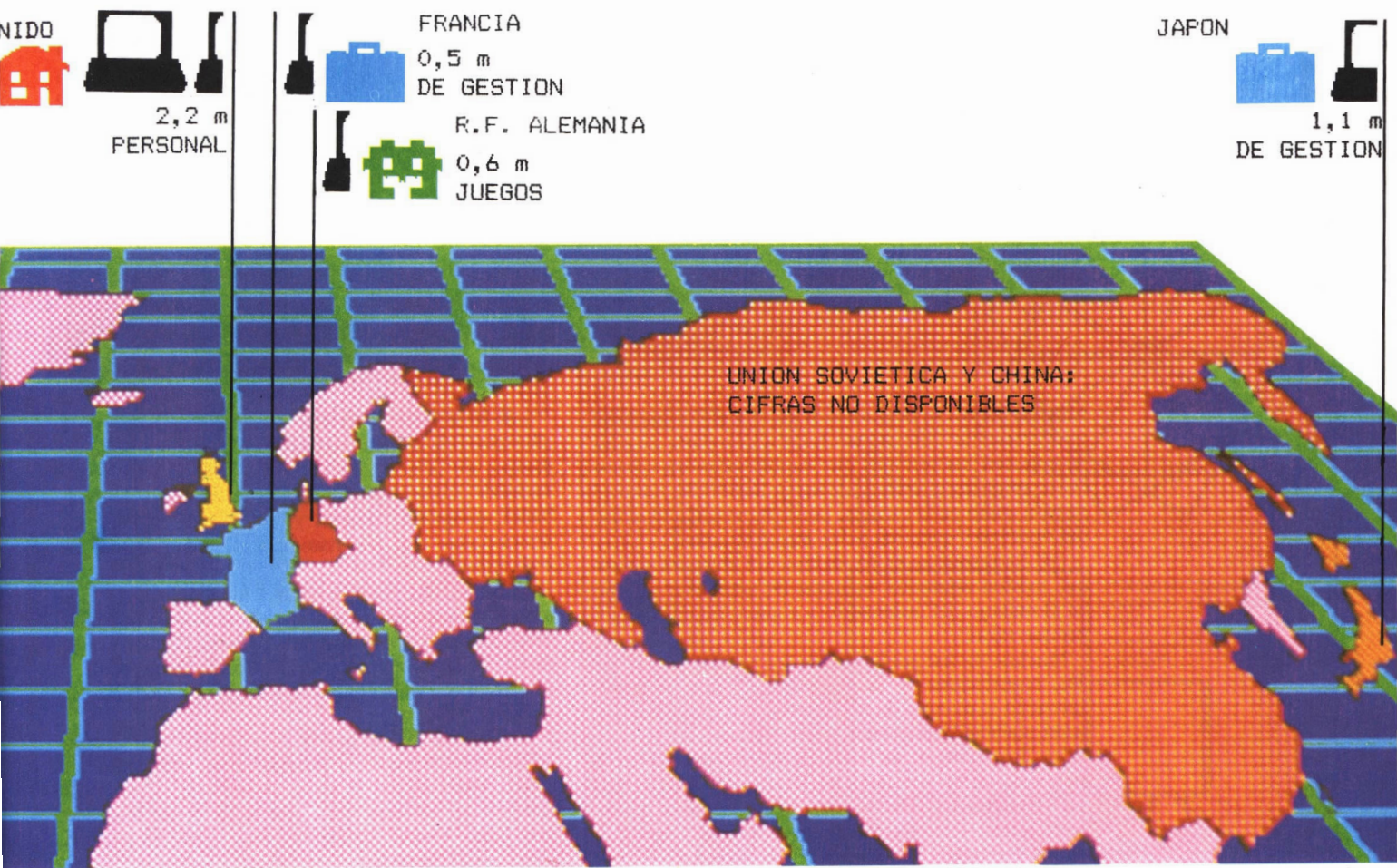
La red de teléfonos es internacional y ahora muchos ordenadores son tan pequeños que pueden sostenerse en la mano o en el regazo. Un atareado ejecutivo puede digitar sus informes a bordo de un Boeing 747 que lo lleva a Nueva York y, a su llegada, enviarlos a casa a la velocidad del electrón. El modem comunica el ordenador personal con el teléfono y permite en teoría un acceso instantáneo a todos los ordenadores del mundo, incluyendo a los ordenadores industriales gigantes. Los bancos de datos están a mano las 24 horas del día y hasta un periodista, que esté consultando el archivo desde Afganistán o Sudamérica, puede redactar su artículo en el tiempo que tarde en hallar una cabina telefónica





Investigación

El potencial de un ordenador avanzado ya ha dejado de ser patrimonio exclusivo del laboratorio de una universidad o de una multinacional. En las escuelas y en los hogares de todo el mundo se están introduciendo sofisticados sistemas. Incluso es posible investigar por CAD (Computer Aided Design, diseño auxiliado por ordenador) y AI (Artificial Intelligence, inteligencia artificial) con un sistema personal. También son asequibles los "plotters" (dispositivos trazadores de gráficos), para realizar dibujos técnicos, y los digitalizadores, que permiten dar entrada a imágenes complejas, como pueden ser mapas o diagramas. En la actualidad, los conocimientos en materia de ordenadores son tan importantes para un currículum de estudios como lo fuera el latín hasta hace algunos años. Los niños del presente están aprendiendo a enfrentarse con el mundo informatizado del mañana



El educador electrónico

Incluso a los más pequeños de la familia les encantará usar el ordenador. Aquí le explicamos cuál es la mejor forma de iniciarlos

Ian McKinnell

Uno de los instrumentos por ordenador más eficaces para niños a nivel de enseñanza primaria es la "tortuga". El robot está conectado a un microprocesador y funciona según un programa denominado LOGO. Los niños pueden dibujar con la tortuga, que resulta muy útil para enseñar conceptos matemáticos como forma, distancia y la relación entre los objetos. ¡Y además es sumamente divertida!

Muchos padres se preguntan si un ordenador personal podría resultar útil para sus hijos. Gran parte de ellos ya conocen las ventajas que supone para un adolescente aprender el manejo de los ordenadores, tanto en su casa como en la escuela; pero ¿tiene utilidad el uso del ordenador para los niños más pequeños?

¡Sí! La respuesta es decididamente afirmativa, aunque existen diferentes maneras de introducir al niño en el manejo del ordenador, y algunas de ellas mejores que otras.

Muchos de los gobiernos de los países desarrollados, convencidos de la necesidad de que los niños aprendan el manejo de los ordenadores en la escuela, están realizando importantes inversiones para dotar a todos los centros de enseñanza primaria cuando menos de un microordenador. Ahora son los propios maestros quienes deben decidir cuál es la mejor manera de aprovechar este "pequeño potencial" para que resulte útil en sus programas de enseñanza.

Los ordenadores no sólo resultan idóneos para la enseñanza de las matemáticas; con un buen programa, aunque en la actualidad son muy escasos, los ordenadores pueden ayudar al niño a aprender música, ballet, geografía, idiomas y, por supuesto, materias relacionadas con las matemáticas, como aritmética y geometría.

El ordenador puede ayudar a los pequeños de dos formas principales. El niño puede utilizar la máquina para explorar su propio mundo, o bien el ordenador es capaz de actuar como un maestro, instruyendo y adiestrando al niño en una amplia gama de temas educativos.

No constituye una buena idea intentar que su hijo de seis años aprenda a programar un ordenador en BASIC. Antes de los 12 años, un niño no es capaz de comprender los conceptos abstractos de un lenguaje semejante. Pese a que algunos niños están en disposición de escribir programas en BASIC a los nueve años o

incluso antes, los trabajos del psicólogo francés Jean Piaget nos muestran que antes de los 12 o 13 años la mayor parte de los niños tienen dificultades para captar ideas abstractas.

En consideración a este problema, los investigadores han descubierto un sistema para que el niño controle y programe un ordenador sin necesidad de manejar esos conceptos abstractos (véase recuadro relativo al LOGO). La forma habitual en que el maestro introduce al niño en el mundo del ordenador constituye una mezcla de ambos métodos.

El aprendizaje jugando con la tortuga

Incluso los niños más pequeños pueden utilizar el ordenador para aprender. En la ilustración de la página contigua se ve a un niño jugando con la "tortuga", un robot mecánico que está conectado a un microordenador. Estas tortugas resultan caras y están ideadas para su utilización en la escuela, pero su fundamento es muy simple: el ingenio tiene dos ruedas y está provisto de un lápiz. El niño le comunica a la tortuga que avance a través de una hoja de papel y, a medida que se va desplazando, le indica si debe o no trazar una línea (la "huella" de la tortuga). De esta manera el niño dibuja, instruyendo al robot en cómo debe formar esquinas y unir líneas. Como los niños son estimulados para determinar exactamente qué movimientos debe hacer la tortuga al objeto de que resulte un dibujo con formas específicas, llegan a descubrir por sí mismos los elementos que conforman la geometría básica. Este planteamiento de "autoayuda" constituye la base del método LOGO. La teoría consiste en que las lecciones aprendidas "heurísticamente" (mediante pruebas y errores) se comprenden mucho mejor que cuando se muestran ejemplos.

Estas dos escuelas de pensamiento nos señalan dos formas diferentes de utilizar los ordenadores con los niños. De acuerdo con el LOGO, los niños de nueve o diez años comienzan con el manejo de una versión de la tortuga en la pantalla del ordenador, dibujando formas complicadas y enseñando al robot cómo recordar diversos procedimientos. Cuando un niño "instruye" a la tortuga para que realice líneas o figuras, ya sea sobre una hoja de papel o en la pantalla, no hace otra cosa que programar la máquina. El LOGO es un lenguaje mediante el cual el niño programa antes de que haya desarrollado la capacidad de comprensión de los conceptos abstractos necesarios para muchos lenguajes de ordenador. Por lo tanto, a través del "juego con la tortuga" los niños más pequeños se familiarizan con la idea del manejo del ordenador y pueden explorar su propio entorno.

El otro planteamiento se vale de la evidente "paciencia" del ordenador para enseñar al niño mediante ejemplos.

A los niños que tienen dificultad para comprender un tema o una idea se les suele ayudar con programas de "procedimiento correcto y ejercicios", los cuales formulan preguntas al niño y luego, en un marcador, le indican cuántas ha acertado. Muchos de estos programas son enormemente atractivos, ya que están provistos de excelentes gráficos a color e interesantes melodías o efectos sonoros. Estos cuestionarios estimulan al niño a aprender, y el ordenador no se cansa ni abandona mientras el niño insista con una respuesta equivocada. Esta paciencia del ordenador se ha mostrado muy valiosa para enseñar a los alumnos de comprensión lenta, y los programas de ejercicios en los que, por ejemplo, el niño ha de seleccionar un sustantivo de entre un grupo de palabras, o formar una palabra a partir de una serie de letras, resultan herramientas muy útiles para la educación. Sin embargo, la utilización del ordenador de esta manera nos lleva a sustituir al maestro humano, y de ahí surge una aseveración importantísima: ningún ordenador puede reemplazar al maestro de carne y hueso. El contacto humano constituye el elemento más importante en el proceso de aprendizaje y, si bien el ordenador es el recurso educativo más eficaz que se conoce, no puede sustituir la enseñanza afectuosa impartida por el maestro.

Los ordenadores proporcionan un gran entretenimiento, por lo que resulta una buena idea que se permita a los niños practicar juegos con ellos. A muchos padres les preocupa la posible alienación que pueden producir en el niño juegos como los "invasores del espacio" o Pac-Man, pero aunque dichos juegos son muy divertidos, no existe la más mínima evidencia que sugiera que el atractivo que ejercen sobre los niños rebase los límites de la simple fascinación.

La lógica del LOGO

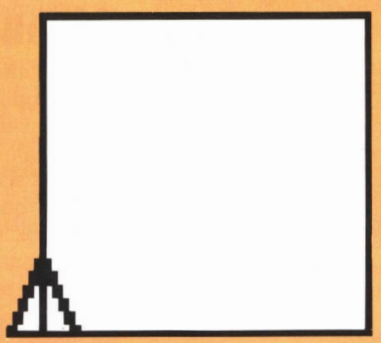
Mostramos aquí cómo se construyen formas en la pantalla mediante la utilización del lenguaje LOGO.

El LOGO es un lenguaje de ordenador desarrollado específicamente para que los niños pequeños (de 4 o 5 años) programen un ordenador. Este lenguaje fue elaborado a finales de la década de los sesenta en el Massachusetts Institute of Technology (Estados Unidos) por un equipo dirigido por Seymour Papert, un matemático que había trabajado con Jean Piaget, el famoso pedagogo, en el centro que éste regentaba en Ginebra.

Para los niños más pequeños, el LOGO se convierte en una "tortuga", ya sea en forma de robot mecánico o como un triángulo luminoso en la pantalla de un ordenador. La orden FORWARD 10 hace que la tortuga se desplace 10 unidades hacia adelante, dibujando una línea. La orden RIGHT 90 determina que la tortuga trace un ángulo recto. Mediante secuencias de órdenes puede conseguirse que la tortuga dibuje cuadrados, triángulos, círculos e incluso formas menos convencionales. Asimismo, es posible enseñar a la tortuga a "recordar" las órdenes. Los niños, sin advertirlo, mientras enseñan a la tortuga no hacen más que programar un ordenador

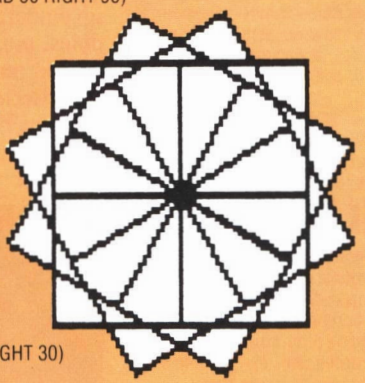
Modo de dibujar un cuadrado:

```
FORWARD 50
RIGHT 90
FORWARD 50
RIGHT 90
FORWARD 50
RIGHT 90
FORWARD 50
RIGHT 90
```



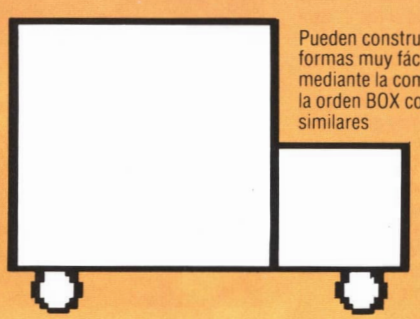
Cómo construir una orden «BOX» (recuadro):

```
TO BOX
REPEAT 4 (FORWARD 50 RIGHT 90)
END
```



La orden STAR (estrella):

```
TO STAR
REPEAT 12 (BOX RIGHT 30)
END
```



Pueden construirse otras formas muy fácilmente mediante la combinación de la orden BOX con otras similares

Fatiga visual

Algunas personas que trabajan frente a pantallas de ordenador sufren de fatiga visual; pero el período en que los niños permanecen frente a la pantalla no es tan prolongado como para que surja este problema. En el caso de los adultos, el problema parece ser de "acomodación", un fenómeno que se produce cuando el ojo permanece fijo a una distancia focal determinada y precisa de un cierto tiempo para reacomodarse. Si usted advierte que su hijo se concentra demasiado en el ordenador, mirando fijamente a la pantalla durante largos períodos, el problema podrá solucionarse haciéndole que tome un breve descanso cada quince minutos.

Es necesario mencionar un problema que plantean los televisores antiguos. Se ha descubierto que algunos televisores en color fabricados antes de 1970 emiten una dosis de baja radiación que puede resultar peligrosa si se trabaja regularmente muy cerca de ellos. Si decide utilizar su antiguo televisor en color para trabajar con el ordenador, es conveniente que compruebe si ha sido fabricado después de 1970

Los niños menores de siete años necesitan ayuda y supervisión para encender el ordenador y el televisor, y cargar un programa. Si éste es bastante bueno, se puede dejar que lo utilicen por sí solos, aunque esto dependerá de su destreza en la lectura y de su capacidad para responder a las preguntas del programa. Las exigencias en cuanto al hardware son muy sencillas. Un microordenador debe ser sólido: los niños golpean el teclado con los puños cerrados, tiran de las conexiones eléctricas y constantemente tocan la pantalla. Si el sistema es frágil, las conexiones inseguras o resulta difícil de utilizar, no despertará el interés de un niño pequeño. Algunos expertos consideran que el teclado idóneo de un ordenador para niños de corta edad debe ser grande y con teclas claramente definidas. Sin embargo, cuando el niño ya ha desarrollado por completo su capacidad motriz (generalmente a los siete años), es capaz de manipular teclados que incluso resultarían difíciles para un adulto. Los teclados sensibles al tacto de los microordenadores Sinclair más baratos no resultan convenientes para niños menores de 9 o 10 años; por el contrario, las versiones mayores del teclado impreso, como el del Philips Videopac 7000, son adecuadas incluso para niños de cuatro años de edad.

Jóvenes programadores

Para los niños de corta edad la elección del software es más complicada. Si proyecta utilizar un sistema basado en cassettes, usted deberá supervisar el proceso de carga y almacenamiento por completo. Por el contrario, si el sistema que utiliza es a base de discos, comprobará que sus hijos pueden manejar de forma correcta los discos flexibles. Para los más pequeños, niños menores de siete años, uno de los mejores dispositivos para almacenar programas es el cartucho ROM, una caja de plástico que contiene un circuito integrado con un programa fijado eléctricamente. El inconveniente de los sistemas de este tipo consiste en que no permiten al usuario almacenar ningún trabajo; sin embargo, los cartuchos son virtualmente indestructibles, por lo que los niños pueden utilizar el ordenador sin ninguna supervisión.

Si decide adquirir un ordenador exclusivamente para sus hijos, intente colocarlo en un lugar determinado, ya que trasladarlo de una habitación a otra, con la conexión y desconexión de cables que ello supone, si bien no dañará el ordenador (a menos que se le caiga al suelo), posiblemente haga que el niño renuncie a todos esos preparativos y opte por una actividad menos conflictiva, como encender el televisor, por ejemplo.

El centro de trabajo ideal

En una situación ideal, el ordenador del niño debería instalarse en su habitación, junto a su propia pantalla de televisión. Si usted está decidido a que la actitud de sus hijos hacia los ordenadores sea positiva, debe considerar la posibilidad de instalarles un centro de trabajo en alguna de sus habitaciones y adquirir un televisor de segunda mano para su uso exclusivo. (Instale la central de ordenadores en la habitación de su hijo mayor. Es posible que él o ella decidan utilizar el ordenador cuando sus hermanos se hayan ido a dormir.) Un televisor en blanco y negro antiguo puede comprarse por muy poco dinero y, siempre que disponga de sintonizador de canales, resultará adecuado para visualizar en pantalla la información de un ordenador.

Existe una amplia gama de argumentos acerca del efecto del color en los ordenadores destinados a niños pequeños; para algunos especialistas el color constituye un elemento vital, mientras que para otros es sólo un atractivo adicional, aunque innecesario. Resulta obvio que si debe optarse entre una conexión permanente a un televisor en blanco y negro en el dormitorio de los niños, y una conexión temporal al televisor en color de la sala de estar familiar, la instalación de carácter permanente es, indudablemente, la más acertada.

Si usted puede instalar un centro de trabajo/juego con ordenador, ya sea permanente o semipermanente, en una de las habitaciones de sus hijos, constituye una idea acertada disponerlo todo de modo que sea posible trasladar el ordenador sin alterar la instalación.

Cuando proceda a instalar la mesa para el ordenador, asegure los cables y conexiones, de modo que los niños no los arranquen accidentalmente. (Asegúrese asimismo de que todos los enchufes estén bien protegidos y cubiertos por cinta aislante para que no ofrezcan peligro. Y cuide que el televisor no esté conectado a la antena; de lo contrario, puede aparecer "accidentalmente" en la pantalla la última película, cuando los niños se hayan ido a la cama.) Es muy importante que el ordenador repose en una base estable y no oscile hacia ningún lado. La Sinclair ofrece una bandeja que mantiene en posición fija su ordenador Spectrum, que es muy ligero; si sus hijos son muy traviesos, idee algún sistema, una abrazadera o algo similar, para fijar el microordenador. Por supuesto, el ordenador que usted adquiera para sus hijos también será utilizado por toda la familia, de modo que, en ese caso, es mejor comprar un duplicado de los cables y (de ser necesario) un segundo paquete de alimentación principal. Éstos son relativamente baratos y le permitirán darles las buenas noches con firmeza a sus hijos, desenchufar el ordenador y la grabadora (o la unidad de disco) de su centro de trabajo en el dormitorio (dejando todos los cables enrollados en el mismo lugar), y enchufarlos en su propio televisor utilizando los duplicados de los cables.

El Big Trak

Aunque parece un carro de combate de juguete, en realidad se trata de un eficaz instrumento de aprendizaje. El Big Trak es un juguete programable mediante un ordenador, que permite al niño planificar con toda precisión los movimientos que desea que efectúe el carro. Este pequeño vehículo es capaz de memorizar hasta 16 pasos y se puede programar para que deambule de una habitación a otra de la casa antes de regresar a su base. Mientras el niño se divierte, el ordenador le ayuda a explorar su mundo físico y a elaborar los pasos individuales necesarios en un programa sencillo para ordenador. Pese a su aspecto agresivo, el Big Trak es del agrado tanto de niños como de niñas

Control del ordenador

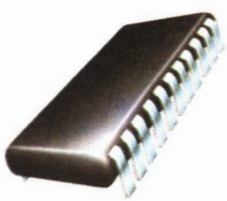
El "hardware" de su ordenador no funcionará sin la ayuda de un "software" adecuado. Conozcamos su significado y valoremos el software que se puede hallar en el mercado

El *software* constituye la mitad invisible de un sistema de ordenador. Sin él, el ordenador no es más que una masa inerte de maquinaria electrónica. Un ordenador que carezca de software no puede hacer absolutamente nada.

Si examina los chips de silicio situados en el interior de un ordenador, verá que consisten en miles, quizá millones, de interruptores electrónicos microscópicos. Así como un interruptor de luz no puede encender o apagar una lámpara por sí mismo, los interruptores de un ordenador necesitan ser encendidos o apagados. Sin embargo, no se encienden o apagan todos al mismo tiempo. Cada uno de ellos ha de ser encendido (o apagado) de manera individual y según una secuencia exacta y precisa en relación con los otros miles de interruptores. El software es la forma en que esto se lleva a cabo.

Software es el nombre con que se denomina a las instrucciones que hacen que el ordenador trabaje. Estas instrucciones asumen la forma de números que, al ser introducidos en la CPU (el corazón del ordenador), preparan y restauran los interruptores internos para que éstos produzcan cosas específicas. Estos números sólo son "comprendidos" por el ordenador cuando están en el denominado sistema binario (convertidos en unos y ceros).

Estos unos y ceros que el ordenador comprende (en el sentido que le hacen realizar tareas específicas) son el producto final de una larga cadena de acontecimientos que comenzaron como ideas en la mente de la persona que escribió el programa. Un programa para ordenador (la palabra "programa" designa una unidad determinada de software) puede existir de muchas maneras diferentes. Lo único que podemos afirmar en concreto de cualquier programa es que debe acabar de una forma comprensible para el ordenador. Tomemos un ejemplo específico. Supongamos que un ingeniero de tráfico desea controlar un conjunto de semáforos a través de un ordenador. Para ello, el ordenador de control necesitará de un programa que prevea la secuencia correcta de acontecimientos (¡de qué serviría que todos los semáforos se pusieran en verde al mismo tiempo!). Pero antes de que este software pueda escribirse, el ingeniero debe pensar en profundidad qué es exactamente lo que él desea que haga el ordenador. Comúnmente estas ideas se escribirían utilizando ora-

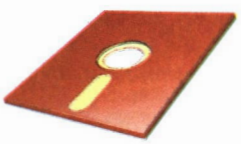


ROM

El ROM (Read Only Memory) es uno de los principales dispositivos de memoria de ordenadores. Producto de la revolución de los chips de silicio, permite almacenar con carácter permanente los programas del ordenador. La mayoría de los ordenadores personales están equipados con un chip ROM que contiene el lenguaje BASIC. En algunos el ROM puede comprarse por separado con el fin de ampliar sus prestaciones incorporándoles otro lenguaje. También existen ROMs procesadores de textos

CASSETTE

Con frecuencia el software se ofrece en cintas de cassette, idénticas a las que se emplean para grabar sonido. Los programas de juegos suelen ofrecerse en esta forma. El programa se pasa de la cinta al ordenador conectando la máquina a una grabadora de cassette normal y reproduciendo (botón "play") la cassette con el programa. Cuando el programa ha sido cargado, la cinta se detiene y generalmente el ordenador ya no necesita volver a "leerla"



DISCO FLEXIBLE

El software (los programas) puede almacenarse grabándolo en un disco de película magnética. La grabación se realiza sobre "pistas" en la superficie, similares a los surcos de un LP común, mediante una cabeza magnética de "lectura y escritura" que también "lee" (reproduce) el programa al ser requerida. Los discos ofrecen enorme capacidad y alta velocidad de operación, pero requieren sofisticadas "unidades de disco" (ver p. 8)

CARTUCHO

Un cartucho es, básicamente, un ROM convenientemente empaquetado. Algunos ordenadores poseen conectores de fácil acceso en los cuales pueden enchufarse los cartuchos. El software que se ofrece en cartuchos suele incluir un lenguaje de programación (como el BASIC) o bien sofisticados programas de juegos

Para que su ordenador funcione, es necesario "alimentarlo" con software (una serie de instrucciones electrónicas). Los dispositivos que aquí se describen son los "medios" que sirven para almacenar dichas instrucciones. Representan las cuatro formas más comunes en las que se proporciona el software. Cada una de ellas ofrece ciertas ventajas particulares. El software se confecciona a la medida de cada ordenador; el programa escrito para un ordenador determinado no siempre puede ser utilizado con otro

ciones corrientes. (Por ejemplo: "En este momento quiero que el semáforo número uno cambie a ámbar manteniendo encendida la luz roja. Luego quiero que tanto la luz roja como la luz ámbar se apaguen y que se encienda la verde".) Es evidente que estas oraciones no están escritas de forma comprensible para ningún ordenador y, por tanto, han de ser convertidas en un programa. El ingeniero utiliza un lenguaje de programación, como el BASIC. Un lenguaje de este tipo permite reescribir los pensamientos ordenados lógicamente (en español) de forma que el intérprete BASIC los pueda comprender. El intérprete BASIC es, en sí mismo, un programa que convierte el programa original (escrito en BASIC) en la forma que le resulte comprensible a la unidad central de proceso (CPU). El software de esta forma se denomina "lenguaje máquina" o "código en lenguaje máquina".

El software que usted compra para utilizar con su ordenador siempre está en lenguaje máquina y es almacenado en una forma rápidamente accesible al ordenador. Algunas veces el software se almacena en el interior del ordenador, en la memoria ROM. Lo más común es que se ofrezca en cassette o en disco flexible. Estos objetos no constituyen en sí mismos el software, sino que son el "medio" a través del cual se proporciona el software. Para ser utilizado por el ordenador, el software ha de ser transferido desde la cassette (o desde el disco flexible o el ROM) al ordenador. Una vez cargadas estas instrucciones (así se denomina el proceso de transferencia de software), el programa puede comenzar a funcionar.

Cómo comprar software

Si usted tuviera en el banco medio millón de pesetas, quizá se diría: "Creo que voy a comprarme un coche". Es poco probable que alguien en esa situación se dijera: "Creo que voy a comprarme una máquina", porque la pregunta obvia sería: "¿Qué clase de máquina? ¿Que sirva para qué?"

Con el software ocurre lo mismo. Un ordenador es, en sí mismo, un objeto inerte, pero el software que usted compre para utilizar con el ordenador es capaz de convertirlo en una máquina de juegos, en una máquina de escribir automática o en un experto contable doméstico. De modo que lo primero que ha de decidir es qué función quiere que realice el ordenador.

Comience por determinar el problema y luego busque el software que le proporcione las soluciones. En la búsqueda del programa adecuado, a medida que usted analice sus necesidades reales se irá produciendo un proceso de decantación natural. Si el punto de partida es cómo entretener a los niños un domingo por la tarde, la etapa siguiente será la de buscar algún tipo de programa que con toda seguridad les proporcione ese entretenimiento. Los juegos para ordenador comprenden desde "masacres" de seres extraterrestres hasta complicadas y emocionantes simulaciones fantásticas. Si lo que usted desea proporcionarle al ordenador es un programa de juegos en serie, la pregunta siguiente será si existen o no para su máquina.

Debido a que las diferencias existentes entre un ordenador y otro no se refieren sólo a su aspecto (cada ordenador posee en su interior una electrónica propia y requiere un software escrito especialmente para él), todos los modelos son prácticamente incompatibles. Un programa que funcione con el Atari 800 no servirá para el Spectrum (a menos que se fabrique una nueva

FIRMWARE

El origen del término "hardware" es obvio: se trata de la parte física y electrónica del ordenador (las conexiones para la alimentación eléctrica, teclado, chips de silicio, etc.). La palabra "software" sugiere, en cambio, una naturaleza intangible, y designa simplemente a una serie de instrucciones. Los especialistas en ordenadores hablan también de "firmware". En las primeras épocas de las computadoras, el software se codificaba y almacenaba en cintas de papel perforadas similares a las que emplean los operadores de télex. Luego las cintas de papel fueron reemplazadas por cassettes y discos magnéticos. En los años 70 se inventó una nueva técnica para almacenar software en chips ROM, diseñados a tal fin. A los chips ROM se les incorporan las instrucciones de software en el proceso de fabricación. Se denomina "firmware" a la combinación de software "intangible" y hardware "tangible"

Editar

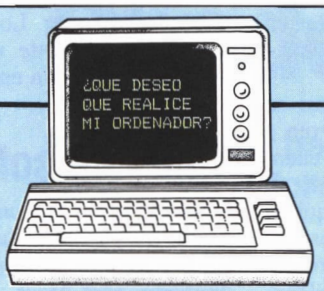
Tratamiento de textos

Con un programa de tratamiento de textos, su ordenador le ofrece avances notables en relación a la máquina de escribir convencional. Hasta los más eficientes mecanógrafos suelen cometer errores, pero con un procesador de textos usted puede conseguir cartas perfectas impresas y, al mismo tiempo, aumentar la productividad.

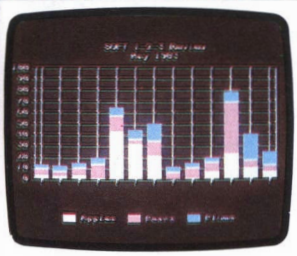
El teclado del ordenador sustituye a las teclas de la máquina de escribir. Las palabras que usted teclea aparecen instantáneamente en la pantalla, así como aparecerían, con la máquina de escribir, sobre el papel. Pero aquí termina toda similitud y entra en juego el potencial del ordenador.

En la pantalla los errores pueden corregirse de forma instantánea. Las palabras se reescriben o se hacen desaparecer. Incluso pueden borrarse párrafos enteros. Sin embargo, los procesadores de textos no sólo sirven para borrar palabras. Si sus pensamientos quedaran expresados con mayor claridad reordenando las oraciones, usted puede hacerlo ahí mismo, en la pantalla. Las palabras u oraciones que desea desplazar a otro sitio de la "hoja" se borran temporalmente (el programa de tratamiento de textos las retira de la pantalla y las almacena en la memoria del ordenador). Luego usted puede insertarlas exactamente donde lo desee.

Una vez redactado el documento en la forma que usted desea, puede imprimirse utilizando la impresora del ordenador, o almacenarse en una cassette o en un disco flexible para uso ulterior



Llevar libros de cuentas



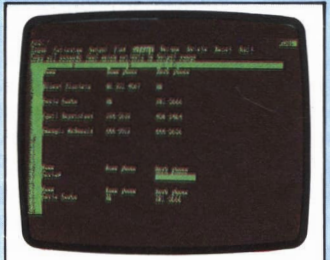
Labores contables

Si consideramos que los ordenadores pueden realizar operaciones matemáticas, no debemos sorprendernos de que existan muchos programas para ayudar al hombre de negocios. La gama de programas de gestión contable es impresionante, desde teneduría de libros automatizada hasta balances completos. Por lo general los programas de este tipo deben manejar grandes cantidades de información y necesitan almacenar un elevado número de documentos. Por este motivo suelen requerir al menos una unidad de disco flexible para hacer frente a tal demanda de almacenamiento.

Los programas contables trabajan generalmente a través de un sistema de preguntas (visualizadas en la pantalla del ordenador) y respuestas (proporcionadas por el operador del ordenador). La información que teclea el operador es manipulada por el programa del ordenador; hechos todos los cálculos, los resultados se almacenan en el disco flexible o se imprimen mediante la impresora.

Este tipo de programas incluyen la expedición automática de facturas, reestructuración de stock y confección de libros mayores, mientras se controla el trabajo en curso. Este software suele ser caro, pero puede constituir una buena inversión para el empresario, puesto que reduce el costo del trabajo y ofrece rápidos resultados

Archivo



Base de datos

Los ordenadores pueden buscar información en los archivos mucho más rápido que una persona; cuanto mayor sea la cantidad de información entre la cual usted ha de buscar un dato, mayor es la ayuda que puede proporcionarle un ordenador. En su forma más simple (y más barata), una base de datos es poco más que una agenda computerizada que puede buscar nombres, direcciones y números de teléfono. Los programas de base de datos más sofisticados y más caros pueden realizar operaciones mucho más complicadas.

Para formarnos una idea del potencial de una base de datos, tomemos el ejemplo de un botánico que esté recogiendo información para un libro que tratará sobre setas exóticas y venenosas. Este botánico habrá confeccionado extensos ficheros acerca de las diversas especies y su medio natural. También habrá tomado apuntes extraídos de diversos libros de consulta y tendrá una lista bibliográfica interminable.

Antes de que el ordenador se convirtiera en una máquina asequible, la información recogida debía ser transcrita en fichas, que tenían que ser ordenadas alfabéticamente en un fichero. Con un programa de base de datos y un ordenador, toda la información puede almacenarse en la memoria de éste. Valiéndose del potencial de la base de datos, el botánico puede obtener al instante la respuesta a sus consultas. Si necesitara confeccionar una lista de todas las setas que existen en Cataluña, la base de datos podría hacerla por él. Si precisara una lista alfabética de todos los libros que contengan las palabras "veneno" o "venenoso" y "hongo" o "seta", también la base de datos trabajaría por él.

Por lo general, las bases de datos sólo se ofrecen en discos flexibles

Manejar números

Hojas electrónicas

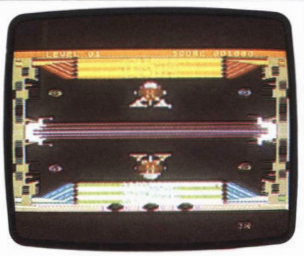
La "hoja electrónica" es la respuesta a todas esas preguntas como "¿y qué pasaría si...?", que suelen resolverse con una calculadora y varias bobinas de papel. Toda empresa que tiene un producto para vender posee muchas variables. La modificación de una de ellas afecta por lo general a las otras.

Consideremos las preguntas que puede formularse el propietario de un cine: "¿Si vendiera todas las localidades, a cuánto podría rebajar el precio de cada localidad?"; "¿Obtendríamos mayores ingresos bajando el precio de los helados y manteniendo la misma cantidad de vendedores, o sería más rentable aumentar el precio y tomar dos empleados nuevos?". Es probable que ambas decisiones afecten a todo el negocio; el descenso de los precios podría implicar mayores ventas, pero reducir el margen de beneficios. Una hoja electrónica es un programa especial que puede ofrecer respuestas inmediatas para este tipo de planteamientos.

Todos los números básicos que han de manejarse se disponen en forma de líneas y columnas y se especifica la relación existente entre cada línea y cada columna (por ejemplo: los números de cada línea de la columna C son el resultado de restar al número de la columna A el número de la columna B). Una vez se ha reunido toda la información real y la hipotética, cada una de las cifras puede ser alterada y la "incidencia" de este cambio en las otras cifras queda instantáneamente a la vista.

Los usuarios de hojas electrónicas suelen ser ejecutivos responsables de costos o ingenieros y científicos que deben trabajar con información numérica variable. Por lo general, las hojas electrónicas requieren unidades de disco y una impresora

Diversión



Juegos

Los ordenadores no sólo son excelentes para el tratamiento de números y de textos. También pueden proporcionar muchas horas de diversión si se utilizan con uno de los muchos programas de juegos disponibles. Éstos abarcan una amplia gama que va desde juegos de mesa, como el ajedrez y el backgammon, hasta los juegos de los salones recreativos (como el "aterrizaje lunar" y los que simulan vuelos). También existen juegos de aventuras muy complejos que pueden durar días y hasta semanas. Muchos juegos para ordenador tienen también un interesante valor educativo.

Los juegos para ordenador requieren una atención y participación constante por parte del jugador. Por lo general esta participación se realiza a través del teclado; una tecla puede utilizarse para disparar un "láser" o un "misil", o para controlar el movimiento de algo que aparezca en la pantalla. El número de teclas a emplear dependerá del juego elegido y del nivel de control que exija el programa.

Una alternativa al teclado muy extendida es la palanca de mando. Estos dispositivos se enchufan en el ordenador y, en cierto modo, operan como las palancas de mando de los aviones. Proporcionan un mayor control y hacen los juegos por ordenador mucho más atractivos

Programas listos para utilizar

La mayoría de los programas que existen en el mercado se venden en disco o en cassette con un manual de instrucciones para su uso. El Apple Writer es un clásico programa de tratamiento de textos. Consta de un único disco flexible y de un manual en el que se explica cómo utilizarlo. El manual incluye un cursillo para que los principiantes puedan comenzar a usarlo de inmediato. Los niveles de documentación varían enormemente. Algunos manuales son tan incompletos y están tan mal escritos, que la utilización del programa resulta difícil y hasta imposible. Éste es un factor a tener en cuenta al comprar un programa

versión especial del Spectrum); de manera que usted debe comprar aquellos programas que hayan sido producidos especialmente para su ordenador.

Pero aún no está usted preparado para hacer su compra. Las consideraciones siguientes se refieren a las limitaciones físicas de su máquina. Verifique cuánta memoria posee su ordenador. Si tiene 16 K de ROM, compruebe si el juego que usted desea requiere que se le agregue memoria. Por regla general, los juegos más interesantes y más sofisticados exigen programas más largos, por lo cual necesitará que su ordenador disponga de mayor memoria. Y no olvide que el software se ofrece en distintos soportes (véase página anterior). Si un programa sólo está disponible en disco flexible y usted posee únicamente cassette, no podrá utilizarlo si antes no adquiere una unidad de disco. Algunos programas (en especial los de juegos) requieren también otros accesorios, por ejemplo, palanca de mando. Probablemente lo que no necesite para sus juegos sea la impresora, pero los programas de oficina a menudo sí la necesitan para imprimir los resultados.

Por último, está lo concerniente a cuánto dinero puede usted gastar. Los precios de ciertos juegos presentados en cassette varían notablemente. Algunos paquetes de gestión en disco suelen ser caros.

Fausto Dorelli

Tipos de software

En cierto modo, los juegos constituyen una clase aparte. Al fin y al cabo, la función de un juego es la de entretener. La mayoría de los otros programas están diseñados para hacer que un trabajo determinado se realice más fácil y rápidamente. En este aspecto, los informáticos han logrado aumentar la rentabilidad y la eficacia de muchísimas maneras. Pensemos en la pobre mecanógrafa cuyas copias no satisfacen al jefe. Con un microordenador y un programa de tratamiento de textos, el ordenador reemplaza a la mecanógrafa y todas las correcciones se hacen en pantalla. Una vez que todas las palabras se han escrito en la forma correcta, la página completa puede imprimirse en papel automáticamente y con sólo pulsar una tecla. Se obtendrán así valiosos resultados en cuanto a economía de tiempo y se evitarán frustraciones.

Otro trabajo agotador que se presta a la computación es la administración financiera. Muchas de las actividades que solían mantener ocupados a ejércitos de empleados (calculando salarios y actualizando los libros de la empresa) pueden ahora llevarse a cabo mediante software escrito con esa finalidad. Los programas en sí mismos son bastante especializados, por lo cual es poco probable que un único programa satisfaga todas las necesidades de una oficina. Las categorías incluyen programas de "nóminas" para calcular los salarios e imprimir las hojas de recibo, programas de "control de stock" para mantener actualizada la relación de los productos vendidos o utilizados (en algunos casos el programa puede encargarse de solicitar nuevos stocks automáticamente) e incluso existen programas para ayudar a decidir el tamaño y la calidad de papel más económicos para imprimir libros o revistas.

Otras tareas en las que los ordenadores alcanzan cotas espectaculares de eficacia son las de archivo y selección de información. Este tipo de programas se denominan "base de datos". Las bases de datos pueden reemplazar muebles archivadores completos y realizar toda la labor de ordenación y clasificación.

La última categoría general de software es la que se conoce como "hoja electrónica". Un programa de hoja electrónica permite confeccionar y modificar cuantas veces se desee complicados presupuestos o previsiones financieras, en lugar de tener que recurrir a la calculadora y a las bobinas de papel.

Todos los tipos de software a los que nos hemos referido se venden "hechos". Están "listos para utilizar" en el sentido que el informático tenía en mente una serie específica de soluciones para los problemas que alguien le había planteado. No obstante, puede llegar el día en que ningún programa de los disponibles en el comercio haga que su ordenador realice exactamente la función para la cual usted lo necesita. ¿Y qué hará usted en ese caso? Una solución, si bien bastante onerosa, consiste en "alquilar" a un programador de ordenadores para que le escriba un programa hecho exactamente a la medida de sus necesidades. La otra solución consiste en que usted mismo aprenda a escribir sus propios programas. Si llega a dominar un lenguaje como el BASIC, le será posible producir programas que hagan que su ordenador realice toda clase de cosas sorprendentes.

Obedeciendo órdenes

Su ordenador hará exactamente lo que usted desee cuando le "hable" en la forma correcta... y él no se equivocará nunca

Es perfectamente factible que cualquier persona utilice un ordenador (en su casa o en el trabajo) sin saber nada en absoluto acerca de cómo funciona. Partiendo de este punto, iniciamos un curso que explica paso a paso, desde el principio, todo lo que usted necesita saber para crear con éxito sus propios programas.

Al cabo de cierto tiempo muchas personas descubren que los programas y los juegos que han comprado para su ordenador empiezan a resultarles un poco aburridos, y se preguntan si pueden modificarlos o incluso escribirse sus propios programas. Pero un ordenador no puede hacer nada por sí mismo. Se le debe proporcionar una lista de instrucciones que le informen con todo detalle qué es exactamente lo que debe hacer y qué pasos dar para poder conseguirlo. Estas instrucciones conforman lo que se denomina un *programa,* y el arte de crear un programa se conoce como *programación*.

La programación no es particularmente difícil. No requiere siquiera que usted tenga aptitud para las matemáticas, a menos, por supuesto, que desee escribir programas para llevar a cabo tareas matemáticas. En principio, lo único que necesita es comprender BASIC.

Su primer lenguaje

Muchos ordenadores personales traen incorporado un lenguaje denominado BASIC. Como su nombre indica, está diseñado para que los principiantes aprendan las nociones básicas de la programación de forma rápida y sencilla. Como cualquier lenguaje humano, el BASIC posee una gramática, un vocabulario y una sintaxis propios, si bien el vocabulario es muchísimo más reducido que el de un idioma normal. El BASIC utiliza varios vocablos ingleses cortos, fáciles de reconocer y sencillos de aprender. En términos generales, el BASIC es adecuado tanto para el principiante como para el usuario más experimentado.

No obstante, uno de los inconvenientes de este lenguaje radica en que, a través de los años, los distintos fabricantes de ordenadores han ido incluyendo sus propias modificaciones. Como resultado de ello existe una gran cantidad de variantes del BASIC, particularmente respecto a las órdenes para controlar los aspectos de la máquina de desarrollo más reciente, por ejemplo, color, gráficos y sonidos. Todas las variaciones del BASIC que se producen en los ordenadores más populares se muestran en cada lección, en el recuadro "Complementos al BASIC".

Debido a las variaciones que experimenta el BASIC de un ordenador a otro, es prácticamente imposible escribir un programa en BASIC que pueda llevarse con cualquier ordenador. No obstante, afortunadamente el lenguaje posee un núcleo común, que por lo general es el mismo en todas las máquinas. Comenzaremos centrándonos en ese núcleo y, a medida que el curso avance, iremos introduciendo programas más complicados.

Los primeros pasos

Empecemos por escribir un pequeño programa y veamos lo que sucede. Éste nos mostrará cómo el ordenador comete, aparentemente, un error. Encienda el ordenador y digite el programa tal y como se lo presentamos, incluyendo todos los espacios. El <CR> al final de cada línea es para recordarle que digite *Carriage Return* (retorno de carro). En su ordenador, esta tecla puede estar señalizada como RETURN, ENTER o incluso con el signo ↵.

```
10 REM LOS ORDENADORES NUNCA SE
   EQUIVOCAN <CR>
20 PRINT "DIGITE UN NUMERO" <CR>
30 INPUT A <CR>
40 LET A = A + 1 <CR>
50 PRINT "CREO QUE EL NUMERO
   DIGITADO ERA"; <CR>
60 PRINT A <CR>
70 END <CR>
```

Después de haber tecleado todo lo anterior, pulse LIST <CR>. El programa que usted acaba de digitar debería reaparecer en la pantalla. LIST es una instrucción para que el ordenador "imprima" un listado del programa en memoria. Si el programa aparece correctamente en pantalla después de digitar LIST, podemos tratar de hacerlo funcionar (RUN). No se preocupe si comete algún error al pulsar el programa. Después de haber listado (LIST) el programa, simplemente vuelva a digitar la línea que contenga el error. No se olvide del número de la línea. Pruebe tecleando

```
25 REM AQUI HAY OTRA LINEA 'REM'<CR>
```

y luego digite (LIST) el programa otra vez. Para librarse de la línea, pulse sólo el número de línea, seguido de <CR>. Cuando esté satisfecho y el programa haya sido digitado correctamente, puede "ponerlo en funcionamiento" digitando RUN<CR>. Inténtelo y verá aparecer en pantalla:

```
DIGITE UN NUMERO
```

Siga adelante y pulse un número. Pruebe con el 7 (utilice los guarismos; recuerde que el ordenador no reconocería que "siete" es 7 a menos que lo programemos especialmente para ello). Después de apretar 7, en la pantalla se leerá:

```
CREO QUE EL NUMERO QUE HA DIGITADO ERA 8
```

¿Se ha equivocado realmente el ordenador, o simplemente estaba obedeciendo órdenes? Si examinamos el programa línea por línea podremos ver cada una de las instrucciones que le hemos proporcionado al ordenador. He aquí la primera línea:

```
10 REM LOS ORDENADORES NUNCA SE EQUIVOCAN
```

Otros lenguajes

El lenguaje más utilizado en la mayoría de microordenadores es el BASIC. Pero de ninguna manera el BASIC es el único lenguaje. Antes de que aparecieran los microordenadores, cuando la mayoría de los cálculos se realizaban en ordenadores de unidad principal que ocupan cuartos enteros, los científicos y los ingenieros empleaban un lenguaje denominado FORTRAN. En el mundo de los micros, otros lenguajes que gozan de popularidad son el PASCAL, el FORTH y el LOGO

Pascal

Al igual que el BASIC, el PASCAL fue desarrollado fundamentalmente como un lenguaje de enseñanza para los estudiantes de programación. Es muy apreciado por los profesores de informática, puesto que favorece la escritura de programas sofisticados y muy bien urdidos. Por lo general el PASCAL se ofrece en discos flexibles y suele ser caro. Para el Spectrum existe una versión económica en cassette

Forth

Los programas escritos en lenguaje FORTH se asemejan mucho menos al idioma inglés que el BASIC o el PASCAL. El FORTH también resulta más difícil de aprender. Sin embargo, tiene la ventaja de poseer una mayor riqueza de expresión, puesto que en unas pocas líneas pueden escribirse complicados programas. Con el FORTH usted puede definir sus propias órdenes, mientras que, en el caso del BASIC, éstas ya vienen predefinidas

Logo

El LOGO es un lenguaje relativamente nuevo que está popularizándose en el campo de la educación. Tiene la gran ventaja de que es lo suficientemente simple como para que incluso los niños pequeños puedan aprenderlo. Puede ayudar a enseñar técnicas de programación y también favorece un acercamiento lógico al diseño de programas desde un primer nivel. El LOGO utiliza gráficos "tortuga" que permiten que las imágenes aparezcan rápidamente en pantalla. También se puede conectar al ordenador una tortuga mecánica. Al pulsar sencillas órdenes en el teclado, la tortuga se mueve y dibuja líneas y formas

REM es la abreviatura de REMark (observación). Todo lo que aparece en la misma línea después de REM es ignorado por el ordenador. Las observaciones constituyen una manera muy práctica de hacernos recordar lo que está haciendo el ordenador. Esta REM en particular no es más que un título, no nos indica lo que está haciendo el programa. Ya avanzado el curso, veremos la gran ayuda que representan unas REM bien escritas. Ahora echemos una mirada a:

```
20 PRINT "DIGITE UN NUMERO"
```

Cuando el BASIC llegue a la palabra PRINT, todo lo que sigue a continuación es "impreso" en la pantalla del ordenador. Observe que la oración está entre comillas dobles. Una de las reglas del BASIC es que los caracteres (letras) que aparecen entre comillas dobles después de la enunciación PRINT aparecerán en pantalla exactamente como fueron digitados. En la línea 60 veremos otra forma de utilizar PRINT. A continuación sigue:

```
30 INPUT A
```

Por ahora pasaremos esta línea por alto; volveremos a ella después de ocuparnos de la línea 40.

```
40 LET A = A + 1
```

Aquí la letra A está utilizada como una variable. Una variable es como una caja etiquetada que puede contener tanto un número como algunos caracteres. En lugar de tener que recordar qué contiene la caja, lo que tenemos que saber es cómo se llama la caja para poder referirnos a ella. Es algo así como decir "Alcánceme la caja que lleva la etiqueta B" en lugar de decir "Alcánceme la caja que contiene los tornillos de 15 mm de cabeza".

En esta línea tenemos una «caja» denominada A. Esta caja se llama variable porque el valor de lo que pongamos en ella puede variar. A una variable podemos asignarle virtualmente cualquier valor. En la línea 30 le habíamos asignado un valor a la variable A, así que volvamos a ella:

```
30 INPUT A
```

La utilización de la palabra INPUT es, en BASIC, uno de los modos de asignarle (otorgarle) a una variable un valor específico. Cuando el programa BASIC llega a una línea que comienza con INPUT, espera a que algo sea digitado en el teclado. INPUT A permite que el ordenador sepa que tenemos una variable denominada A y que lo que se digite en el teclado será el valor asignado a esa variable. En este punto, digitando 7<CR> coloca 7 en la caja A o, utilizando términos informáticos, asigna el valor 7 a la variable A. Ahora que ya sabemos en qué consiste una variable y que conocemos uno de los modos de asignarle un valor, volvamos a ocuparnos de la línea 40.

```
40 LET A = A + 1
```

El nombre de la variable a la cual se le asigna un valor aparece siempre a la izquierda del signo igual. Aquí le estamos dando a A un nuevo valor. La sentencia significa "Hagamos (LET) que el nuevo valor de A sea igual al antiguo valor más 1". El antiguo valor de A era 7. Ahora lo hemos convertido en 7 + 1, de modo que el nuevo valor es 8.

```
50 PRINT "CREO QUE EL NUMERO DIGITADO ERA";
```

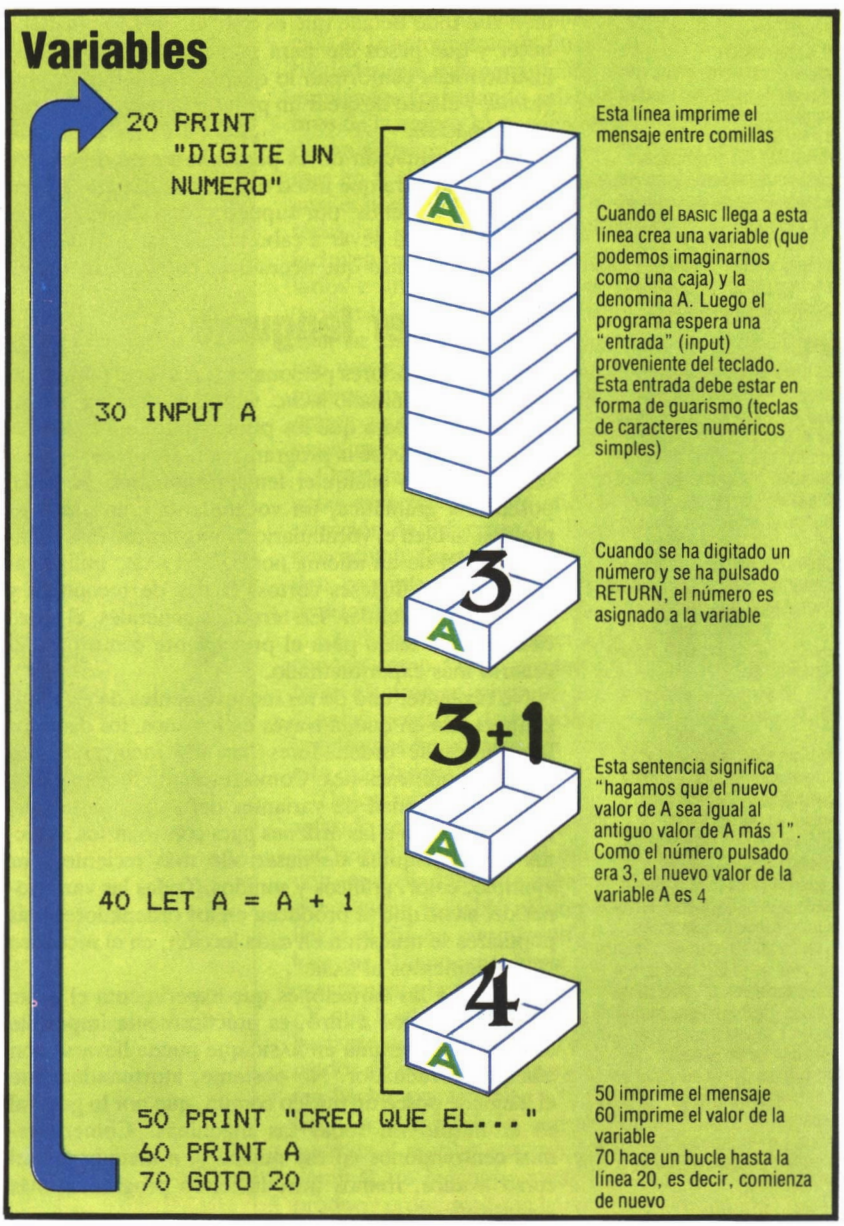
Ésta es, nuevamente, nuestra sentencia de impresión. Esta sentencia "imprime" la secuencia de caracteres (es decir, las palabras o los números que usted ha digitado) entre las comillas dobles. Observemos el punto y coma al final de la línea. Éste ayuda a especificar las posiciones en las cuales las cosas se imprimen en pantalla. Ya avanzados en el curso, retomaremos con mayor detalle el uso del punto y coma. Ahora ocupémonos de:

```
60 PRINT A
```

Aquí hay otra sentencia PRINT, pero esta vez A no está entre comillas. Ya sabemos que el programa no imprimirá en pantalla una A verdadera, porque hemos visto ya que para ello se requiere el entrecomillado. Sin las comillas, el BASIC busca una variable que posea la misma etiqueta que el carácter que sigue a PRINT. Si lo encuentra, imprime el valor de la variable. (¡De no encontrarlo, proporcionaría un mensaje de error!) Este programa ya posee una variable denominada A y por ello el BASIC imprime su valor. ¿Y cuál es ese valor?

Si usted cree que la respuesta es 7, recuerde que el BASIC trabaja con los programas línea a línea, siguiendo el orden de los números de línea. Llegados a la

El recuadro inferior muestra cómo se utilizan las "variables" en BASIC. También ilustra cómo se utiliza la sentencia GOTO (véase página siguiente) para formar un bucle



Complementos al BASIC

Sólo el Sinclair Spectrum utiliza el elemento LET de la instrucción. En otros ordenadores, éste está implícito, es decir, puede pasarse por alto. Por ejemplo, la línea 40 se podría escribir A = A + 1 en lugar de LET A = A + 1

Este elemento no se utiliza en el Spectrum. Se supone que la última línea digitada del programa es el final de éste

En el Spectrum aparece en la pantalla como dos palabras separadas (GO TO), aunque sólo se oprime una tecla. La mayoría de los otros ordenadores aceptan que la instrucción se digite en dos palabras

línea 60, el valor de A ya había sido cambiado a 8, y eso es lo que el ordenador imprimirá. Finalmente llegamos a:

```
70 END
```

La sentencia END (fin) le dice al BASIC que se ha llegado al final del programa. Algunas versiones de BASIC insisten en que todos los programas deben terminar en END, mientras que otras no (ver el recuadro "Complementos al BASIC").

Observe que cuando usted pone en funcionamiento el programa, éste sólo lo hace una vez. Para que funcione otra vez, debe volver a digitar RUN<CR>.

Veamos ahora una manera de hacer funcionar el programa todas las veces que deseemos utilizando la sentencia GOTO.

Utilizando GOTO

A continuación proporcionamos el mismo programa pero con una línea adicional. Si usted había apagado el ordenador para tomar un descanso, digítelo. De no ser así, bastará con teclear las líneas 70 y 80. Éstas aparecen en azul en el siguiente listado.

```
10 REM LOS ORDENADORES NUNCA SE
   EQUIVOCAN<CR>
20 PRINT "DIGITE UN NUMERO"<CR>
30 INPUT A<CR>
40 LET A = A + 1<CR>
50 PRINT "CREO QUE EL NUMERO
   DIGITADO ERA";<CR>
60 PRINT A<CR>
70 GOTO 20<CR>
80 END<CR>
```

Después de haberlo digitado y listado (LIST), trate de imaginarse lo que sucederá antes de intentar llevarlo (RUN). Luego pulse RUN<CR> y, al igual que en la primera versión del programa, aparecerá:

```
DIGITE UN NUMERO
```

Digite cualquier número (utilizando las teclas numéricas) y oprima RETURN. El ordenador le sumará 1 al número y lo expondrá al final del mensaje.

```
CREO QUE EL NUMERO DIGITADO ERA 8
```

Esto es seguido inmediatamente por un nuevo mensaje DIGITE UN NUMERO. Digitando otro número y volviendo a apretar RETURN, el programa repetirá este ciclo *ad infinitum*. La razón para que se produzca esta repetición la hallaremos en la línea 70:

```
70 GOTO 20
```

Cuando el BASIC llega a una sentencia GOTO, en lugar de continuar con la línea siguiente, va hacia (GOes TO) la línea cuyo número se especifica. En este caso se dirige nuevamente hasta la línea 20 y hace que todo el programa sea repetido otra vez. Y así continuaría indefinidamente realizando estos bucles. Si usted desea dejar de llevar el programa, se encontrará con que no existe manera de interrumpir estos bucles. El programa sigue adelante esperando su entrada.

Como usted comprenderá, existen maneras de escribir el programa que permiten pararlo cuando lo deseemos, y de ellas nos ocuparemos en el próximo capítulo. Mientras, deberemos detener el programa. Si el ordenador utilizado tiene una tecla BREAK, puede usarla para hacer que el programa deje de funcionar. Digitando RUN<CR> el programa comenzará nuevamente.

Observe que aún tenemos la expresión END al final del programa. Tal como hemos escrito este programa, con la sentencia GOTO 20 creando un bucle interminable, nunca llegaremos hasta el final, ¡pero algunas versiones de BASIC insisten en que utilicemos siempre un END al final!

Si no encuentra una manera de detener el programa, inténtelo oprimiendo la tecla RESET. Con ello es prácticamente seguro que el programa se detendrá. Luego pulse nuevamente LIST. Si obtiene un listado, usted podrá "editar" el programa en los ejercicios que proporcionamos a continuación. Si no obtiene un listado, ello significa que el RESET de su ordenador destruye el programa en memoria y, por lo tanto, usted deberá digitarlo completo otra vez.

Ejercicios

Estos ejercicios han sido cuidadosamente seleccionados por grados y están concebidos para que resulten amenos. Intentar resolverlos es una de las mejores maneras de verificar si usted ha comprendido el material que le hemos presentado y si efectivamente avanza en sus conocimientos de informática.

Antes de comenzar con los ejercicios, trate de modificar unas pocas líneas para ver qué efectos producen estos cambios en la forma de llevar el programa. No existe ninguna posibilidad de que el ordenador resulte dañado, aunque usted cometa errores o pulse teclas equivocadas. Para cambiar una línea, digite el programa y luego verifique el resultado haciendo el listado (LIST). En la pantalla volverá a aparecer todo el programa. Teclee el número de la línea que desea cambiar, seguido de la nueva línea. Intente con esta sentencia:

```
10 REM LOS ORDENADORES ALGUNAS VECES SE
   EQUIVOCAN<CR>
```

luego digite LIST otra vez. Observe cómo ha cambiado la primera línea. Si desea borrar toda la línea, basta con que pulse el número de línea seguido de <CR>. Pruebe:

```
10<CR>
  LIST
```

La línea 10 debería desaparecer. Ponga de nuevo la línea 10 volviendo a digitar toda la línea al completo; ¡y no se olvide del número de línea!

■ Reescriba el programa de modo que el ordenador imprima realmente el número digitado. Una pista: el truco está en quitar una línea entera.

■ Vuelva a digitar la línea 70, de modo que el programa pase a la línea 80. Haga el listado del programa (LIST). Haga funcionar el programa (RUN). ¿Por qué el programa no se ha comportado de la misma manera que antes?

■ Modifique la línea 60 para que el ordenador imprima en pantalla una A en lugar del valor de la variable A.

■ Reescriba la línea 60 de modo que el ordenador vuelva a imprimir el valor de la variable A. Quite por completo la línea 10 (la línea de REM). Digite RUN. ¿Hay alguna diferencia en la forma en que se lleva el programa?

■ Incluya una nueva observación (REM) en la línea 25. Pueden agregarse nuevas líneas con sólo digitar el nuevo número seguido de la nueva sentencia. Coloque una observación en la línea 25 para recordarle lo que ocurrirá luego; podría ser algo así como "espera una entrada proveniente del teclado". Después de digitar la nueva línea y de pulsar <CR>, vuelva a listar (LIST) el programa y verifique que su nueva observación aparezca en el lugar correcto.

■ Reescriba el programa de modo que multiplique por 10 el número que usted digite. Necesitará modificar la línea 50 para imprimir algo parecido a EL NUMERO QUE USTED DIGITO MULTIPLICADO POR 10 ES. En esta ocasión no deseamos sumar al valor de la antigua variable, sino multiplicarlo por 10. Para designar "multiplicado por", el BASIC utiliza el signo *. (No utilice una X, porque el BASIC sólo reconoce en ella a una letra y no al signo de multiplicación).

Hemos recorrido ya un gran trecho. Hemos visto cómo escribir comentarios, que para el BASIC son observaciones (REMarks), cómo imprimir (PRINT) secuencias de caracteres en la pantalla, cómo imprimir (PRINT) el valor de una variable en pantalla y cómo hacer que el programa vaya hacia (GOTO) un número de línea determinado.

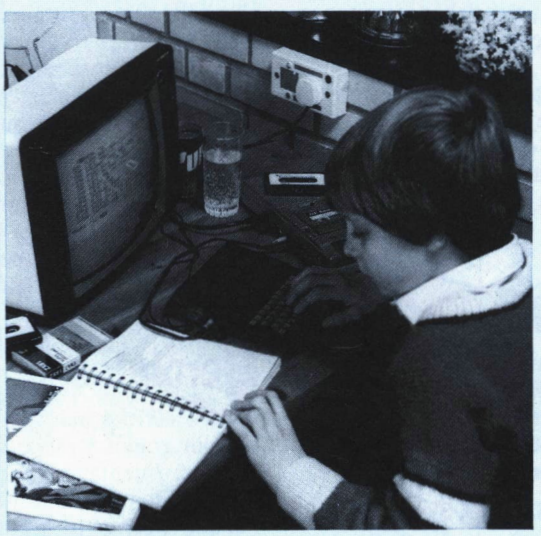
El próximo tema del que nos ocuparemos es cómo salir de un bucle utilizando una expresión IF-THEN. Descubriremos cómo hacer que el programa "realice" para nosotros un número de ciclos determinados en lugar de seguir haciendo bucles de manera indefinida.

Y entonces surgió el BASIC

En la actualidad, el BASIC es el lenguaje de programación más popular del mundo. Los lenguajes de ordenador se inventaron para que el operador humano pudiera comunicarse más fácilmente con la máquina, y el BASIC es uno de los lenguajes más sencillos de aprender y de utilizar. Consiste en una serie de instrucciones simples en inglés combinadas, cuando es necesario, con los símbolos matemáticos del teclado de una máquina de escribir.

El BASIC es un lenguaje al que se llega a dominar rápidamente. A los pocos minutos de haber desembalado un microordenador, usted puede estar escribiendo programas sencillos. Fue desarrollado en 1965 en el Dartmouth College de New Hampshire (Estados Unidos), con el único fin de simplificar los lenguajes ya existentes. Sus inventores fueron dos maestros, Thomas Kurtz y John Kemeny. El uso universal del BASIC ha supuesto la introducción en el mismo de ligeras variaciones de lenguaje. Pero el núcleo del lenguaje BASIC sigue siendo respetado por todos los fabricantes.

Un programa es una secuencia de instrucciones que el ordenador ejecuta para llevar a cabo una tarea específica. La tarea puede ser realizar una previsión financiera mensual o trasladar a un "invasor del espacio" a través de la pantalla del televisor. El programa aparece como una serie de líneas numeradas. Las órdenes se aprenden rápidamente y hasta el programa más complicado no utiliza más que combinaciones y repeticiones de las órdenes elementales.

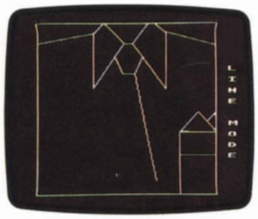


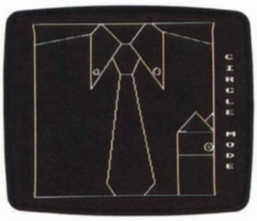
El BASIC ha desmitificado la programación y ha hecho del ordenador algo accesible a cualquier persona

La mayoría de los ordenadores salen de fábrica con el BASIC incorporado. Los ordenadores también pueden programarse en "código de lenguaje máquina" (al que se describe como un lenguaje de "bajo nivel", porque su estructura se asemeja a la lógica de los circuitos electrónicos). El BASIC es un lenguaje de "alto nivel" porque estructuralmente se aproxima al inglés cotidiano. Para aplicaciones técnicas y especializadas existen muchos otros lenguajes de alto nivel, pero el BASIC es el lenguaje idóneo para poder introducir todos los demás. Es un lenguaje sencillo y poderoso.

El artista electrónico

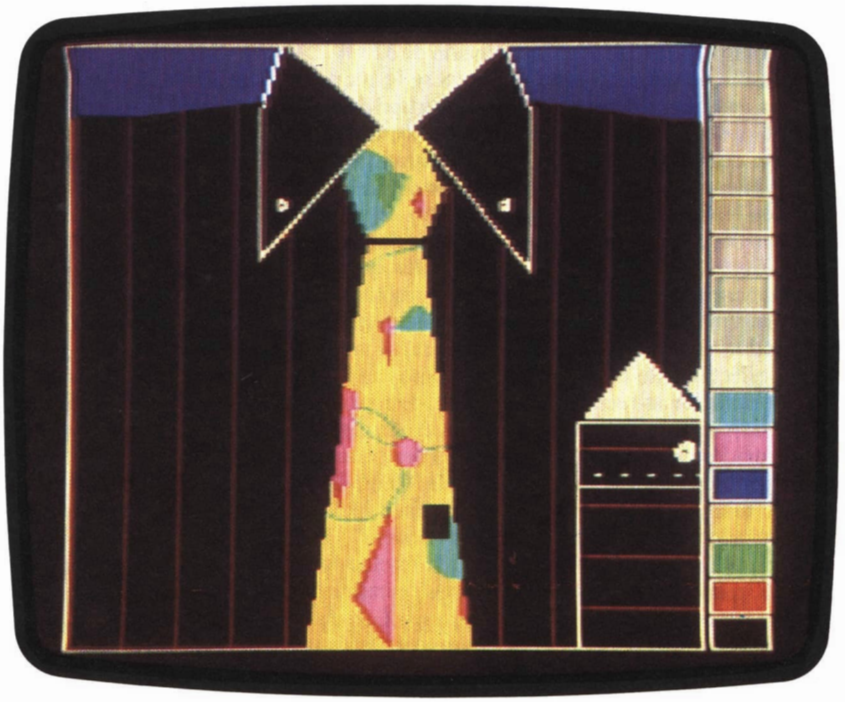
Sofisticadas o sencillas, las imágenes por ordenador se componen de miles de puntos diminutos. Y el ordenador debe recordar el color y el brillo de cada uno





Pintando con un micro

Pintar imágenes con un microordenador es sencillo si se dispone de un software para gráficos. Estos programas permiten dibujar complicadas imágenes a través de las órdenes que se digitan en el teclado. Por lo general se empieza colocando líneas rectas en la pantalla. Pueden agregarse círculos, triángulos, cuadrados y otras formas predefinidas. El color puede utilizarse tanto para las líneas como para zonas completas de la pantalla. Las zonas pueden pintarse seleccionando un color, ya sea de una paleta visualizada en pantalla o a través del teclado. La zona que queda encerrada entre sus líneas es coloreada de forma automática. En algunos programas se puede mover un "pincel" a través de la pantalla, como un cursor. La fotografía grande muestra el resultado de este proceso de "construcción"



Usted está pilotando un aeroplano en vuelo rasante sobre las azoteas de los edificios; se le ha detenido un motor; enfrente de usted se dibuja la silueta amenazante de un rascacielos... y en medio de la pista en la que pretendía desesperadamente aterrizar está ardiendo otro avión. Todo esto sucede ante sus ojos, en la pantalla del televisor. Es una forma de gráficos por ordenador.

En el caso de los programas educativos, como los destinados a que los niños aprendan a deletrear o a realizar operaciones aritméticas, es evidente que, a menos que un programa sea visualmente atractivo, no podrá mantener la atención de los pequeños durante mucho tiempo.

Es probable que quienes utilizan los microordenadores con fines empresariales traten básicamente con números que representen las cantidades de dinero que se han recibido o se han gastado, el stock de un artículo determinado, etc. Pero esta clase de información se comprende y se interpreta con más facilidad cuando se muestra en forma de imágenes. La capacidad del ordenador para "volver a dibujar" rápidamente una pintura, para incorporar nueva información, distintas alternativas, etc., y para emitir textos (impresos) cuando sea requerido en ese sentido, resulta asimismo de gran valor para algunas aplicaciones de gestión.

El lienzo del ordenador

¿Cómo crea pinturas el ordenador? Para responder a esta pregunta ocupémonos en primer lugar del "lienzo" sobre el que pinta el ordenador. Un microordenador produce imágenes en su pantalla de visualización de datos iluminando o "encendiendo" uno o varios puntos en una de diversas posiciones en la pantalla. Estos puntos están dispuestos en líneas a través de la pantalla y en columnas a lo largo de ella, de manera que la situación de cada punto viene dada por el lugar que ocupe en línea y columna. La iluminación de ciertos puntos, mientras los restantes permanecen apagados, da como resultado imágenes específicas. Esto es válido no sólo para las pantallas monocromáticas, sino también para las visualizaciones a color. En este caso las pinturas se forman al hacer que los puntos adquieran los colores adecuados.

Para visualizar una simple letra o un número, el ordenador se vale de una matriz de puntos rectangular. Esta matriz se conoce como "matriz de puntos". En un microordenador típico ésta consistirá probablemente en un bloque de ocho líneas que contengan ocho puntos cada una.

El número de puntos en la pantalla no es el mismo para todos los ordenadores, pero una cuadrícula bastante típica estaría compuesta por 192 líneas de 256 puntos cada una; o sea, 192 líneas y 256 columnas.

Es evidente que cuantos más puntos haya en la pantalla de visualización de datos del ordenador, mayor será la definición de las imágenes. El grado de definición observable al visualizar gráficos se denomina *resolución*. Se dice que un ordenador que puede visualizar 192 líneas de 256 puntos cada una, tiene una resolución de 256 × 192. Cuanto más alta sea la resolución —es decir, cuantos más puntos puedan colocarse

en pantalla— menos "granulosa" será la imagen.

Todos los microordenadores poseen una densidad máxima de puntos que pueden visualizarse en sus pantallas para gráficos, pero algunos ejemplares también pueden programarse de modo que utilicen matrices de puntos menos densas. Por ejemplo, un microordenador puede poseer una resolución máxima de 640 × 256; o sea, que le es posible alcanzar una densidad máxima de puntos dada por 256 líneas, cada una de las cuales contiene 640 puntos. Sin embargo, también puede programarse para que utilice sólo 320 de estas columnas, es decir, una resolución de 320 × 256, o incluso resoluciones inferiores si así se le requiere. En una máquina con capacidad para este tipo de variación, la resolución debe determinarse al comienzo de la parte del programa que produce los gráficos.

Las líneas y las curvas producidas por un sistema de gráficos por ordenador basado en puntos en realidad no son continuas, como lo serían trazadas con un lápiz. Son senderos de puntos iluminados que están más o menos próximos entre sí según la resolución del sistema. Un sistema de baja resolución sólo será capaz de producir curvas toscas, y una línea recta tampoco será perfectamente recta, ya que los puntos en la pantalla forman líneas rectas sólo en ciertas direcciones (a lo largo de líneas y columnas y a través de las diagonales formadas por la matriz de puntos). Para trazar una línea recta el sistema de gráficos debe iluminar los puntos más próximos al sendero de la línea indicada. El resultado puede ser un efecto "en escalera". También en este caso, cuanto más alta sea la resolución del sistema, menos evidentes resultarán estas "escaleras".

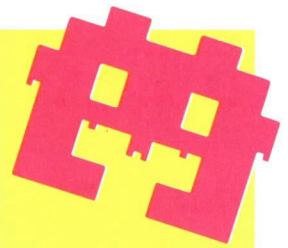
Como los gráficos se exhiben en una pantalla de televisión, para proporcionar una continuidad de visualización la imagen debe ser continuamente "refrescada" o "redibujada", porque, de lo contrario, sólo aparecería por un instante y luego desaparecería. Por este motivo, la imagen debe representarse de alguna manera en el ordenador para que, en caso necesario, éste pueda tomarla como referencia. En realidad la representación de la imagen se almacena en una zona especial de la memoria del ordenador denominada *memoria de pantalla*. En una visualización monocromática, cada punto de la pantalla corresponde a un bit de la memoria de pantalla. Una imagen se representa estableciendo que los bits que corresponden a los puntos iluminados sean 1 y dejando 0 para los bits correspondientes a los puntos no iluminados. De esta manera, si un ordenador puede mantener una visualización monocromática con una resolución de 256 × 192, debe poseer una memoria de pantalla de 256 × 192 o 49 152 bits, es decir, 6 kilobytes, ya que 1 kilobyte es igual a 8 192 bits.

Para gráficos a color se necesita más memoria. Dos bits pueden representar cuatro colores diferentes, como en el ejemplo siguiente:

bit 1	bit 2	*color*
0	0	blanco
0	1	rojo
1	0	azul
1	1	negro

Para representar cualquier imagen en cuatro colores, es preciso asignarle dos bits a cada punto de la pantalla. Del mismo modo, para imágenes en ocho colores, a cada punto de la pantalla deben asignársele tres bits, mientras que para visualizaciones en 16 colores corresponderían cuatro bits para cada punto, y así sucesivamente. Por tanto, un microordenador con capacidad para visualizar colores con una resolución de 160 × 256, debe tener una memoria de pantalla de 160 × 256 × 4, que equivale a 160 kilobits (20 kilobytes).

La necesidad de la memoria de pantalla explica, en parte, por qué algunos ordenadores están diseñados para funcionar a distintas resoluciones. Si no posee suficiente memoria de pantalla, el ordenador no puede almacenar y, en consecuencia, no puede visualizar imágenes de alta resolución.

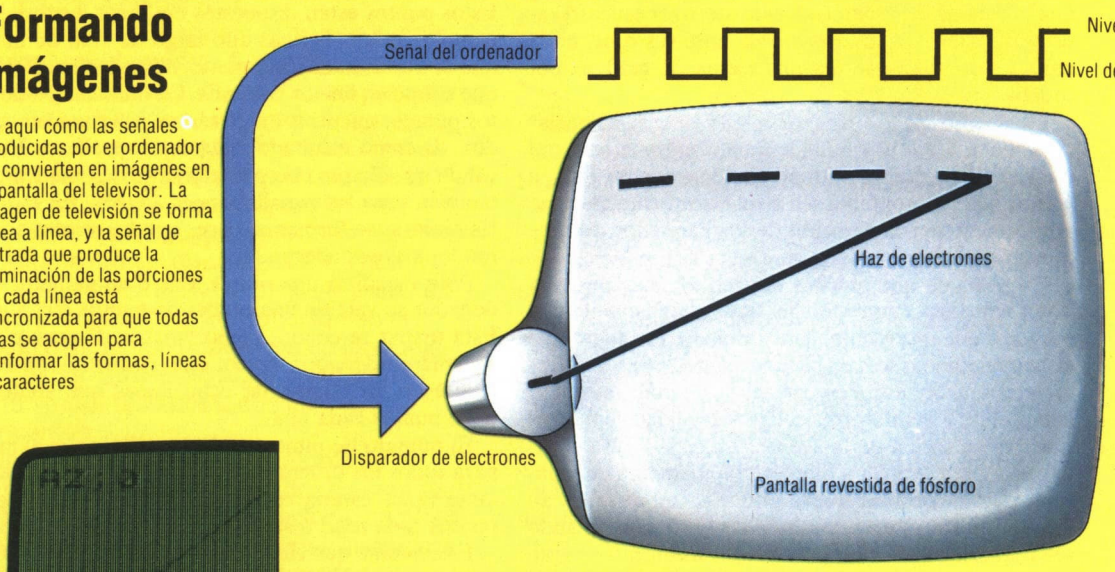


Creando gráficos

Producir imágenes con el ordenador personal puede ser sencillo. La mayoría de los modelos cuentan con órdenes especiales en BASIC para ayudar a "dibujar" o definir en la pantalla tanto a un "invasor extraterrestre" como a un auténtico cuadro. Asimismo, existe un software especial para crear dibujos animados en la pantalla.

Formando imágenes

He aquí cómo las señales producidas por el ordenador se convierten en imágenes en la pantalla del televisor. La imagen de televisión se forma línea a línea, y la señal de entrada que produce la iluminación de las porciones de cada línea está sincronizada para que todas ellas se acoplen para conformar las formas, líneas y caracteres



La señal del ordenador llega al televisor a través del conector de antena. Cuando la señal está arriba, el haz de electrones se enciende mientras explora a través de la pantalla, y se apaga cuando la señal está abajo. El haz se mueve rápidamente a lo ancho de la pantalla, luego salta y comienza ligeramente más abajo, con la línea siguiente. Para cubrir toda la pantalla se requieren cientos de líneas como ésta. Cuando el haz llega hasta la parte inferior de la pantalla, se dirige nuevamente hasta la parte superior y repite el proceso

Rizando el rizo

En esta segunda parte del curso nos ocuparemos de cómo interrumpir un bucle, cómo volver a él una determinada cantidad de veces y de la numeración de las líneas

La primera parte de nuestro curso de programación BASIC terminaba con el programa listado abajo. El programa funcionaba bien pero, en virtud del GOTO de la línea 70, efectuaba una y otra vez un bucle hasta el principio y no terminaba nunca. La única manera de salir del bucle era utilizando la tecla BREAK o la tecla RESET.

Ahora vamos a conocer uno de los modos de salir de un bucle como éste, incorporando una "comparación" en el programa. ("Comparación" es la acción de fijar una condición, para luego comprobar si existe.) La forma habitual de hacerlo consiste en probar con un número que en realidad jamás desearíamos utilizar en el programa. El programa nos permitía digitar un número que luego el ordenador imprimía en la pantalla después de sumarle 1. Podemos decidir que nunca desearíamos dar entrada a un número mayor que 999. En este caso hemos de comparar para ver si el número al cual le hemos dado entrada es mayor que 999. Digite el programa y luego agregue:

```
35 IF A > 999 THEN GOTO 80<CR>
```

Ahora ponga en marcha otra vez el programa y verá que funciona como antes; a menos, claro está, que dé entrada a un número mayor que 999. Pruebe digitando 1000<CR> y vea qué ocurre.

¿Por qué esta vez el programa se detuvo? La causa está en el IF (si) de la línea 35. Cuando el BASIC encuentra una expresión IF, sabe que a continuación viene una comparación lógica. El signo > significa "mayor que". Por lo tanto, la línea 35 significa IF (variable) A (es mayor que) 999 THEN GOTO (línea) 80 ("si la variable A es mayor que 999, luego volver hasta la línea 80"). Si usted digita 1 000, el valor de A se convierte en 1 000, que es mayor que 999, de modo que el programa "luego vuelve hasta" (THEN GOTO) la línea 80, que es el final del programa. Si A no fuera mayor que 999, la parte THEN (luego) de la línea sería ignorada y el programa continuaría con la línea siguiente.

De modo que al hacer funcionar este programa, usted puede dar entrada a todos los números que desee, con la condición de que no sean mayores que 999. Tan pronto como usted dé entrada a un número mayor que 999, la sentencia IF-THEN lo detecta y concluye el programa yendo hasta END (final). Cuando un programa BASIC llega hasta el final o se termina, se le dará a usted un aviso de "listo" en pantalla. Según sea su ordenador, este aviso tendrá diversas formas. En el Dragon, el aviso es OK. En el Sord, READY. Cualquiera que sea la forma asumida, el aviso de listo es la manera que tiene el BASIC de decirle a usted que no hay ningún programa funcionando y que está a la espera de sus órdenes.

Existen muchas variantes en la forma en que las distintas versiones de BASIC utilizan THEN. Consulte el recuadro de "Complementos al BASIC"

Otras comparaciones que se utilizan en BASIC son < (menor que), = (igual a), >= (mayor o igual a), <= (menor o igual a) y <> (distinto a). A medida que el curso avance, iremos encontrándonos con frecuencia con estas comparaciones.

Antes de seguir adelante, es conveniente realizar algunos ejercicios para acostumbrarse a utilizar estas comparaciones.

Ejercicios

■ Cambie una de las líneas de modo que el programa se interrumpa si A = 1000.

■ Cambie una de las líneas de modo que el programa se interrumpa si el número al cual le da entrada es menor que cero.

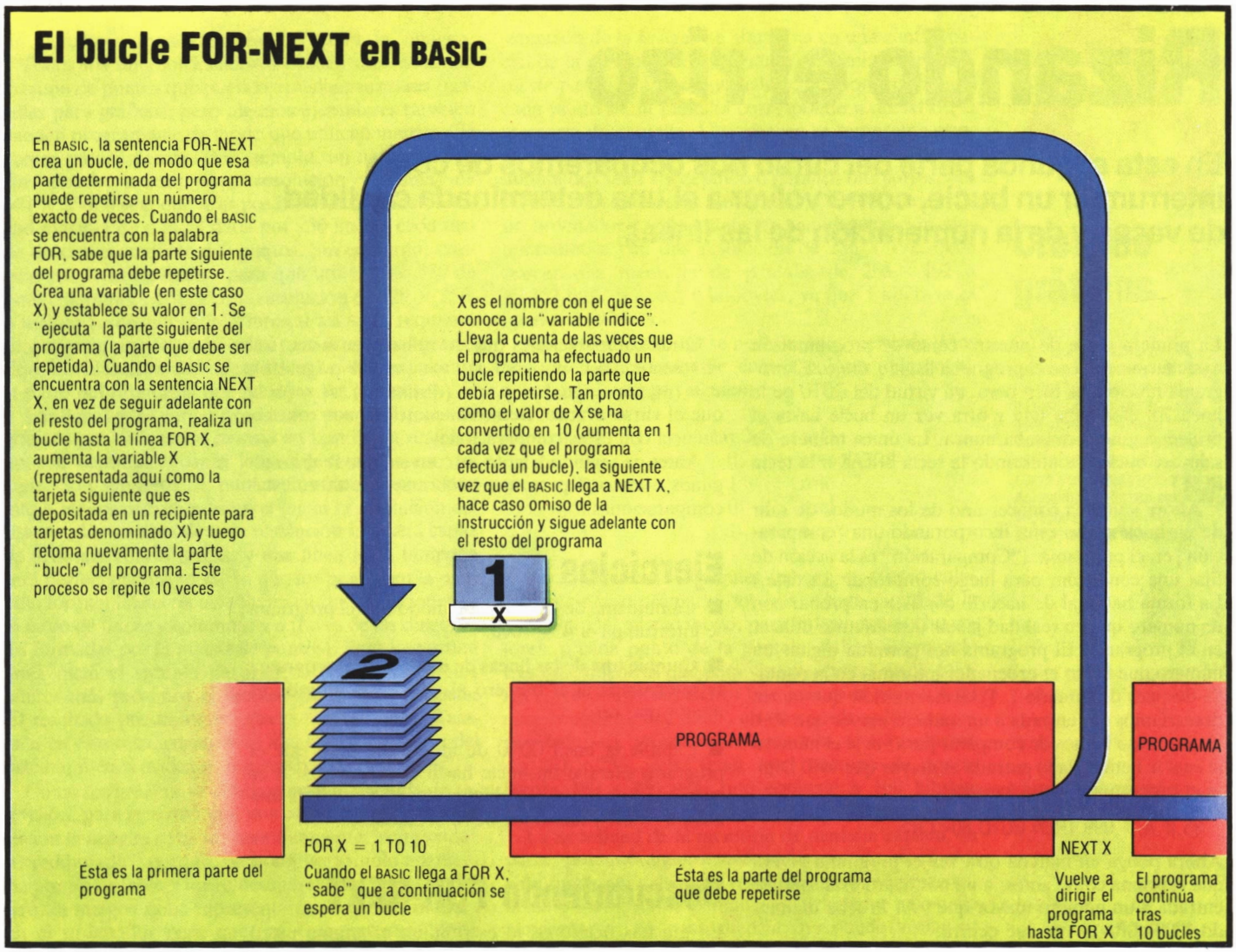
■ Cambie la línea GOTO de modo que haga que el programa efectúe un bucle hasta el principio si A es igual a o menor que 500. Una pista: no necesitará una línea IF-THEN y una línea GOTO separadas.

Descubriendo FOR-NEXT

En muchas ocasiones al escribir un programa usted deseará repetir algunos puntos del mismo un determinado número de veces. El GOTO de la línea 70 hacía que el programa efectuara un bucle todas las veces que lo deseáramos. Luego agregamos una sentencia IF-THEN en la línea 35 que nos permitía escaparnos del bucle dando entrada a un número "fuera de escala".

```
10 REM LOS ORDENADORES NUNCA SE EQUIVOCAN

20 PRINT "DIGITE UN NUMERO"

30 INPUT A

40 LET A = A + 1

50 PRINT "CREO QUE EL NUMERO DIGITADO ERA ";

60 PRINT A

70 GOTO 20

80 END
```

El bucle FOR-NEXT en BASIC



Sin embargo, tal como aprendimos en la primera parte del curso, en algunas ocasiones utilizar GOTO para hacer un bucle no es adoptar el procedimiento más acertado.

Volvamos a nuestro antiguo programa, ahora modificado para que se ajuste a la verdad, para multiplicar por 10 la entrada del número, y hagámoslo exactamente ocho veces.

```
10 REM MULTIPLICADO POR 10
20 FOR X = 1 TO 8
30 PRINT "DIGITE UN NUMERO"
40 INPUT A
50 LET A = A* 10
60 PRINT "SU NUMERO MULTIPLICADO POR 10 ES";
70 PRINT A
80 NEXT X
90 END
```

Digite este programa, lístelo (LIST) para verificar que no haya ningún error y luego hágalo funcionar (RUN). Se le solicitará un número sólo ocho veces. Luego el programa se detendrá. La causa de esta detención debemos hallarla en la línea 20.

```
20 FOR X = 1 TO 8
```

Esta línea forma parte de un bucle FOR-NEXT. Este bucle constituye una de las más útiles estructuras que ofrece el lenguaje BASIC. Merece que la estudiemos atentamente.

Tal como la hemos utilizado aquí, hemos creado una variable denominada X. (En la primera parte del curso hemos explicado qué es una variable.) Podríamos haberla llamado de cualquier manera (excepto A, que ya la estamos utilizando con otro fin). FOR siempre debe utilizarse seguido de NEXT, pero NEXT aparecerá más tarde en el programa (despues del trozo que ha de repetirse). El elemento FOR de un bucle FOR-NEXT tiene siempre la siguiente forma:

FOR variable = valor inicial TO valor final
(*para* variable = valor inicial *hasta* valor final)

En nuestro ejemplo FOR X = 1 TO 8 hemos llamado X a la variable y le hemos dado un valor inicial de 1. El ordenador ejecuta entonces la siguiente parte del programa; el número que digitamos es multiplicado por 10 y luego impreso en la pantalla. Después de esto llegamos a NEXT X y el programa hace un bucle volviendo hasta donde está la variable X, es decir, a la línea 20. Después de hacerlo aumenta X en 1, de modo que X adquiere un valor de 2. Luego se vuelve a ejecutar la parte del programa comprendida entre el bucle FOR-NEXT. Al llegar nuevamente hasta NEXT, en la línea 80, el programa vuelve a hacer un bucle y convierte a X en 3.

El programa continúa repitiéndose de esta manera hasta que X se convierte en 8. Entonces el bucle se termina; NEXT X ya no vuelve hasta FOR X y el programa continúa en la línea siguiente.

Otros usos de los bucles FOR-NEXT

Con frecuencia los bucles FOR-NEXT se utilizan para producir dilaciones en el programa. En algunas ocasiones a usted no le interesará que todo se realice a la máxima velocidad; en estos casos, puede introducir una "demora". Probablemente habrá advertido que en el programa MULTIPLICADO POR 10 las respuestas eran transmitidas de modo tan rápido que parecían instantáneas. Hagamos que el ordenador parezca tomarse un tiempo para pensar antes de dar las respuestas; para lograrlo, insertaremos una demora valiéndonos de FOR-NEXT. Agregue en su programa las líneas que aparecen en color azul.

```
10 REM MULTIPLICADO POR 10
20 FOR X = 1 TO 8
30 PRINT "DIGITE UN NUMERO"
40 INPUT A
50 LET A = A* 10
52 FOR D = 1 TO 1000
54 NEXT D
60 PRINT "SU NUMERO MULTIPLICADO POR 10 ES";
70 PRINT A
80 NEXT X
90 END
```

Hemos agregado otras dos líneas, 52 y 54, en nuestro bucle FOR-NEXT original. Ahora examinémoslas.

```
52 FOR D = 1 TO 1000
54 NEXT D
```

D se establece en 1 y el programa continúa con la línea siguiente. Ésta es la sentencia NEXT correspondiente. En realidad, en el interior del bucle no sucede nada; el programa se limita a efectuar un bucle hasta la línea 52 y aumenta D a 2. Esto ocurre 1 000 veces antes de que el programa continúe con la parte siguiente, que está imprimiendo la respuesta. Los ordenadores son veloces, pero para todo se requiere un tiempo finito, de manera que efectuar 1 000 veces un bucle lleva una cantidad de tiempo considerable. Los ordenadores difieren en cuanto al tiempo que invierten en efectuar un bucle. En el Epson HX-20, este bucle FOR-NEXT tarda 2,9 segundos, mientras que en el Spectrum tarda 4,5 segundos. Experimente cambiando el número que utiliza como límite máximo en la línea 52.

Para hacer que el ordenador se comporte más como un ser humano, agregue estas tres líneas:

```
56 PRINT "AHORA DEJEME VER..."
57 FOR E = 1 TO 1000
58 NEXT E
```

Liste (LIST) el programa y hágalo funcionar (RUN). Ahora tenemos dos demoras cuya función es sólo dilatar el tiempo del programa.

Agregue estas dos líneas:

```
51 REM ESTE BUCLE PIERDE EL TIEMPO
55 REM ESTE PIERDE AUN MAS TIEMPO
```

Ahora liste (LIST) el programa y examínelo con atención. Observe cómo todas las líneas extras que hemos agregado se han situado exactamente en los lugares precisos. Lo que nos lleva al último punto de esta parte del curso: los números de las líneas.

Comenzamos nuestro programa original con la línea 10 y continuamos saltando de 10 en 10 para cada nueva línea, terminando con la línea 90. Podríamos haber elegido otros números cualesquiera, por ejemplo 1, 2, 3... 9. Pero, de haberlo hecho, ¿cómo habríamos podido intercalar las líneas extras? A los programadores siempre se les ocurren ideas nuevas y mejores que introducir, de modo que dejemos lugar para ellas en estos grandes huecos entre los números de las líneas en las versiones "Mark I" de sus programas. Si lo desea, puede comenzar incluso por la línea 100 e ir saltando de 50 en 50 o de 100 en 100.

Algunas versiones de BASIC incluyen una orden muy útil denominada AUTO. El BASIC del Epson HX-20 la tiene. No obstante, el Dragon 32, los ordenadores Sinclair y el Commodore Vic 20 no cuentan con ella. Si su BASIC posee AUTO, puede ahorrarse mucho tiempo al hacer que los números de las líneas se generen automáticamente. Averigüe si su BASIC posee AUTO digitando:

```
AUTO 100, 10<CR>
```

Si su BASIC posee AUTO, en su pantalla se verá:

```
100
```

La pantalla muestra el número 100 seguido de un espacio y luego el *cursor*. El cursor es una marca (algunas veces una línea, otras un cuadrado) que muestra en la pantalla dónde aparecerá el carácter siguiente. Usted puede comenzar a entrar la primera línea del programa desde la posición del cursor. Cuando usted pulse <CR> la línea siguiente aparecerá automáticamente, comenzando por el número de línea 110. El AUTO, si su ordenador dispone de él, puede utilizarse tanto solo como con uno o dos "argumentos". La palabra *argumento* es un término matemático. En la expresión 2 + 3 = 5, los argumentos son 2 y 3. La orden AUTO puede utilizarse sola (por ejemplo, AUTO<CR>), o bien con un argumento (por ejemplo, AUTO 100<CR>) o con dos argumentos (por ejemplo, AUTO 300,50). AUTO empleado solo suele hacer que los números de línea comiencen a partir de 10 y sigan saltando de 10 en 10. Si se utiliza sólo un argumento (por ejemplo, AUTO 100<CR>), el primer número será 100 (en este caso) y luego los números irán saltando de 10 en 10. Si se utilizan dos argumentos, la primera cifra especifica el número de línea inicial y la segunda el número según el cual se irán produciendo los aumentos. AUTO 250,50<CR> dará 250 como número inicial, el número siguiente será 300 y así se irá aumentando, de 50 en 50. Incluso en el más sencillo de los micros, es prácticamente imposible que se dé el caso de que usted se quede sin líneas.

En el próximo apartado de este curso nos ocuparemos especialmente de las diversas maneras en que se puede mejorar la presentación visual de un programa en la pantalla y de los distintos modos de imprimir información.

Complementos al BASIC

IF

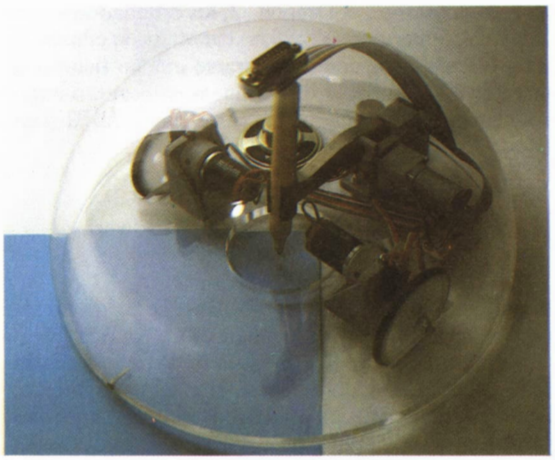
La mayoría de los microordenadores pueden utilizar esta instrucción tanto en forma de IF A>999 THEN 80, como en forma de IF A>999 GOTO 80. (El Spectrum utiliza IF A>999 THEN GOTO 80)

AUTO

No disponen de esta orden el Commodore VIC 20, el Dragon 32 y el Sinclair Spectrum

Micromundos

La mayoría de los programas educativos con ordenadores no son sino libros de texto electrónicos. El LOGO es diferente: se vale de un ordenador para crear un "entorno de aprendizaje"



Ian McKinnell

La tortuga que camina
La tortuga se diseñó como un dispositivo con el cual pensar, particularmente durante el ciclo de aprendizaje de geometría y relaciones espaciales. Cuando los niños no están seguros acerca de cómo instruir a la tortuga para que realice una maniobra determinada, tienden a asumir el papel de la tortuga y gatear por el suelo obedeciendo las instrucciones en LOGO. Esto convierte el aprendizaje en una experiencia mucho más "real"

Desde que el micro apareciera por primera vez, en 1977, los educadores con amplitud de miras comprendieron rápidamente el enorme potencial que ofrecía como medio auxiliar de enseñanza en la escuela. En la actualidad la mayoría de las escuelas de los países desarrollados poseen al menos una máquina y en muchas de ellas la informática es una materia más de estudio. A pesar de ello, la incidencia del micro en los métodos de enseñanza tradicionales ha sido ínfima.

La prueba más evidente a este respecto es la gama de programas educativos para ordenadores personales que se ofrece comúnmente, que, en líneas generales, denota una notable falta de imaginación. La mayor parte de estos programas se pueden describir como "libros de texto electrónicos", en los cuales el ordenador le presenta al alumno una serie de "cuadros" en la pantalla (equivalentes a las páginas de un libro de texto) y luego comprueba en qué medida el alumno ha asimilado la información, a través de una serie de preguntas tipo test que deben ser contestadas mediante la elección de una entre varias respuestas, que el ordenador califica automáticamente sin la intervención del profesor.

Los paquetes de programas de este tipo son muy fáciles de escribir en un ordenador personal y ofrecen la ventaja de acompañar el texto con gráficos a todo color (y, en ciertos casos, también animados). Sin embargo, esto no es más que una automatización del método tradicional, y no una forma original de aplicar el potencial del microordenador.

El lenguaje LOGO presenta una perspectiva diferente. A partir de la obra del profesor Seymour Papert, del Massachusetts Institute of Technology, el LOGO se define como "una filosofía de educación y una familia de lenguajes de programación para ordenadores concebida para ayudar a la realización de esa filosofía".

Muchas personas han considerado erróneamente al LOGO como un simple lenguaje de programación y han comparado sus órdenes y configuraciones con las del BASIC, llegando a la conclusión de que el LOGO es un lenguaje mucho mejor para los principiantes. Este razonamiento se aparta de la verdadera cuestión. Papert jamás tuvo la intención de crear un sistema destinado a enseñar al niño a programar. Él lo concibió como un entorno donde el alumno pudiera aprender diversos temas, un ambiente en el que, en realidad, pudiera descubrir una manera de aprender.

Gran parte de esta filosofía deriva del eminente filósofo suizo de la educación Jean Piaget, quien afirmó que el niño, en un entorno adecuado, era capaz de aprender cualquier tema por sí mismo del mismo modo que aprende a andar y a hablar. Sin embargo, la obra de Piaget era completamente teórica, y Papert se abocó a la tarea de crear un ambiente práctico para las teorías de aprendizaje de Piaget.

Que los métodos de educación tradicionales no cumplen este objetivo se hace evidente a partir del hecho de que la mayoría de los adultos teme aprender y no disfruta con la idea de tener que abarcar nuevas áreas de conocimiento. En su libro *Mindstorms — children, computers and powerful ideas (Confusión mental: niños, ordenadores y conceptos eficaces)*. Papert afirma que el síntoma más común de esta actitud del adulto es el miedo generalizado a las matemáticas o, como dice él, la "matematofobia".

Una de las causas de esta postura reside en que la mayor parte de estas disciplinas se imparten de la misma forma, en tanto que sus aplicaciones son totalmente diferentes. Al niño se le enseña, por ejemplo, a multiplicar en la misma forma en que aprende las capitales del mundo: mecánicamente. El proceso de aprendizaje se divorcia de lo que se está aprendiendo, cuando ambas cosas deberían ser inseparables.

Para el propio Papert, el aprendizaje de un conocimiento nuevo, ya sea volar, cocinar o aprender un idioma extranjero, es como practicar una afición. Él atribuye esta actitud suya a su infancia, cuando descubrió, a muy temprana edad, cómo funcionaban las ruedas dentadas y aplicaba este concepto cada vez que se enfrentaba a un nuevo problema. También Albert Einstein solía decir que cuando se encontraba con algo que no comprendía, lo descomponía en conceptos que había aprendido antes de los cinco años.

Estas eficaces ideas se incorporaron al LOGO, como se puede ver en nuestro ejemplo del LOGO en acción. La primera característica importante de este lenguaje es la tortuga, que fue diseñada como "un dispositivo con el que pensar" del mismo modo en que Papert usaba las ruedas dentadas cuando era un niño. Para los niños pequeños, la tortuga asume la forma de un robot móvil diseñado especialmente, que está conectado a un micro y que se puede desplazar por el suelo digitando órdenes en LOGO. Normalmente la tortuga lleva un lápiz para dibujar formas en el suelo y también puede estar dotada de un pequeño altavoz y de detectores de colisión para guiarla a través de un camino con obstáculos.

Los niños, por lo general, luego de emplear las tortugas móviles, pasan a utilizar las tortugas de pantalla, formas que se pueden mover a través de la pantalla del

Aprendiendo la curva

Este ejemplo ficticio pero típico muestra cómo el LOGO estimula a un grupo de niños a resolver problemas con los cuales nunca antes se habían encontrado.

```
TO CURVE
REPEAT 80
  FORWARD 1
  RIGHT 1
END
CURVE
```

—Si queremos pétalos necesitamos dibujar una curva.
—Pero la tortuga siempre se mueve en línea recta.
—¿Y qué pasaría si la obligáramos a desplazarse una distancia corta, luego la hiciéramos girar sólo un poquito, luego otra vez en línea recta, y así varias veces? Eso sería como una curva.
—De acuerdo; la distancia más pequeña es uno, y el ángulo más pequeño es uno. Hagámosla hacer eso ochenta veces.

```
CURVE
CURVE
```

—¡Eh!, eso es justo lo que queríamos.
—Dos de ellas harán un pétalo; probemos.

```
TO PETAL
CURVE
RIGHT 90
CURVE
END
PETAL
```

—Eso no tiene nada que ver con un pétalo. ¿Qué pasó?
—Que siguió a partir de la última curva: le deberíamos haber dicho que fuera en otra dirección.
—Pero, ¿cuánto la hacemos girar?
—Probemos con noventa; eso suele dar resultado.
—Y hagamos una palabra nueva: PETAL (pétalo), para ahorrar tiempo.

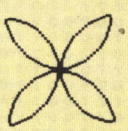
```
TO PETAL
CURVE
RIGHT 100
CURVE
END
PETAL
```

—Eso está mejor, pero noventa grados no era suficiente. ¿Qué hacemos ahora?
—Pensemos algo y veamos si funciona, en vez de hacer suposiciones.
—Sí; recuerda que aprendimos que si la tortuga gira hacia la derecha alrededor de algo gira a través de un total de trescientos sesenta grados.
—Bien, sabemos que gira ochenta en la primera curva; entonces tiene que girar ochenta en el camino de vuelta, o sea ciento sesenta grados.
—Dejando doscientos para girar en la punta del pétalo.
—No, porque para volver a la posición desde donde empezó, necesitará girar también en la otra punta.
—Entonces deberíamos probar con la mitad de doscientos.

```
PETAL
PETAL
PETAL
PETAL
```

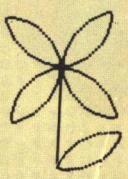
—Magnífico: con cuatro como éste haremos una flor.

```
TO FLOWER
REPEAT 4
  PETAL
  RIGHT 10
END
FLOWER
```



—No salió muy bien que digamos; está dispareja.
—Nos olvidamos de poner algunos giros entre los pétalos.
—Pero entonces, ¿por qué no los dibujó uno encima del otro?
—Porque después de dibujar un pétalo, la tortuga queda mirando cien a la izquierda de donde empezó originalmente.
—De modo que cada pétalo gira cien hacia la izquierda cada vez.
—Eso está bastante bien: lo que necesitamos es noventa, así que agreguemos un giro de diez hacia la derecha entre cada pétalo.

```
FLOWER
RIGHT 180
FORWARD 100
RIGHT 180
PETAL
```



—¡Al fin! Gira la tortuga para que podamos dibujar el tallo.
—Cien debería ser suficiente.
—Pongamos una hoja en la punta; puede ser igual a un pétalo.
—Pero esta vez recuerda el ángulo: la tortuga ha de dar vuelta hacia la derecha.

Liz Dixon

ordenador. La tortuga es un eficaz dispositivo con el cual los niños pueden aprender los conceptos básicos de las relaciones espaciales, que les permitirán luego iniciarse en la geometría avanzada.

El control de la tortuga es, sin embargo, sólo una pequeña aplicación del LOGO, pero es el aspecto más promocionado a nivel publicitario porque visualmente es el más interesante. Más importante es el concepto de establecer, a partir de ideas sencillas, ideas más sofisticadas y, a la inversa, descomponer grandes problemas en problemas más pequeños del tipo de los que ya se han asimilado previamente.

Estos procesos se pueden ver con toda claridad a través de la conversación imaginaria entre un grupo de niños que están aprendiendo a instruir a la tortuga para que dibuje una flor (véase recuadro). Comienzan con sólo tres órdenes disponibles: FORWARD (adelante), que hace que la tortuga se mueva hacia adelante hasta una distancia calculada; RIGHT (derecha), que obliga a la tortuga a describir un ángulo determinado, y, por último, REPEAT (repetir), que repite una cierta cantidad de veces las líneas que hay instrumentadas en el programa.

A partir de estas ideas fundamentales los niños construyen primero una "herramienta" (un programa) para dibujar una curva (TO CURVE... END). Toda esta secuencia se puede producir ahora simplemente digitando CURVE. Siguiendo el mismo proceso, después de experimentar y de un aprendizaje posterior, se define una orden PETAL (pétalo), que se vale de la orden CURVE. Finalmente se desarrolla una orden FLOWER (flor), que dibujará la imagen completa.

El LOGO no es el único lenguaje que incorpora este tipo de estructuras, pero es el único diseñado para que lo utilicen niños pequeños. Prescinde de muchas de las formalidades y procedimientos asociados con la programación en otros lenguajes. De hecho, se trataba de que el niño no fuera consciente de estar programando un ordenador, sino sólo de estar resolviendo un problema.

En algunas situaciones de aprendizaje, el alumno no se confunde ni siquiera a este nivel. El maestro establece una serie de eficaces herramientas utilizando el LOGO, todas las cuales se refieren a un tema o área de conocimiento determinados. Al niño se le permite luego explorar el tema empleando las herramientas y descubrirlo por sí mismo. Estas áreas se denominan "micromundo": entornos limitados en los cuales se utiliza el ordenador para imitar algo del mundo real o alguna área de conocimiento.

Probablemente el mejor ejemplo de lo que es un micromundo sea el modelo en LOGO de la física de Newton. Aunque la primera ley de Newton dice que sin la influencia de fuerzas externas un cuerpo se seguirá moviendo en línea recta a una velocidad constante, las mentes de los pequeños observan que en el mundo real todo va moviéndose más despacio. Esto determina que el aprendizaje se bloquee. No obstante, utilizando el LOGO se puede construir un micromundo en el que todo se comporte de acuerdo con las leyes de Newton y, con la ayuda de herramientas con las cuales se empujan los objetos a través de la pantalla, los niños aprenden rápidamente y por sí mismos las tres leyes de Newton.

El LOGO es un concepto poderoso que vale la pena aprender mediante un ordenador personal. Las tortugas móviles no son baratas. Están comenzando a salir al mercado versiones de LOGO que emplean tortugas de pantalla para diversos ordenadores personales.

Términos clave

Bits y bytes

El ordenador sólo comprende números. Pero son números de apariencia un tanto extraña

Siempre que se escribe acerca de ordenadores, se utilizan las palabras *bits* y *bytes*. Estos vocablos describen la manera en que los ordenadores almacenan y emplean los números.

Y lo hacen de un modo bastante diferente al nuestro. Nosotros representamos los números mediante 10 símbolos diferentes (de 0 a 9) y los manejamos en múltiplos de 10. (Esto se conoce como una "base" de 10.) Los ordenadores, por el contrario, para llevar a cabo toda su magia matemática se sirven sólo de dos números: el cero y el uno. Los bits y los bytes son formas de representar combinaciones entre estos dos números.

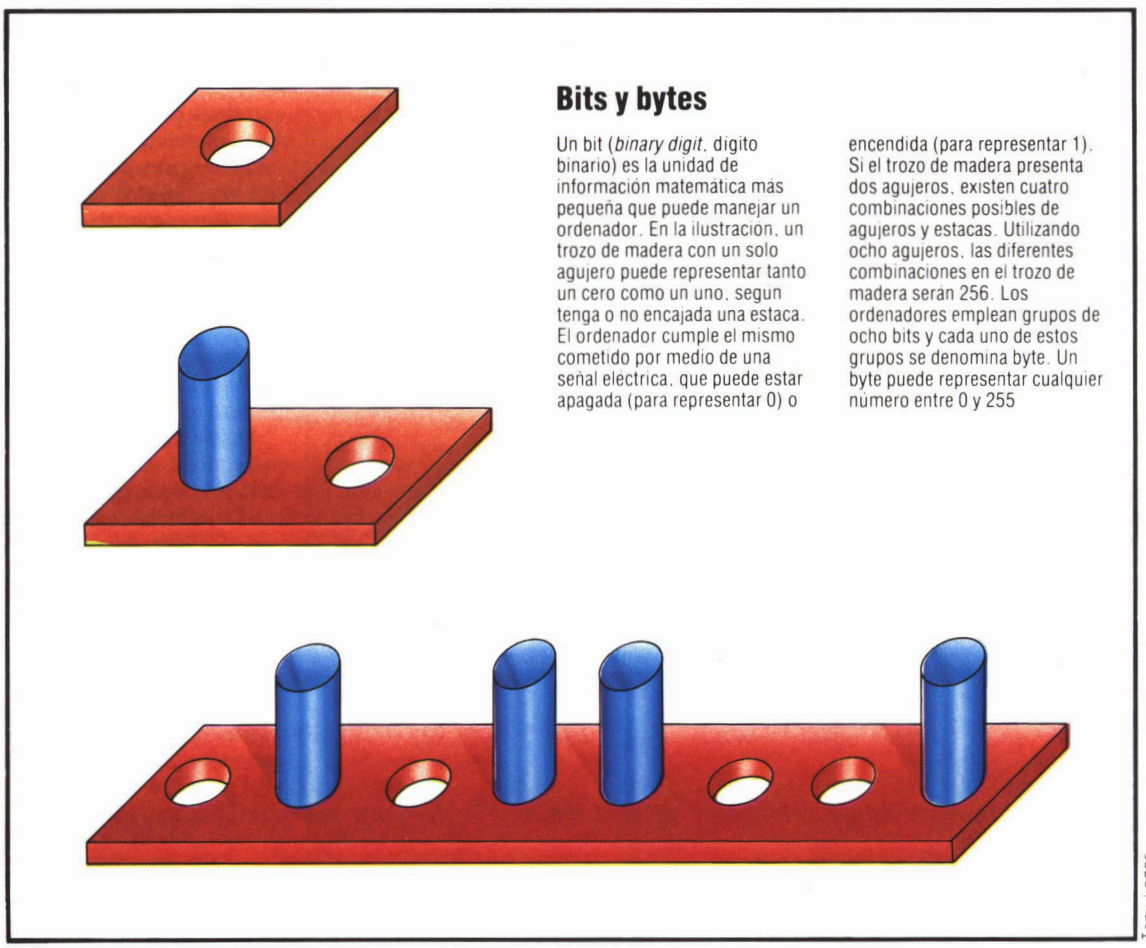
Un bit es la unidad de información más pequeña que puede manejar un ordenador. Es la forma que tiene el ordenador de representar los números cero y uno. Un grupo de ocho bits se denomina byte; un byte permite al ordenador representar cifras muy elevadas.

Ocupémonos en primer lugar de los bits: qué son y por qué se denominan así. Los ordenadores son dispositivos electrónicos y, en consecuencia, todas sus funciones las realizan, en última instancia, con señales eléctricas. Una señal eléctrica puede estar "encendida" o "apagada"; éste es el principio por el cual las señales eléctricas pueden representar números.

La ilustración muestra un trozo de madera que presenta un agujero en el cual puede encajarse una estaca. Aun cuando encaje en un solo agujero, la estaca puede representar dos números y constituye una analogía excelente para describir la forma en que funciona un ordenador. El agujero puede no estar ocupado por una estaca, en cuyo caso representa un cero, o bien estar ocupado por ella, en cuyo caso representa un uno. El mismo trozo de madera puede, por tanto, simbolizar tanto un cero como un uno.

En el ordenador se consigue el mismo efecto con una señal eléctrica: si está apagada representa un cero; si está encendida, un uno. Un cable, o un trozo de madera con un agujero, puede, pues, utilizarse para representar dos condiciones: sin estaca o con estaca, ausente o presente, encendido o apagado, 0 o 1.

Esta unidad mínima de información se denomina *bit*. La palabra bit denota su reducido tamaño y representa dos posibles situaciones; deriva de **BI***nary digi***T**. Considerado desde otra perspectiva, un bit



Bits y bytes

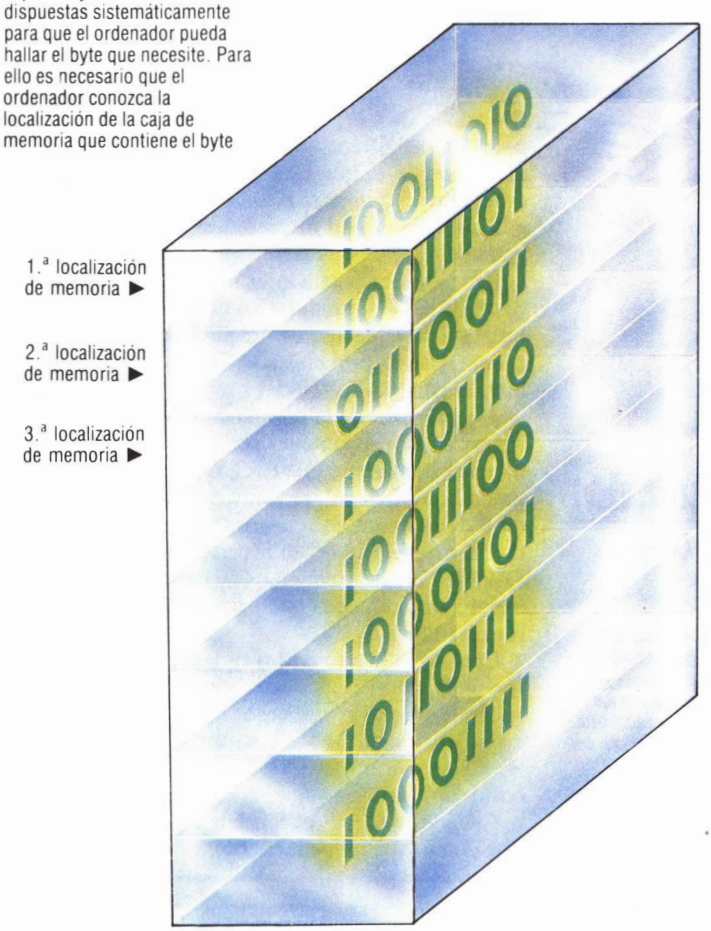
Un bit (*binary digit*, dígito binario) es la unidad de información matemática más pequeña que puede manejar un ordenador. En la ilustración, un trozo de madera con un solo agujero puede representar tanto un cero como un uno, según tenga o no encajada una estaca. El ordenador cumple el mismo cometido por medio de una señal eléctrica, que puede estar apagada (para representar 0) o encendida (para representar 1). Si el trozo de madera presenta dos agujeros, existen cuatro combinaciones posibles de agujeros y estacas. Utilizando ocho agujeros, las diferentes combinaciones en el trozo de madera serán 256. Los ordenadores emplean grupos de ocho bits y cada uno de estos grupos se denomina byte. Un byte puede representar cualquier número entre 0 y 255

Tony Loodge

0	00000000	128	10000000
1	00000001	129	10000001
2	00000010	130	10000010
3	00000011	131	10000011
4	00000100	132	10000100
5	00000101	133	10000101
6	00000110	134	10000110
7	00000111	135	10000111
8	00001000	136	10001000
9	00001001	137	10001001
10	00001010	138	10001010
11	00001011	139	10001011
12	00001100	140	10001100
13	00001101	141	10001101
14	00001110	142	10001110
15	00001111	143	10001111
16	00010000	144	10010000
17	00010001	145	10010001
18	00010010	146	10010010
19	00010011	147	10010011
20	00010100	148	10010100
21	00010101	149	10010101
22	00010110	150	10010110
23	00010111	151	10010111
24	00011000	152	10011000
25	00011001	153	10011001
26	00011010	154	10011010
27	00011011	155	10011011
28	00011100	156	10011100
29	00011101	157	10011101
30	00011110	158	10011110
31	00011111	159	10011111
32	00100000	160	10100000
33	00100001	161	10100001
34	00100010	162	10100010
35	00100011	163	10100011
36	00100100	164	10100100
37	00100101	165	10100101
38	00100110	166	10100110
39	00100111	167	10100111
40	00101000	168	10101000
41	00101001	169	10101001
42	00101010	170	10101010
43	00101011	171	10101011
44	00101100	172	10101100
45	00101101	173	10101101
46	00101110	174	10101110
47	00101111	175	10101111
48	00110000	176	10110000
49	00110001	177	10110001
50	00110010	178	10110010
51	00110011	179	10110011
52	00110100	180	10110100
53	00110101	181	10110101
54	00110110	182	10110110
55	00110111	183	10110111
56	00111000	184	10111000
57	00111001	185	10111001
58	00111010	186	10111010
59	00111011	187	10111011
60	00111100	188	10111100
61	00111101	189	10111101
62	00111110	190	10111110
63	00111111	191	10111111
64	01000000	192	11000000
65	01000001	193	11000001
66	01000010	194	11000010
67	01000011	195	11000011
68	01000100	196	11000100
69	01000101	197	11000101
70	01000110	198	11000110
71	01000111	199	11000111
72	01001000	200	11001000
73	01001001	201	11001001
74	01001010	202	11001010
75	01001011	203	11001011
76	01001100	204	11001100
77	01001101	205	11001101
78	01001110	206	11001110
79	01001111	207	11001111
80	01010000	208	11010000
81	01010001	209	11010001
82	01010010	210	11010010
83	01010011	211	11010011
84	01010100	212	11010100
85	01010101	213	11010101
86	01010110	214	11010110
87	01010111	215	11010111
88	01011000	216	11011000
89	01011001	217	11011001
90	01011010	218	11011010
91	01011011	219	11011011
92	01011100	220	11011100
93	01011101	221	11011101
94	01011110	222	11011110
95	01011111	223	11011111
96	01100000	224	11100000
97	01100001	225	11100001
98	01100010	226	11100010
99	01100011	227	11100011
100	01100100	228	11100100
101	01100101	229	11100101
102	01100110	230	11100110
103	01100111	231	11100111
104	01101000	232	11101000
105	01101001	233	11101001
106	01101010	234	11101010
107	01101011	235	11101011
108	01101100	236	11101100
109	01101101	237	11101101
110	01101110	238	11101110
111	01101111	239	11101111
112	01110000	240	11110000
113	01110001	241	11110001
114	01110010	242	11110010
115	01110011	243	11110011
116	01110100	244	11110100
117	01110101	245	11110101
118	01110110	246	11110110
119	01110111	247	11110111
120	01111000	248	11111000
121	01111001	249	11111001
122	01111010	250	11111010
123	01111011	251	11111011
124	01111100	252	11111100
125	01111101	253	11111101
126	01111110	254	11111110
127	01111111	255	11111111

Bytes en memoria

Los bytes son grupos de ocho dígitos binarios (bits). El ordenador utiliza los bytes para almacenar números entre 0 y 255. Cada byte se almacena en una "celda" de memoria separada y estas celdas están dispuestas sistemáticamente para que el ordenador pueda hallar el byte que necesite. Para ello es necesario que el ordenador conozca la localización de la caja de memoria que contiene el byte



Tony Lodge

puede servir para contar, pero sólo de cero a uno.

Un trozo de madera con dos agujeros puede simbolizar cuatro casos diversos, o contar de cero a tres. Ambos agujeros pueden estar vacíos; el agujero derecho puede estar ocupado por una estaca; el agujero izquierdo puede tener una estaca, o bien ambos agujeros pueden estar ocupados. La parte inferior de la ilustración de la página contigua muestra un trozo de madera con ocho agujeros. En este caso existen 256 permutaciones posibles entre agujeros y estacas, y éstas están reflejadas en la tabla: los unos representan a las estacas y los ceros a los agujeros.

Este grupo formado por ocho dígitos binarios (bits) se denomina *byte*. Un byte puede, pues, representar 256 situaciones diferentes (y contar de 0 hasta 255).

Cuando decimos que un ordenador "almacena" un byte, significa que su memoria conserva un número (entre 0 y 255) que utilizará cuando sea requerido en ese sentido. Cada byte posee su propia "caja" (dirección de memoria) y estas "cajas" se disponen en un orden preestablecido (el gráfico superior ilustra cómo están apiladas una encima de la otra). Cuando el ordenador necesita recuperar un número, sólo ha de saber en qué caja (dirección) está almacenado el byte.

Todos los números entre 0 y 255 pueden representarse mediante singulares combinaciones de unos y ceros (tabla a la izquierda). Los bits son almacenados y utilizados por el ordenador en grupos de ocho. Cada grupo de ocho bits se denomina byte

Toma de contacto

Al principio parece que todos los teclados son iguales; sin embargo, unos son mejores que otros y trabajan de forma diferente

El teclado de un ordenador constituye una parte importante del sistema del mismo. Después de todo, es el medio por el que usted puede comunicarse con el ordenador. La importancia del teclado es equiparable a la de la capacidad de memoria, o a la calidad de los gráficos.

En los microordenadores se ha adoptado el teclado tipo QWERTY, similar a los de las máquinas de escribir; este tipo de teclado recibe dicha denominación porque las seis primeras letras de la línea superior forman la palabra QWERTY.

A comienzos de la década de los cincuenta, cuando los ordenadores comenzaron a utilizarse con fines comerciales, el diseño QWERTY, el sistema de mecanografía convencional, se convirtió en el dispositivo de entrada estándar para los ordenadores. En la actualidad el propietario de un ordenador debe someterse al QWERTY, un sistema muy fácil para un mecanógrafo experimentado, pero que para el recién iniciado suele resultar difícil.

Cuando el precio de los ordenadores ascendía a cifras muy altas, el coste de un teclado mecánico era insignificante. Sin embargo, los sucesivos desarrollos en la tecnología de los microprocesadores redujeron notablemente el coste de los componentes electrónicos de los mismos.

Cuando apareció el Sinclair ZX81, un teclado del tipo de una máquina de escribir podía representar una parte significativa del coste de fabricación de un microordenador. El teclado móvil mecánico, en el que se basa el Dragon, por ejemplo, tiene auténticos interruptores debajo de las teclas (véase la ilustración de la página siguiente). Cuando se pulsa la tecla, los contactos internos se cierran para completar un circuito. Los interruptores de este tipo contienen numerosos componentes y aumentan considerablemente el coste del teclado.

La solución a este problema la constituye un nuevo tipo de teclado, más económico. El teclado "sensible al tacto" del ZX81 se desarrolló a partir de la idea de que la mayoría de las personas que adquieran un microordenador estarían interesadas principalmente en utilizarlo para juegos y para escribir pequeños programas.

Estas aplicaciones implican una actividad mínima con el teclado, por lo cual parecía lógico que los posibles usuarios de un microordenador estuviesen preparados para decidirse por un teclado de inferior calidad. Aunque se prescinda de las ventajas de un teclado convencional del tipo de máquina de escribir, se obtiene, en cambio, un ahorro considerable.

El ZX81 fue diseñado con un teclado sensible al tacto, eliminando la mayoría de las piezas. Este recurso propició el abaratamiento del modelo, pero no representó una solución definitiva. El teclado sensible al tacto ofrece el inconveniente de que no proporciona mucha "realimentación táctil" (o sea, usted nunca está seguro de que la tecla pulsada ha sido registrada en el ordenador, a menos que lo compruebe mirando la pantalla).

En su siguiente producto (el Spectrum), la Sinclair introdujo el teclado de membrana (véase diagrama). Este tipo de teclado representa una nueva mejora, pero todavía carece de la realimentación táctil de que goza el teclado de tipo máquina de escribir.

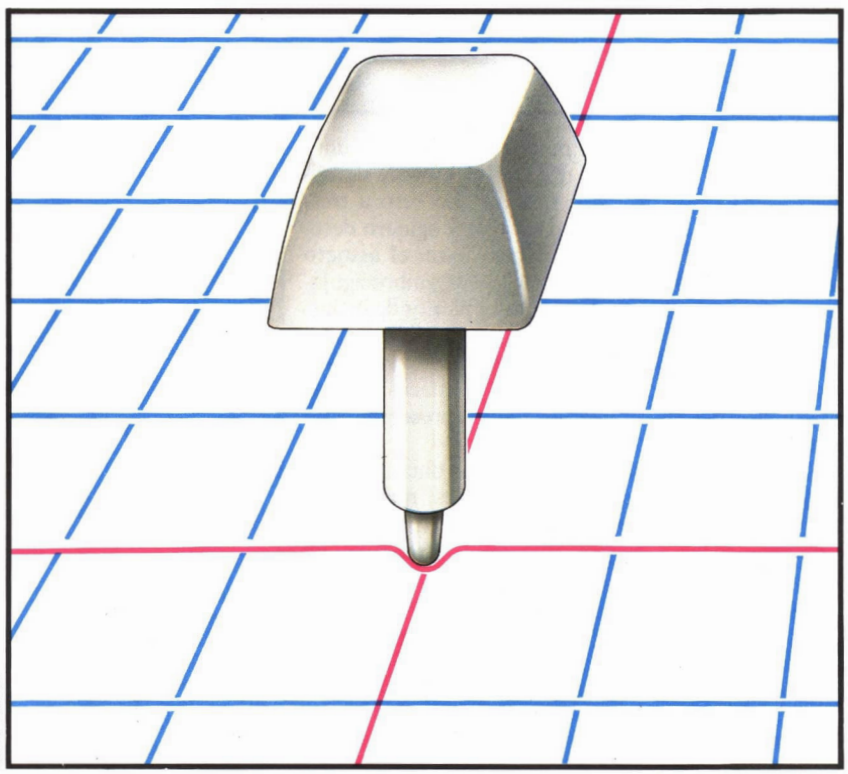
Algunos ordenadores relativamente baratos (entre los que se incluye el Dragon) poseen teclados de máquina de escribir "profesional". Las ventajas de los teclados de este tipo se ponen claramente de manifiesto cuando el ordenador es sometido a una intensa utilización para tratamiento de textos. La familiar sensación de que se está trabajando con una máquina de escribir permite que la tarea se efectúe con muchísima rapidez.

Entre el teclado completamente móvil y el de tipo membrana del Spectrum, existe otro tipo de teclados, que a menudo son denominados "teclados similares al de las máquinas de calcular" (por ejemplo, los que incorpora el New Brain y el Oric-1). Las teclas proporcionan una "sensación" mejor, pero son pequeñas, rígidas y resultan menos aptas para la escritura al tacto que las teclas completamente móviles del tipo de máquina de escribir.

Una forma de superar parcialmente la falta de realimentación táctil de los teclados sensibles al tacto y de

La matriz del teclado

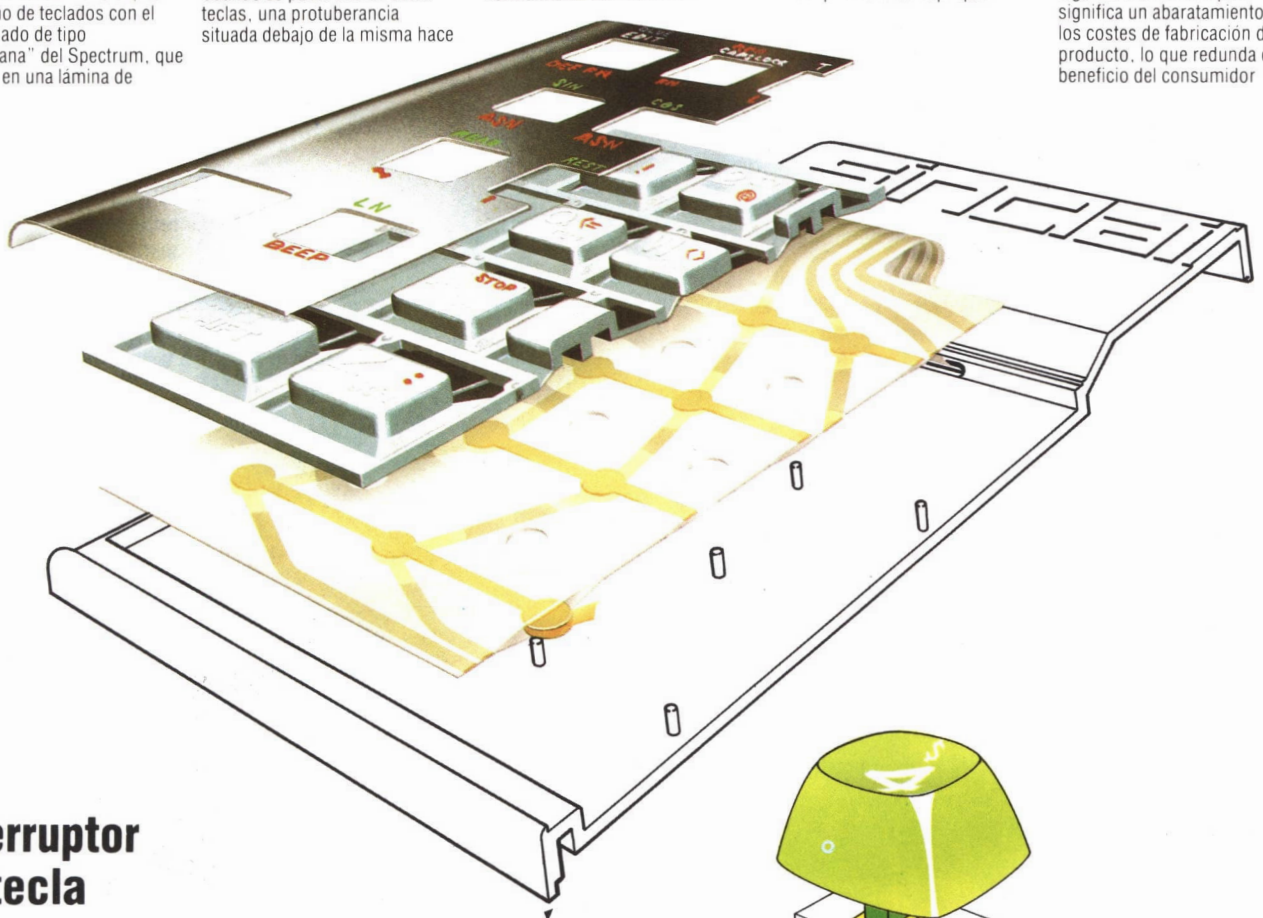
Las teclas de un ordenador son, en realidad, interruptores que están conectados a una rejilla de cables. En la ilustración vemos cómo, cuando es pulsada una tecla, se conectan dos de los cables de la rejilla. A cada tecla corresponde únicamente un par de cables, por lo que sólo efectúa una conexión en la rejilla, permitiendo que el ordenador descubra cuál es la tecla que ha sido pulsada



David Mallot

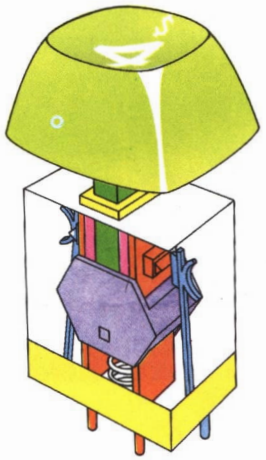
El teclado Sinclair

La Sinclair revolucionó el arte del diseño de teclados con el denominado de tipo "membrana" del Spectrum, que consiste en una lámina de caucho moldeado, con protuberancias en forma de teclas, montada sobre un relleno de contactos que conforman la matriz o la rejilla del teclado. Cuando se pulsa una de estas teclas, una protuberancia situada debajo de la misma hace que los contactos se unan. El ordenador verifica qué contactos se han cerrado y puede deducir a qué tecla corresponden. Los contactos cerrados por la tecla normalmente se mantienen separados mediante una burbuja de aire atrapada entre dos láminas de plástico. La fuerza que hace saltar las teclas devolviéndolas a su posición inicial proviene de la propia elasticidad del caucho, que se estira cuando la tecla es pulsada. Este enfoque original a la ingeniería de teclados ciertamente ha eliminado algunos elementos, pero significa un abaratamiento en los costes de fabricación del producto, lo que redunda en beneficio del consumidor



Interruptor de tecla

Los interruptores de teclas del tipo de máquinas de escribir incorporan, por lo general, un par de contactos. Estos normalmente están separados y no permiten que pase corriente eléctrica. Cuando se pulsa la tecla, una pieza de plástico moldeado (en color malva en la ilustración) se mueve hacia abajo y permite que los contactos se acerquen y cierren un circuito. Un muelle interior hace que la tecla vuelva a su posición. Cuando se cierran los contactos se produce un flujo de corriente que es detectado por el ordenador. Los cables conectados a los contactos de cada interruptor están dispuestos en una rejilla. Para saber cuál es la tecla que ha sido pulsada, el ordenador verifica cuáles son los cables "verticales" de la rejilla y cuáles los "horizontales" que conducen corriente. Las teclas de este tipo son muy complejas desde el punto de vista mecánico y los costes de fabricación son más elevados. Ofrecen una gran fiabilidad y proporcionan una sensación más "positiva" que las teclas de membrana plástica. Esta sensación táctil proviene de la resistencia que ofrece el muelle. Una tecla bien diseñada proporciona tal realimentación táctil que el usuario sabe si ha pulsado la tecla de forma correcta. Las cabezas de las teclas también están modeladas para que la digitación resulte más cómoda. Los teclados de este tipo son los más indicados si el ordenador ha de utilizarse de forma intensiva

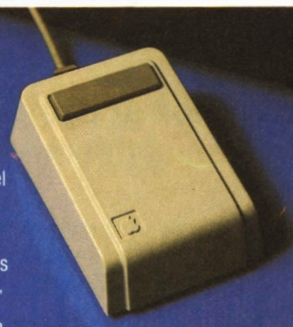


David Weeks

los del tipo membrana, consiste en emitir un "beep" (sonido corto y agudo) cada vez que se pulsa una tecla. Mediante dicha señal sonora, el usuario puede advertir que la tecla ha sido pulsada y reconocida por el ordenador.

Los diseñadores del Sinclair ZX81 y del Spectrum introdujeron una forma inédita y muy útil de reducir la cantidad de pulsaciones mecanográficas que se precisan para la entrada de programas BASIC. Cada tecla representa algo más que una letra del alfabeto o un simple número. Mediante la utilización de una tecla de "función" especial junto a las teclas normales, puede conseguirse que aparezcan en pantalla palabras completas en BASIC sin necesidad de digitarlas letra por letra. Por ejemplo, la palabra PRINT en BASIC puede ser elaborada simplemente pulsando la tecla de función especial y la tecla de la letra P simultáneamente. En su modelo M5, la firma Sord desarrolla una idea similar.

El fin de la digitación



Hasta no hace mucho tiempo, el único modo de lograr que un ordenador realizara alguna operación consistía en introducir en él las instrucciones digitándolas en el teclado. Esto, con frecuencia, resultaba un trabajo tedioso, que empeoraba más si cabe en el caso de que el usuario no fuera un mecanógrafo diestro y veloz. Comprendiendo que para muchas personas esta dificultad suponía una barrera que las disuadía de utilizar (y, desde luego, de adquirir) los ordenadores, los fabricantes hallaron una solución tan sencilla como brillante: el "ratón" (similar al mando de bola). El "ratón" puede desplazarse a través de cualquier superficie plana y, a medida que lo hace, el cursor se mueve asimismo a través de la pantalla de visualización de datos. De esta manera, uno puede moverse rápidamente hasta cualquier sector de la pantalla, y, pulsando el botón, se inicia la operación que se desea. Los "ratones" también pueden utilizarse para realizar gráficos, trazar líneas o rellenar con colores la pantalla

La revolución del ordenador

Ordenadores en la granja
Una empresa dedicada a la realización de software ha diseñado un programa llamado "Optimizador", que minimiza el coste del pienso para una granja porcina. La proporción de cereales varía a diario en función del tipo de alimentación. Usando el ordenador, el granjero da las cantidades necesarias diarias a los cerdos y el valor del nutriente de los cereales disponibles en términos de proteínas, energía y vitaminas. El programa determina entonces la proporción de cereales más económica. Este software ayuda al granjero a calcular las mezclas de piensos más eficientemente. Persisten, no obstante, unos ciertos problemas de "gusto". Los animales prefieren una dieta regular, y son reacios a tomar una comida completamente distinta a la anterior. El programa, a pesar de todo, ha obtenido un gran éxito mundial y ha sido vendido desde Thailandia a México

El ordenador puede ahorrar tiempo y esfuerzo a las pequeñas empresas y multiplicar, al mismo tiempo, sus beneficios

El ordenador nació en los laboratorios de las universidades y centros militares. Las primeras máquinas se construyeron para el cálculo de las trayectorias de los proyectiles disparados, por ejemplo, desde un acorazado en medio de una tempestad, y para la predicción meteorológica.

No obstante, no tardaría mucho tiempo en ser evidente la utilidad de los ordenadores en aplicaciones comerciales. Inicialmente, sólo las grandes compañías podían disponer del capital necesario para su mecanización. Pero la revolución de la microelectrónica a finales de los años setenta puso el ordenador al alcance de la pequeña empresa.

¿Cómo puede un ingenio creado para fines militares ayudar, por ejemplo, a una papelería, dedicada a la venta de objetos de escritorio, periódicos y revistas? El ordenador puede admitir y almacenar gran cantidad de datos. Asimismo, puede reorganizar la información que le ha sido proporcionada de forma más útil. En una papelería-librería suele haber grandes existencias de diversos artículos, y su control resulta una tarea harto pesada.

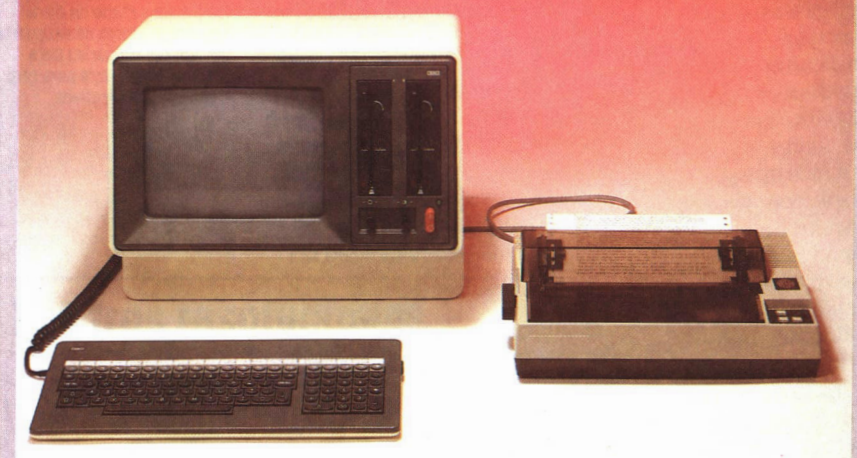
Las cifras de ventas son introducidas en el ordenador y un programa controla el stock de cada artículo. En el momento en que las existencias de cualquier artículo son menores que el stock mínimo, el ordenador genera un mensaje para que se realice el pedido correspondiente. La memoria del ordenador puede efectuar también un pedido estándar. Entonces, el vendedor añade los detalles particulares de ese artículo, y el ordenador imprime la hoja de pedidos.

Pero el ordenador no se detiene ahí. Las largas y tediosas tareas de cálculo de nóminas, impuestos, confección de cuentas de fin de ejercicio, etc., caen dentro del ámbito de su aplicación.

Los ordenadores no son sólo prácticos, sino a veces totalmente indispensables. Sin su ayuda, el hombre nunca hubiera puesto el pie en la Luna. Existen problemas en el diseño de cohetes y en la navegación que sólo pueden ser resueltos mediante ordenadores. Gracias a ellos, la humanidad tiene acceso a campos de la ciencia y la técnica que antes le estaban vedados. Pero para el propietario de un pequeño negocio, que se pregunta qué puede hacer un ordenador que no pueda efectuar él personalmente, la respuesta es sencilla: dinero. Actividades que hasta ahora precisaban más dinero en organización y administración del que producían, son rentables con la llegada del microordenador.

Supongamos que al principio la orientación de este negocio era la venta de periódicos y revistas. Pero el escaso margen de beneficio obtenido llevó a su propietario a comerciar en otros artículos. Ahora el ordenador puede reportarle beneficios al mantener un control de las demandas diarias de cada cliente. Los clientes pueden cambiar de periódico cada día sin trastocar el sistema de pedidos. A la distribuidora se le solicita el número exacto de periódicos o revistas necesarios. A fin de mes, el ordenador calcula la cuenta de cada cliente e imprime la factura adecuada. Gracias al ordenador, muchos negocios vuelven a ser rentables.

El sistema para la pequeña empresa



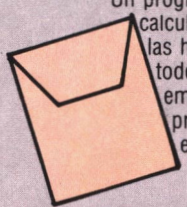
Un ordenador apropiado para una pequeña empresa debe ser una máquina de gran calidad, capaz de soportar muchas horas de uso diario. Es esencial un buen teclado y una unidad de disco para realizar los programas de oficina. El ordenador NCR ilustrado aquí es un sistema típico. El único gasto accesorio requerido es el coste de la impresora

Contabilidad

Todas las cuentas de la empresa pueden realizarse con un software especial de contabilidad. El diario de ventas, los libros de caja o de doble entrada pueden automatizarse para producir las cuentas mensuales, trimestrales o anuales en un formato adecuado para contables o auditores.

El mejor software de contabilidad comprende "módulos" que trabajan a la vez, de forma que los nuevos datos suministrados por una sección de este software (las salidas de caja, por ejemplo) actualizan el resto de las secciones del sistema de contabilidad. La ventaja para el empresario se traduce en pocas horas-hombre dedicadas a asentamientos rutinarios en los libros, cuentas más exactas y altos beneficios al final de la jornada

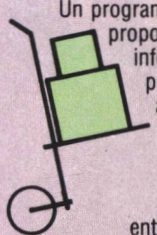
Nómina



Un programa de nómina calcula el salario e imprime las hojas de nómina de todos los empleados de la empresa. Estos programas deben tener en cuenta las horas trabajadas, el salario base, las deducciones típicas, incluyendo impuestos y seguridad social, etc.

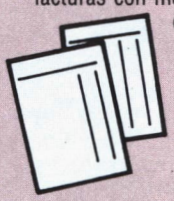
Dichos programas funcionan normalmente haciendo el operador una serie de preguntas sencillas: ¿Cuántos empleados hay en la nómina?, ¿Cómo se pagan los salarios, semanal o mensualmente?, etc. Estas preguntas se contestan digitando la información en el teclado del ordenador

Control de stocks

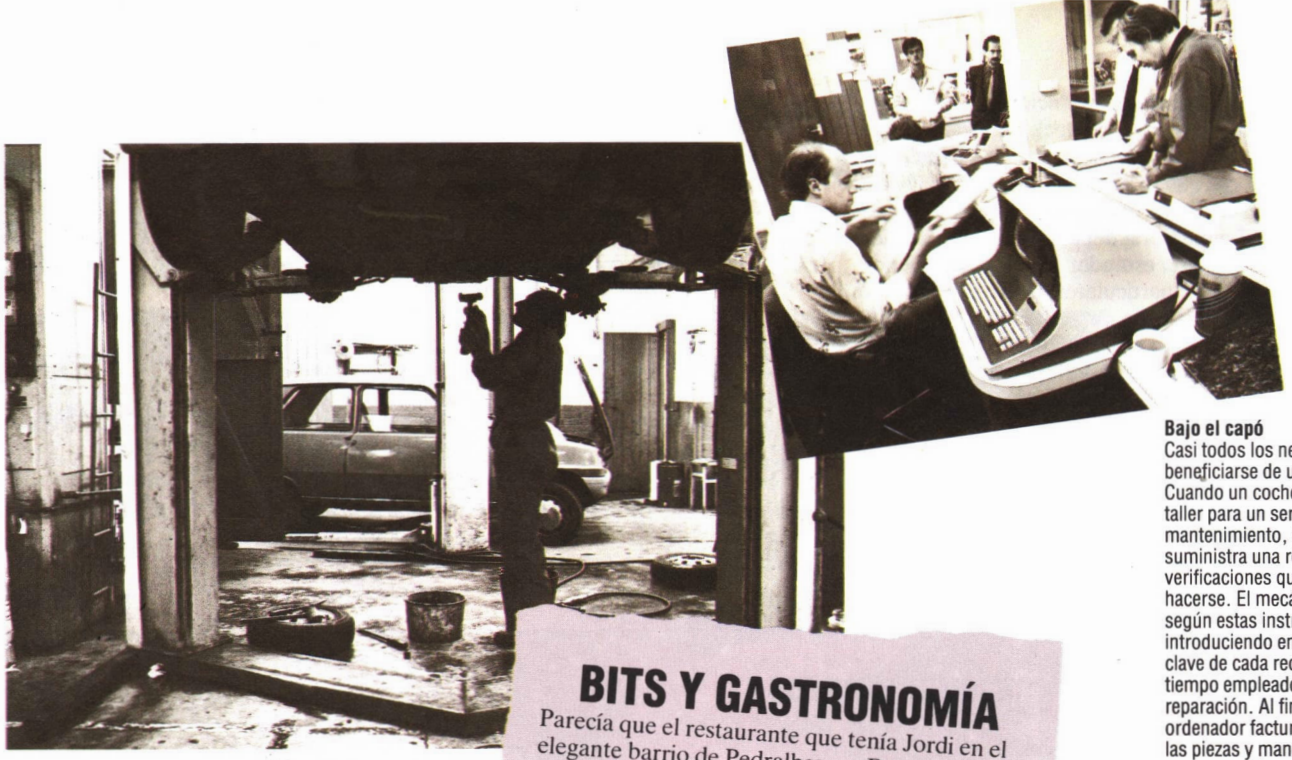


Un programa de control de stocks proporciona al operador del sistema información sobre su variación. El programa puede, incluso, efectuar automáticamente el pedido de los artículos cuyo nivel de existencias esté por debajo del mínimo. Algunos programas pueden obtener los datos de entrada mediante un lector de barras óptico en el momento en que es vendido el artículo, de forma que el precio puede ser visualizado en la caja registradora y al mismo tiempo actualizarse el inventario

Facturación



Un programa de facturación imprime las facturas con menos problemas y más exactitud que si se realiza de forma mecánica. Al hacer funcionar un programa de este tipo, se le pregunta al operador una serie de cuestiones, y éste, mediante el teclado, introduce las respuestas apropiadas. El programa consulta los registros para comprobar si la factura es legítima y los detalles son adecuados. El programa también intercambia datos con el control de stocks para mantener actualizadas las existencias



Tony Sleep

Bajo el capó
Casi todos los negocios pueden beneficiarse de un ordenador. Cuando un coche entra en el taller para un servicio de mantenimiento, el ordenador suministra una relación de las verificaciones que deben hacerse. El mecánico trabaja según estas instrucciones, introduciendo en el ordenador la clave de cada recambio y el tiempo empleado en cada reparación. Al final, el ordenador facturará el valor de las piezas y mano de obra. Los ordenadores se ocupan de las labores rutinarias para dejarle tiempo libre para otros trabajos más creativos

BITS Y GASTRONOMÍA

Parecía que el restaurante que tenía Jordi en el elegante barrio de Pedralbes, en Barcelona, tenía bastante éxito; normalmente se encontraba muy concurrido. No obstante, el dueño se hallaba disconforme con los gastos generales y sobresueldos, que había que restar de la recaudación.

Un ordenador le proporcionó algunas respuestas. Recogiendo y despachando los pedidos y expidiendo detalladas cuentas para los clientes, la máquina minimizó los errores.

El propietario quiso conservar las gratificaciones al personal, y esto requirió un análisis de productividad. Por este motivo, decidió adquirir un ordenador y software que incluyera un programa de nómina y otro de control de existencias y de coste de las diversas recetas.

EN LA PELUQUERÍA

En una elegante peluquería de Barcelona, un microordenador formula a la cliente algunas preguntas de tipo personal. La mujer es interrogada acerca de sus gustos o preferencias sobre el acondicionador, tratamiento de color, permanente, etc.

De este modo, el peluquero puede trabajar a partir de las precisas instrucciones del ordenador, que especifica un producto determinado y cómo debe ser usado. La cliente recibe un programa personal para el tratamiento de su cabello, de acuerdo con las respuestas proporcionadas al ordenador.

Los programas de diagnóstico son simples de operar y tardan pocos minutos en aparecer. El dueño de la peluquería opina que, gracias a la ayuda del ordenador, puede realizar teñidos de cabello de alta calidad, y del gusto de su numerosa clientela.

Existen muchos otros paquetes de software para empresas. Los programas de tratamiento de textos están entre los más solicitados. Todas las correcciones realizadas mediante un programa de tratamiento de textos pueden efectuarse en una pantalla. Cuando el texto ha sido completado, se imprime una copia perfecta tantas veces como sea necesario. Aunque la repetición sea aburrida para una persona, no lo es para un ordenador.

El tratamiento de textos es un área muy apropiada para el ordenador. Con los avances tecnológicos, cada vez serán más las personas que dispondrán de un ordenador en la oficina o en su casa. La comunicación directa entre ordenadores reemplazará finalmente al envío por correo de facturas o pedidos. Los ordenadores serán en el futuro tan indispensables como lo es hoy el teléfono.

PAN Y ORDENADOR

Un día una familia decidió instalar una panadería en el barrio de Salamanca, en Madrid. Al poco tiempo de abrir el negocio, los propietarios se dieron cuenta de la disparidad de los pedidos: 20 bollos por aquí, 400 barras por allá, y de que esta contabilidad podría llevarla muy bien un ordenador.

Ahora, la flamante máquina puede incluso facturar semanalmente, hacer la nómina y calcular las remesas, el coste de la harina y demás ingredientes, descuentos, etc.

Introducir 350 pedidos en la memoria del ordenador fue una labor ardua. Pero una vez esta tarea estuvo terminada, el microordenador pudo trabajar solo durante todos los fines de semana, facilitando sobremanera la administración del negocio.

Pintando con números

Tres tipos fundamentales de gráficos de ordenador y cómo usarlos para crear imágenes

Un cuadro se compone de cientos de pinceladas aplicadas por el pintor sobre un lienzo desnudo. ¿Pero cómo puede un artista crear una imagen usando un ordenador?

Para las realizaciones gráficas, se emplean tres sistemas principales. El diseño gráfico debe tener en cuenta el grado de control sobre la resolución, o nitidez, de la imagen final. Cada microordenador utiliza gráficos de bloque, *pixels* o gráficos de alta resolución.

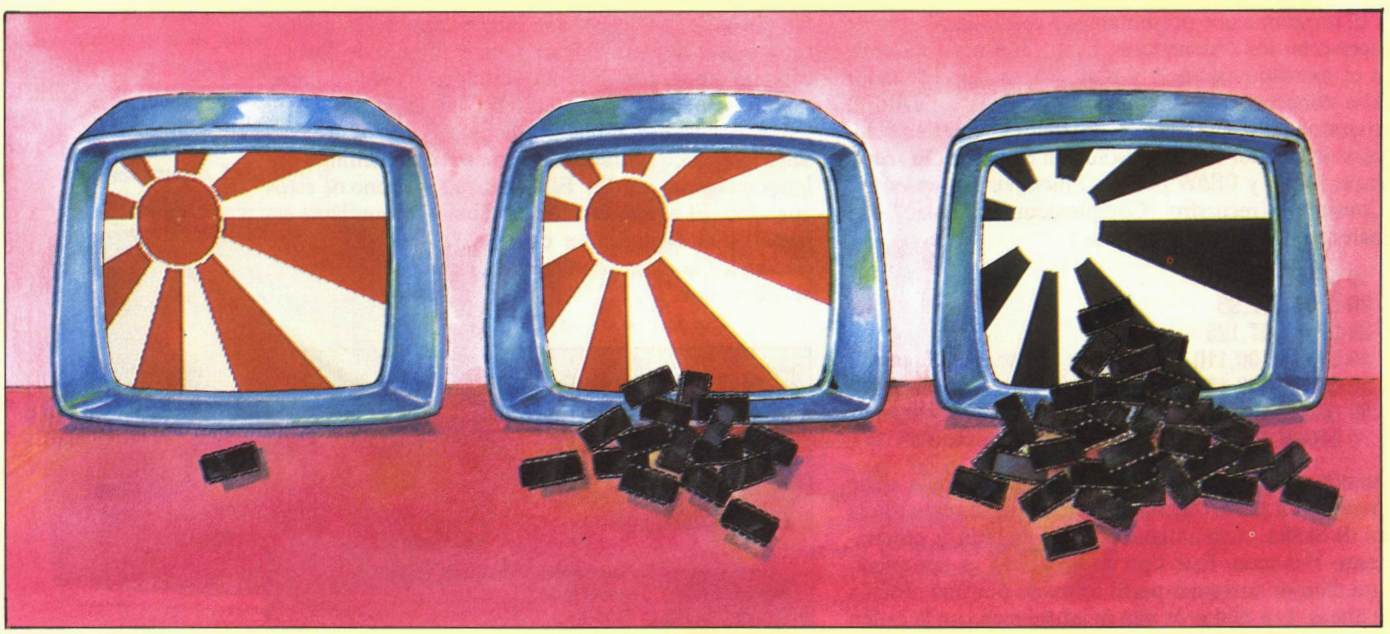
En los gráficos de alta resolución, el artista desea controlar los puntos luminosos del monitor. La única limitación es la capacidad de memoria del ordenador. La memoria modela la pantalla del televisor. Con un ordenador de 32 K de memoria, cada punto luminoso corresponde a una parte de la pantalla según el modelo del aparato.

En los gráficos de bloque, lo que el artista pierde en control sobre los puntos individuales que configuran la pantalla, lo gana en comodidad. Las formas elementales están ya programadas y se pueden obtener a través del software para construir una imagen. Se controlan directamente con el teclado y por lo general están situadas en la parte frontal de cada tecla. De esta forma, el teclado del ordenador se convierte en una paleta de pintor.

Cada forma está configurada dentro de una pequeña matriz de puntos: ocho filas por ocho columnas. Algunos microordenadores ofrecen además la posibilidad de definir los propios caracteres. Para definir el nuevo carácter y añadirlo a la gama del ordenador, se emplea un programa menor.

Los *pixels* (contracción de *picture cell*: unidad pictórica) se sitúan entre los gráficos de bloque y los de alta resolución. Éstos permiten al artista controlar cada pixel, que contiene más de un punto luminoso, pero

Las tres resoluciones



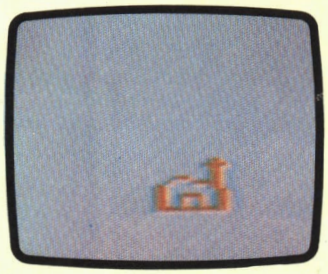
Los gráficos de bloque de baja resolución, de sólo 1 Kbyte de memoria, son suficientes para almacenar todos los detalles que deben aparecer en la pantalla. Si únicamente hay 40 bloques en cada fila y sólo 24 filas en la pantalla, el número total de bloques es 960, algo menos de un Kbyte (un Kbyte es igual a 1 024 bytes). Por tanto, un Kbyte de memoria para la pantalla puede almacenar ocho bits de información (255 combinaciones diferentes) para cada uno de los bloques de la pantalla.

En resolución media, el detalle que se obtendrá en la pantalla estará en función de la capacidad de la memoria disponible. Ocho Kbytes de RAM representan 65 536 puntos luminosos. Si el aparato es en color, una parte de la memoria se empleará para especificar el color de cada punto, reduciéndose la memoria para el número de puntos. Si se pueden reproducir ocho colores, se necesitarán tres bits para determinar el color de cada punto.

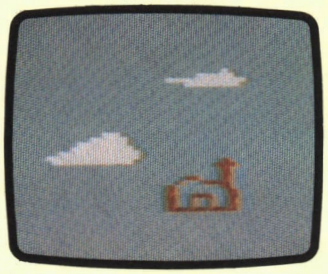
Un aparato de alta resolución requiere una gran cantidad de memoria. A veces, los aparatos de alta resolución tienen hasta 640 puntos por línea y la pantalla está formada por 240 líneas. Esto da un total de 268 800 puntos. Si sólo se va a representar un color, serán necesarios 33 600 bytes (268 800 dividido por 8). Como se ha dicho anteriormente, cada bit equivale a un cero o a un uno, y esto corresponde a que cada punto en la pantalla esté apagado o iluminado

Jo Lawrence Ian McKinnell

Gráficos sprite



Sprites es el nombre que recibe la forma de lograr imágenes más naturales, y más fácilmente, en un ordenador. Originalmente desarrollados por Texas Instruments, los gráficos sprite pueden conseguirse en la actualidad en la mayoría de los ordenadores personales, incluidos el Commodore 64, los modelos Atari y el Sord M5. En los gráficos convencionales, las



imágenes se componen en una pantalla simple, como si se dibujara en una hoja de papel. Con los sprites el ordenador tiene diversos planos o "capas", cada uno de los cuales contiene su propia imagen. En algunos ordenadores, como el Sord M5, pueden hacerse hasta 32 planos separados.
La manera más fácil de representar estos planos consiste en imaginarse

láminas de plástico transparente. Si la lámina "más cercana" al observador contiene el dibujo de un árbol, mientras que la última incluye la imagen de una nube, ésta será vista pasar por detrás del árbol como si flotara en el cielo.
Al poner diversos elementos de la pintura en planos separados, podrán crearse efectos tridimensionales muy convincentes.

Los sistemas gráficos sprite tienen también muchas otras ventajas. La imagen "dibujada" en cada uno de los planos se llama, en el lenguaje de los ordenadores, *objeto*. Al crear un objeto (un pájaro, por ejemplo), el programador puede olvidar los detalles de cómo fue compuesto. Si quiere desplazarlo por la pantalla, puede especificar una velocidad y una dirección.

menos de ocho por ocho bloques. Cada pixel se puede hacer surgir individualmente y situar en la pantalla en la posición deseada.

En los gráficos de alta resolución se utilizan dos órdenes BASIC para realizar líneas: MOVE, para situar el inicio de una línea, y DRAW, para hacerla aparecer. Cada final de línea se identifica con un par de números, que representan la fila y la columna de la pantalla donde termina la línea. Normalmente, en la pantalla las filas se enumeran de arriba abajo, y las columnas, de izquierda a derecha. Por tanto, el punto situado en la fila cero, columna cero se encuentra en la esquina izquierda inferior de la pantalla.

El programa que presentamos a continuación puede hacerse en los ordenadores Lynx (aún no disponible en el mercado español) y BBC Micro. El ordenador debe conectarse en la función "gráficos". Digite el programa tal como aparece. Puede funcionar en cualquier ordenador de alta resolución con tan sólo reemplazar MOVE y DRAW por sus funciones equivalentes. (Consulte el recuadro "Complementos al BASIC" que aparece en esta misma página.)

```
10 MOVE 100,50
20 DRAW 92,95
30 DRAW 57,125
40 DRAW 100,110
50 DRAW 143,125
60 DRAW 108,95
70 DRAW 100,50
```

Cuando el programa funciona, en la pantalla debe aparecer un perfil "Trinacria". Este perfil recuerda la isla de Sicilia, cuyo antiguo nombre en latín es precisamente *Trinacria*. Este tipo de programa es apropiado para dibujar cualquier perfil formado por una retícula de líneas. Los números que se introducen con la orden DRAW indican la posición en la fila y la columna de cada punto que forma el perfil. Con estas funciones, se puede realizar cualquier tipo de dibujo. La única limitación es la resolución de la pantalla del ordenador y la perseverancia que tenga hasta el extremo de que incluso una curva puede representarse con puntos. Y las órdenes MOVE y DRAW proporcionan el acceso a tales puntos.

El siguiente programa realiza una imagen de un cono formado por una serie de círculos de sección variable. Tal como está escrito, este programa funcionará en el Sinclair Spectrum. Otros ordenadores, como el Dragon y el Oric, poseen además la función CIRCLE. (Consulte nuevamente el recuadro "Complementos al Basic".)

```
10 FOR K = 2 TO 40
20 CIRCLE 40 + K, 40 + K,K
30 NEXT K
```

Estos dos programas muestran cómo un ordenador puede realizar gráficos mediante funciones numéricas. No obstante, una digitación intermitente y desordenada conduce siempre al fracaso en el intento de llevar a cabo un dibujo continuo. Por esta razón existe equipamiento especial para el diseño de imágenes. Éste puede conectarse al microordenador y así no es necesario introducir los miles de números que permiten obtener imágenes nítidas. El digitalizador es uno de estos aparatos. El artista efectúa su dibujo con un lápiz especial, y el digitalizador convierte el movimiento de éste en los números de filas y columnas que interpreta el ordenador.

Complementos al BASIC

PROGRAMA 1

Para introducir el programa en el BBC Micro, debe precederse de
5 MODE 0

PROGRAMA 2

Para introducir el programa en el Oric, debe precederse de
5 HIRES
y la línea 20 debe reemplazarse por
20 CURSET 40 + K, 40 + K, 0
25 CIRCLE K, 1

Para introducir el programa en el Dragon, debe precederse de
5 PMODE 4,1: SCREEN 1,1: COLOR 0,5:
PCLS
y la línea 20 debe reemplazarse por
20 CIRCLE (40 + K, 40 + K), K

Puntuación

Por qué al escribir un programa de ordenador se debe prestar atención a cada detalle de puntuación

Quizá haya advertido, en la relación del programa de la primera parte de programación BASIC, que al final de la línea 50 hay un punto y coma. La función de este signo de puntuación en BASIC no se explicó en ese momento; sin embargo, es muy importante. El punto y coma se emplea en casi todas las versiones de BASIC para indicar que las próximas PRINT continuarán imprimiendo en la misma línea y al lado del último carácter impreso. Las líneas 50 y 60 de la página 20 eran:

```
50 PRINT "CREO QUE EL NUMERO DIGITADO ERA";
60 PRINT A
```

La línea 50 imprime las palabras entre comillas. La línea 60 imprime el valor de la variable A. El poner el punto y coma hace que el valor de la variable A sea impreso directamente tras las palabras entrecomilladas de la línea 50. Si no se utilizara el punto y coma, el valor de la variable A sería impreso en la línea siguiente a la de las palabras.

El programa que aparece a continuación está concebido para ilustrar algunas de las propiedades útiles del punto y coma, tal como se emplea en BASIC. Intente escribir el programa y hacerlo funcionar. A partir de ahora, omitiremos la advertencia <CR> al final de cada línea para indicar que debe apretar la tecla RETURN. Este programa le permite introducir una gama de temperaturas en grados centígrados (llamados también Celsius) y obtener automáticamente su equivalente en grados Fahrenheit.

```
10  REM PROGRAMA PARA CONVERTIR GRADOS C A F
20  PRINT "INTRODUCIR LA TEMPERATURA MAS BAJA"
30  INPUT L
40  PRINT "INTRODUCIR LA TEMPERATURA MAS ALTA"
50  INPUT H
60  FOR X = L TO H
70  LET F = X * 9/5 + 32
80  PRINT X; "EN CENTIGRADOS ES" F;
    "EN FAHRENHEIT"
90  NEXT X
100 END
```

Introduzca este programa, apriete LIST para verificar que ha sido incorporado correctamente y luego apriete RUN. Primero se le pedirá que introduzca la temperatura más baja. Teclee −5. A continuación deberá incorporar la temperatura más alta. Teclee 10. Y ahora el programa convertirá todas las temperaturas, en intervalos de un grado, desde −5 a 10 en unidades centígradas a sus equivalentes en la escala Fahrenheit. En la pantalla deberá aparecer algo semejante a:

```
-6   EN CENTIGRADOS ES 21,2   EN FAHRENHEIT
-5   EN CENTIGRADOS ES 23   EN FAHRENHEIT
-4   EN CENTIGRADOS ES 24,8   EN FAHRENHEIT
-3   EN CENTIGRADOS ES 26,6   EN FAHRENHEIT
-2   EN CENTIGRADOS ES 28,4   EN FAHRENHEIT
-1   EN CENTIGRADOS ES 30,2   EN FAHRENHEIT
-0   EN CENTIGRADOS ES 32   EN FAHRENHEIT
```

Observe que las columnas no son exactamente iguales debido a la coma de los decimales, pero cada valor en grados centígrados está impreso con su equivalente en grados Fahrenheit en una sola línea. Después de pasar el programa unas cuantas veces, vuelva a escribir la línea 80 tal como está, pero sustituyendo todos los punto y comas por comas. Pase el programa otra vez. Como puede apreciar, ahora aparece todo desordenado.

Para comprobar por qué ha sucedido esto, probemos un programa muy simple, con el fin de comparar el efecto de las comas con el de los punto y comas. Digite NEW <CR>. Introduzca:

```
10 REM COMPARAR; CON,
20 PRINT "ESTA LINEA USA PUNTO Y COMAS"
30 PRINT "H"; "E"; "L"; "P"
40 PRINT "ESTA LINEA USA COMAS"
50 PRINT "H", "E", "L", "P"
60 END
```

Cuando el BASIC imprima la línea 30, aparecerá en la pantalla HELP; en cambio en la línea 50 será H E L P. (Véase el recuadro "Complementos al BASIC" para las variaciones entre ordenadores diferentes.) La coma tiene muchos usos en el BASIC, pero con relación a PRINT tiene el efecto de que cada letra aparezca en la pantalla (o impresa en papel) espaciada, normalmente entre 8 y 16 espacios, según la versión del BASIC. Si PRINT se utiliza sin comas o punto y comas, las letras aparecerán en líneas distintas.

Además de ilustrar sobre el uso del punto y coma en el BASIC, nuestro programa de conversión de temperaturas vuelve a examinar varios puntos de los dos primeros apartados de nuestro curso de programación BASIC. Las líneas 30 y 50 sitúan las variables L y H en los valores mínimo y máximo de las temperaturas que se quieren convertir. La línea 60 es la primera parte de un bucle FOR-NEXT. Parece diferir del bucle FOR-NEXT que hemos encontrado hasta ahora al utilizar letras en vez de números. De hecho, no hay ninguna diferencia.

Las letras que empleamos aquí, L y H, son variables con valores numéricos que corresponden a los valores digitados en el INPUT L y el INPUT H. Si, como se ha sugerido anteriormente, ha introducido −5 y 10, la expresión FOR X = L TO H es equivalente a FOR X = −5 TO 10.

La línea 80 dice: PRINT el valor de X (que empieza a la temperatura más baja y cada vez se incrementa en 1 hasta la temperatura más alta), seguido directamente en la misma línea (ésta es la razón del uso del punto y coma) por las palabras entrecomilladas, seguidas a su vez directamente (otro punto y coma) por el valor de F. Si se fija detenidamente en F, comprobará que es el valor de la temperatura en grados centígrafos, convertidos en Fahrenheit al multiplicar por nueve, dividir por 5 y sumarle 32. La línea NEXT X asegura la continuación del proceso hasta alcanzar el límite superior en el bucle FOR-NEXT.

Antes de continuar con una variación más sofisticada en el apartado PRINT, merece la pena revisar la línea 70 de nuestro programa de conversión de temperaturas:

```
70 LET F = X* 9 / 5 + 32
```

Esta línea asigna un valor a la variable F (abreviatura de Fahrenheit). El programa, primero, toma el valor de X (temperatura en centígrados), lo multiplica por 9, lo divide por 5 y le suma 32. La expresión de esta fórmula en un libro corriente de física sería F = C × 9 : 5 + 32. El lenguaje BASIC utiliza los símbolos * para la multiplicación, / para la división, + para la adición y − para la sustracción.

En la aritmética corriente, y también en BASIC, es importante el orden en que se efectúan las operaciones. La multiplicación tiene siempre la máxima prioridad, seguida de la división, la adición y la sustracción. Si una parte de la expresión matemática está entre paréntesis, debe calcularse en primer lugar. Si se quiere que una adición sea efectuada antes que una multiplicación, debe ponerse entre paréntesis. Por ejemplo, si quiere saber el saldo de su cuenta corriente en dólares más sus ahorros en la misma moneda, podría expresarlo en parte de un programa semejante al siguiente:

```
D = (C + S)* 0,0066
```

Si su cuenta corriente es de 120 000 pesetas (C) y sus ahorros son de 300 000 pesetas (S) y una peseta vale 0,0066 dólares, primero se sumarán las pesetas (C + S) y luego se multiplicará por 0,0066 para convertirlas en dólares. Sin el paréntesis, el valor de sus ahorros sería multiplicado en primer lugar por 0,0066 y luego sería añadido al resultado el valor de la cuenta corriente. ¡Que no era lo que quería calcular! Compruebe siempre que las operaciones aritméticas sean realizadas en el orden correcto.

Print Using

Para comprobar una última característica importante de nuestro programa de conversión de temperaturas, digítelo otra vez y póngalo en funcionamiento. Introduzca −10 como valor de la temperatura más baja y 10 para la más alta. Como ya hemos visto, la impresión en la pantalla es muy desigual. Esto es debido a los punto y comas utilizados en la línea 80 para concatenar (unir) todas las partes que se imprimen, en vez de hacerlo en líneas separadas. Lo cual es positivo, excepto que el espacio ocupado por las cifras —en grados centígrados y Fahrenheit— varía. Esto hace que las columnas queden desalineadas.

Casi todas las versiones de BASIC tienen una característica PRINT llamada PRINT USING, que permite poner en orden la impresión de las palabras o de los números. Si se quiere imprimir el valor de X y se sabe por adelantado que varía desde —99 a 99, las cifras pueden imprimirse correctamente alineadas usando PRINT USING "# # #"; X. Los tres signos permiten imprimir hasta tres dígitos, o dos dígitos precedidos del signo menos. Si se introducen más de tres dígitos, no serán impresos correctamente. Sin embargo, si se introducen sólo uno o dos dígitos, serán colocados correctamente. Si son necesarias comas decimales, pueden incluirse en la posición correcta con los signos. Por ejemplo, la expresión puede tener la forma PRINT USING "# # # · # #"; X. Utilice un signo para cada dígito. Todas las comas decimales se alinearán automáticamente.

Modifique el programa original, cambiando la línea 80 y añadiendo las líneas 82, 84 y 86:

```
80 PRINT USING "# # #"; X;
82 PRINT "EN CENTÍGRADOS ES";
84 PRINT USING "# # # · # #"; F;
86 PRINT "EN FAHRENHEIT"
```

Digite otra vez LIST y RUN. Todas las columnas aparecerán ahora perfectamente alineadas.

En los próximos capítulos del curso, veremos cómo se puede "guardar" un programa, no siendo necesario volverlo a escribir cada vez.

Ejercicios

■ Introducir una "temperatura más baja" que −1 000. ¿Por qué no funciona ahora el programa? ¿Cómo modificaría el PRINT USING en la línea 80 para hacerlo funcionar?

■ Cambiar la línea 84 de forma que sólo se impriman números enteros (sin decimales).

■ Escribir un programa para convertir una serie de cifras de pesetas a dólares, utilizando un valor de cambio de 0,0066 dólares por peseta.

Complementos al BASIC

PRINT USING

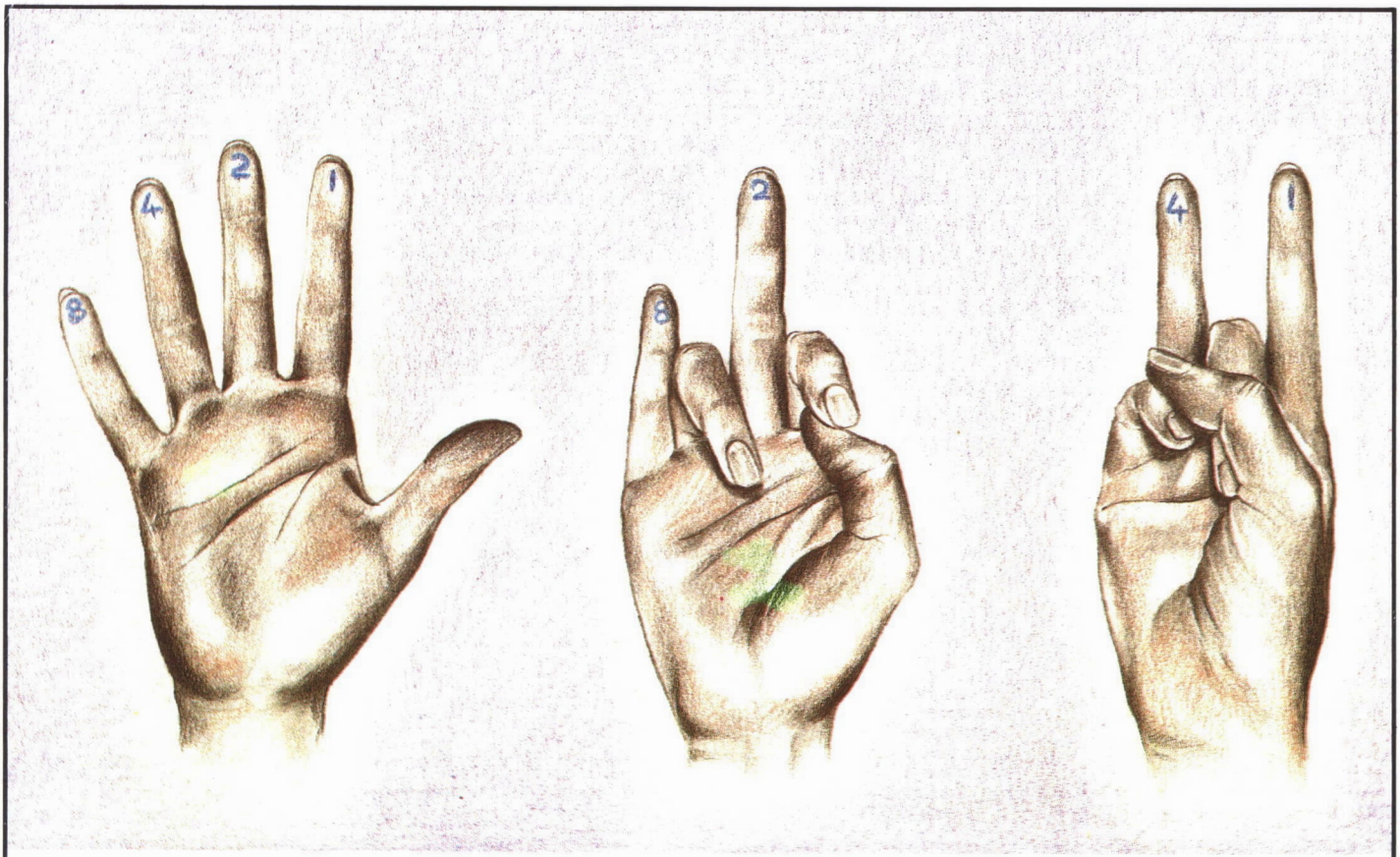
Este mando no existe en el Commodore 64, Oric, Spectrum, ZX81 o BBC Micro. Sin embargo, el BBC puede limitar el número de decimales a imprimir, lo cual se consigue utilizando la siguiente instrucción:
@ % = 131594

COMA

El uso de la coma entre campos impresos separará los temas a imprimir insertando un número de espacios, que varían según el ordenador que se emplee. En el BBC y Commodore 64, son 10 espacios; en el Dragon, Spectrum y ZX81 son 16. El Oric utiliza sólo 4 espacios

Cuando 1 y 1 son 10

Los ordenadores realizan sus prodigiosos cálculos únicamente con dos dígitos: 0 y 1



La forma más fácil de convertir números binarios pequeños a sus equivalentes decimales consiste en imaginar que se escribe el "valor de posición" de cada columna binaria en los dedos de la mano derecha. Siempre que el número binario no exceda de cuatro dígitos, todo lo que hay que hacer es extender el dedo adecuado si el dígito binario correspondiente es un 1, o doblarlo si es 0.
Cuando se extienden los dedos adecuados para el número 1010, se obtiene 8 y 2, que sumados dan el número 10 en el sistema métrico decimal. La tercera ilustración muestra cómo decodificar 0101: da un 4 y un 1, que se convierten en 5 en la forma decimal. Mediante este método, intente hallar los equivalentes decimales de los números 1110 y 0110.
El método puede ampliarse utilizando las dos manos para expresar números binarios mayores. Al objeto de realizar esta operación, los dedos de la mano izquierda (con la palma frente a usted) deben corresponderse con los valores 16, 32, 64 y 128, con el valor 16 para el dedo meñique y el 128 para el índice

Elaine Keenan

Mucha gente está tan acostumbrada a nuestro sistema numérico que no se le ocurre pensar que puedan existir otros.

Los romanos idearon un sistema para la representación numérica utilizando las letras del alfabeto. Así, la X significaba 10, la L 50, la C 100, la D 500, etc. El sistema romano resultaba idóneo como forma de registrar números simples; sin embargo, no sería apto para el cálculo con ordenador. En números romanos, incluso las sumas son difíciles, por una razón: no existe el concepto de "valor de posición"; la posición de un dígito no aporta ningún dato sobre cuál es su "valor".

Fíjese en los números 506 y 56. La única diferencia aparente es el cero que aparece en medio del primer número. Su función consiste en informarnos que en el número 506 no hay "decenas", sólo cinco "centenas" y seis "unidades".

Cada "columna" o posición en un número convencional tiene un "valor" asociado con él. La columna de la derecha del número es la de las "unidades", la siguiente (en dirección a la izquierda) es la columna de las "decenas", la próxima la de las centenas y así sucesivamente. El dígito utilizado en cada "columna" significa qué valor de ella está representado.

Usted puede preguntarse qué relación tiene lo expuesto anteriormente con los ordenadores y el sistema binario. Los ordenadores son máquinas electrónicas, que pueden operar fácilmente con números mediante la utilización de niveles de voltaje. Cinco voltios representan un uno, y cero voltios un cero. Como aprendimos en "Bits y bytes", los unos y ceros resultan perfectamente adecuados para representar cualquier número, por grande que sea.

Usando el conocido sistema decimal de base 10 (denominado también *sistema denario*), el número 506 es una manera concisa de representar el equivalente a

quinientos seis nudos en una cuerda o quinientas seis muescas en un bastón. En aritmética binaria, este número se representa por un incómodo 111111010.

Ya que el sistema empleado aquí es el "binario", el valor de situación de cada dígito y de cada columna es distinto. En vez de incrementar el valor en potencias de 10, las columnas aumentan en potencias de 2.

La columna de la derecha sigue siendo la de las "unidades", pero como sólo hay dos símbolos (0 y 1), nos quedamos sin dígitos en cuanto empleamos el 1. En el sistema decimal, esta circunstancia sólo se presenta cuando llegamos al 9; la próxima columna usa un dígito que dice: nos hemos quedado sin símbolos —no tenemos nada para representar números mayores que 9—, por esta razón utilizaremos la columna de las "decenas" y con un 1 indicaremos que ahora tenemos un "conjunto" de diez.

El sistema binario funciona de la misma forma. En vez de formar grupos de diez y escribir 10, el sistema binario agrupa conjuntos de 2, y por tanto los dígitos binarios 10 representan el número decimal 2.

Si escribimos el número quinientos seis en los sistemas decimal y binario, advertiremos claramente la semejanza esencial existente:

Centenas		Decenas		Unidades		
5		0		6		
$= 5 \times 100 +$		$0 \times 10 +$		6×1		$(= 506)$

256	128	64	32	16	8	4	2	1
1	1	1	1	1	1	0	1	0

$= 1\times256 + 1\times128 + 1\times64 + 1\times32 + 1\times16 +$
$+ 1\times8 + 0\times4 + 1\times2 + 0\times1 \qquad (= 506)$

Las reglas aritméticas en el sistema binario son exactamente las mismas que las del conocido sistema decimal: la única diferencia consiste en que después del 1 ya no hay otros símbolos numéricos, en lugar de suceder esto con el 9. Realicemos unas sumas para probarlo. Los equivalentes decimales están representados entre paréntesis.

(3)	11
+(5)	+101
(8)	1000

(1 + 1 = 0 se acumula 1)
(1 (acumulado) + 1 = 0 se acumula 1)
(1 (acumulado) + 1 = 0 se acumula 1)
(1 (acumulado) + 0 = 1)

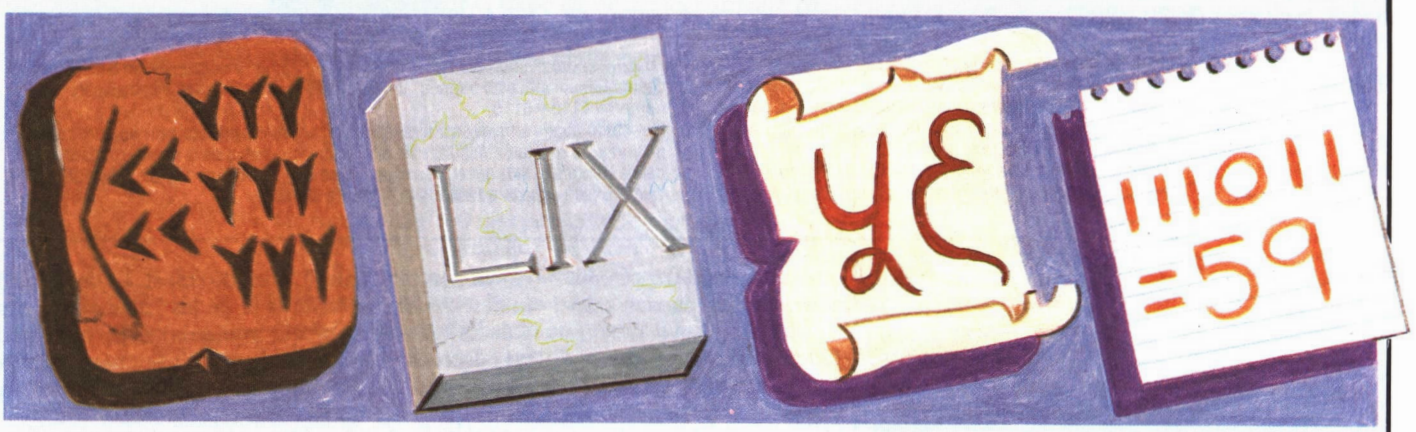
En el sistema binario, como se observa, sumar 1 y 1 significa que se agotan los símbolos, puesto que sólo se pueden emplear unos y ceros. Por tanto se dice: "Uno y uno igual a cero y se lleva uno" (al igual que en el sistema decimal, en el que si sumamos 1 y 9 agotamos los símbolos —no existe ningún símbolo mayor que 9—, y por ello decimos de nuevo: "Nueve y uno igual a cero y se lleva uno"). A continuación se representa una suma ya resuelta y otras dos que usted mismo debe solucionar.

(4)	100
+(6)	+110
(10)	1010

(7)	111	(3)	11
+(2)	+ 10	+(12)	+1100
(?)	?	(?)	?

Después de lo expuesto, habrá observado que los números binarios son mucho más largos que sus equivalentes en el sistema decimal. Intente sumar 11010110 a 1101101 (recuerde que debe mantener las columnas alineadas del mismo modo que si realizara una suma en el sistema decimal de un número con otro más pequeño).

Historia de los números



Babilónico **Romano** **Hindú** **Binario**

Los antiguos babilonios tenían un sistema numérico avanzado, basado en el número 60 en vez del 10. En la ilustración se representa el número 59 en escritura cuneiforme babilónica. El empleo del 60 como número base tenía muchas ventajas e incluso en la actualidad quedan algunas reminiscencias de este sistema. Un minuto tiene 60 segundos, una hora 60 minutos y seis veces 60 grados en un círculo: todo ello constituye un vestigio de aquel sistema matemático perfeccionado hace 4 000 años.

El sistema romano representó un atraso considerable. Los números se representaban con las letras del alfabeto, pero la posición de cada cifra no indicaba su valor, por lo que resultaba prácticamente imposible realizar la operación aritmética más simple.

Los hindúes utilizaban nueve signos para designar los números del 1 al 9 y más tarde añadieron otro signo que representaba el cero. Pero su contribución vital fue la introducción del "valor de posición": la idea de que la posición de un dígito en un número determina su "valor". Así el "valor" de 3 en 30 es de tres decenas. Los árabes adoptaron el sistema hindú, que gradualmente se extendió por Europa. Uno de los matemáticos árabes más importantes se llamaba Al-Jwarizmi. La pronunciación latinizada de su nombre determina el término matemático de algoritmo, y su libro *Al-yabr wa'l Mugabalah* nos trae a la memoria la palabra álgebra.

Los ordenadores utilizan el sistema binario porque los números, con independencia de su magnitud, pueden representarse usando sólo unos y ceros

Andy Leslie

El código de barras

Esas misteriosas barras sobre las tapas de los libros, en las cubiertas de revistas o fascículos y en los productos de los supermercados transmiten su mensaje a un ordenador y ayudan a llevar el control de almacenaje y distribución más eficazmente

Decodificación de barras

COMIENZO FIN

La ilustración superior muestra un código de barras, que representa el número 72. A simple vista parece sólo una serie de líneas negras de distinto grosor. En este caso cada serie incluye cinco barras, dos de las cuales son anchas. La posición de las dos líneas anchas en cada serie indica el número. Unas barras adicionales señalan el principio y el final de cada unidad de información individual. En esta ilustración, el número 72 está codificado en dos unidades, el dígito 7 y el dígito 2.

Existen muchas formas diferentes de codificar información en un código de barras. Dado que el grosor de la barra es variable, ésta puede representarse numéricamente como un 1 o un 0. Esto nos conduce directamente a la matemática binaria de los ordenadores. El *Universal Product Code* es un código de barras ligeramente diferente que se utiliza en el comercio. Según este código las barras pueden ser de diversos espesores, se requieren menos barras y la información se lee a partir de la anchura de la línea.

La razón de que los códigos de barras se haya extendido tanto en las librerías y en los supermercados se debe a que pueden ser leídos por una máquina. El librero se vale de un "lápiz" sensible a la luz, que ilumina el código de barras y registra la cantidad de luz reflejada. En contraste con el fondo blanco, las barras negras casi no reflejan luz. La luz reflejada se convierte en una señal eléctrica que es amplificada. Cuando se registra luz se traduce en un 1 binario, y si no se registra luz y por lo tanto no se produce señal, en un 0 binario.

Las barras proporcionan secuencias de ceros, y cuanto más anchas son, más ceros contienen. Del mismo modo, el fondo blanco proporciona secuencias de unos. De esta manera, la "varita mágica" alimenta el ordenador con los patrones de dígitos binarios, a partir de los cuales puede determinar la composición de un código de barras

¿Se ha preguntado en alguna ocasión qué significan esas barras impresas en los productos empaquetados del supermercado, en las tapas de un libro o en las cubiertas de algunas revistas o fascículos? Pues bien, se trata del *código de barras*, un ingenioso recurso que puede ser leído en una fracción de segundo por una "varita mágica" sensible a la luz y que introduce en un ordenador la información relativa a esos productos. Mediante este sistema se puede saber, en cualquier momento, cuál es el stock, la fecha de fabricación o impresión, la fecha de caducidad, etc., de un determinado producto.

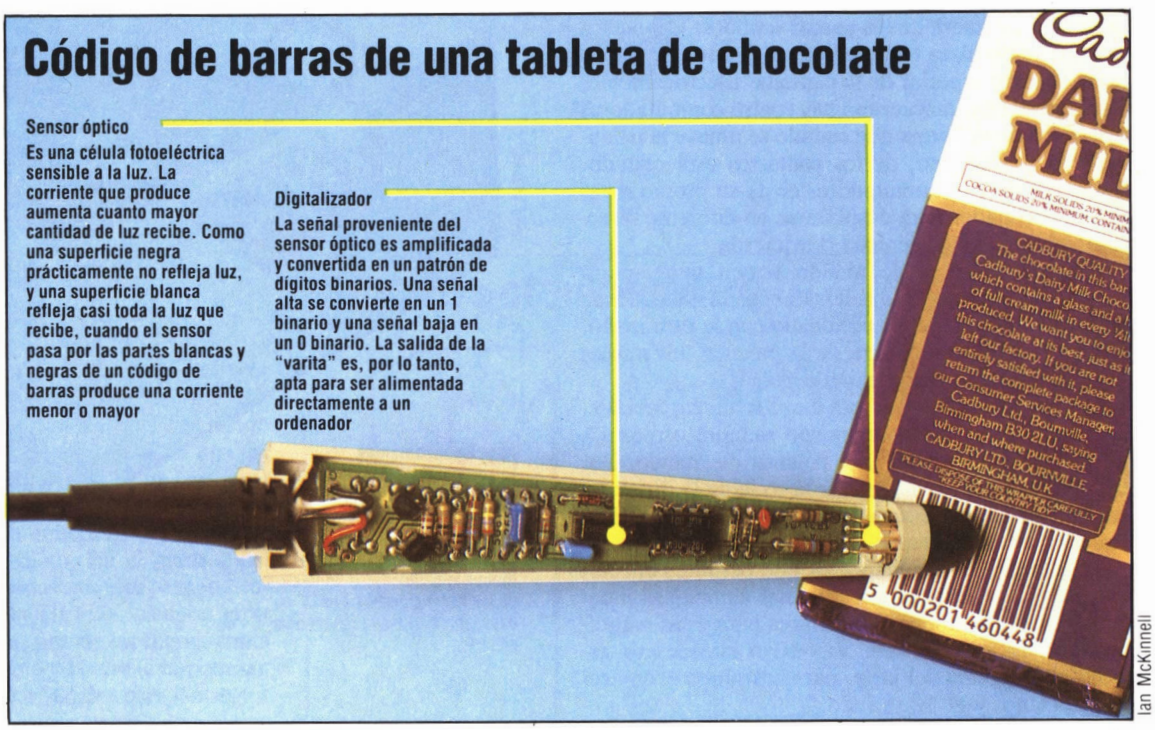
Veamos cómo funciona este control en el caso de los libros de bolsillo. Cada libro que se publica en los principales países del mundo posee un International Standard Book Number (ISBN), que consiste en uno o más dígitos en los cuales constan el idioma o la zona geográfica en los que se publica el libro (84 para España y todos los países de lengua castellana; el 0 y el 1 para Gran Bretaña, Estados Unidos y otros países de lengua inglesa; el 2 para Francia; el 3 para Alemania, Austria y la Suiza de lengua alemana, etc.), entre dos y siete dígitos para identificar al editor, y entre uno y seis dígitos para identificar el título del libro y su edición. Esto arroja un total de nueve dígitos, y luego existe un dígito de control (que el ordenador utiliza para verificar si todos estos dígitos le han sido proporcionados en el orden correcto).

Para la codificación en barras, los libros se numeran de acuerdo al sistema European Article Numbering (EAN; numeración europea de artículos), que utiliza un total de 13 dígitos (la mayoría de los productos de las tiendas de comestibles suelen utilizar un número corto de ocho dígitos). Los tres primeros dígitos corresponden a la "bandera" de la EAN (para los libros es el 978). Luego viene el ISBN y, por último, un dígito de control EAN alternativo.

En España, el ISBN (junto con su propio dígito de control) también se imprime en números en el código de barras; éstos pueden ser leídos tanto por el ojo humano como por un lector de caracteres ópticos (Optical Character Reader).

Los lectores de caracteres ópticos constituyen otro desarrollo interesante de consecuencias mucho más amplias. En la actualidad, existen máquinas que son capaces de leer literalmente la palabra impresa mediante la exploración óptica de la línea. La señal de salida del lector se acopla al ordenador, el cual puede entonces procesar la información de diversas maneras. Las palabras leídas por el explorador pueden, por ejemplo, visualizarse en la pantalla del ordenador, eliminando la agotadora tarea de mecanografiarlas. Y todo ello gracias a este ingenio que está revolucionando el mundo: el ordenador.

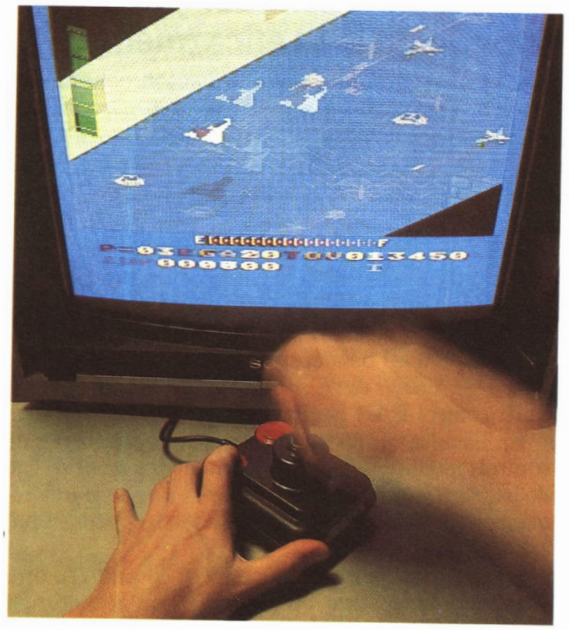
Código de barras de una tableta de chocolate



Ian McKinnell

Mandos veloces

Algunos dispositivos que ayudarán a los entusiastas de los juegos por ordenador a acelerar la acción



En un juego por ordenador, quizá tenga que pilotar una nave espacial a través de las líneas enemigas y disparar sus misiles para destruir un blanco. La palanca de mando transfiere el control de la nave espacial desde un teclado sofisticado hasta sus propias manos. Es una réplica fiel de las que llevan los aeroplanos.

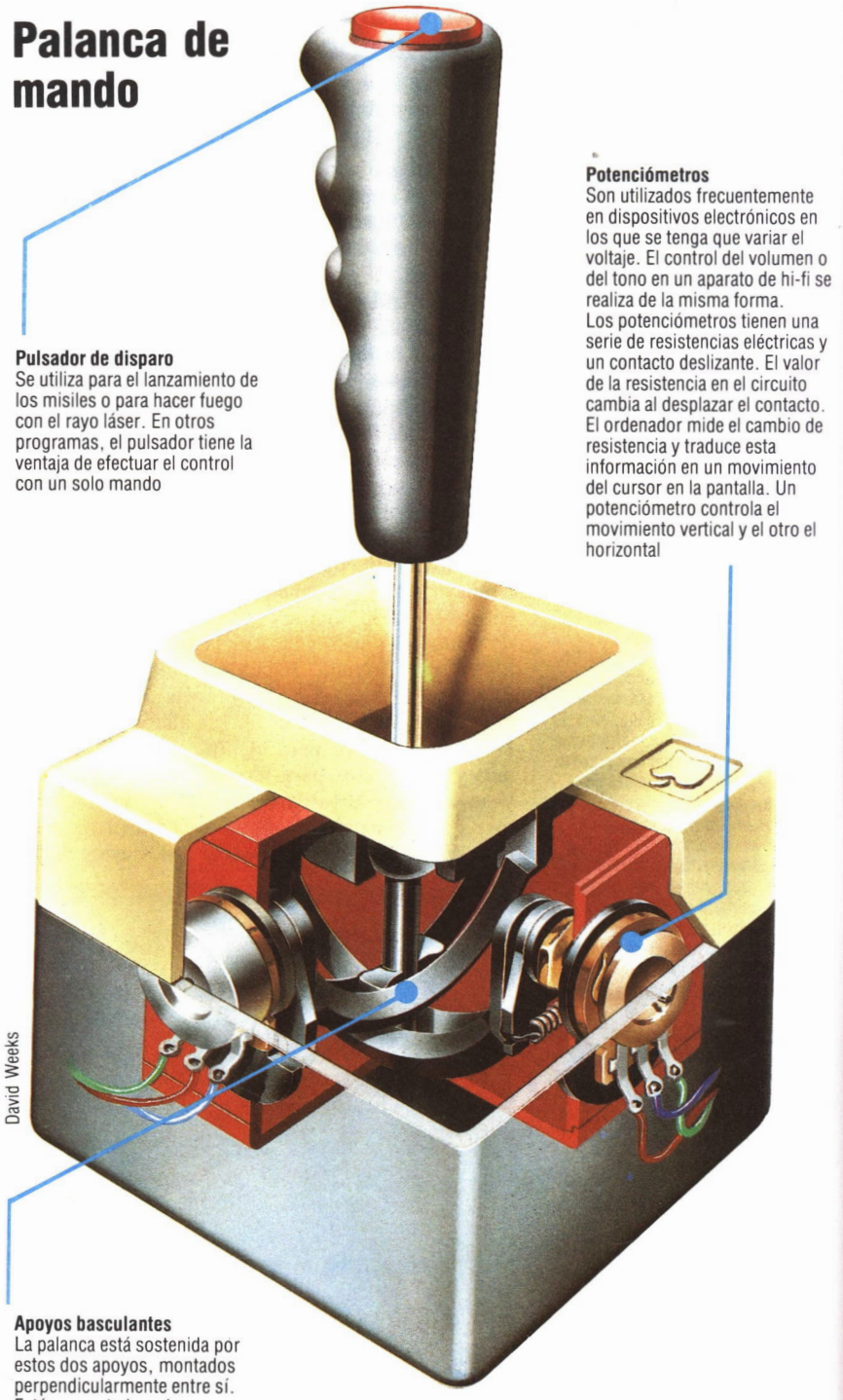
La palanca de mando se conecta a la parte trasera del microordenador. La nave espacial, o cualquier otro objeto que se controle, se mueve en la misma dirección que la palanca. Normalmente puede desplazarse en cualquiera de los cuatro sentidos. Cuando la palanca se desplaza hacia adelante, la nave se mueve hacia la parte superior de la pantalla. Eléctricamente, en el interior del mecanismo hay cuatro conmutadores colocados de tal forma que cuando se mueve la palanca, uno, y sólo uno, de los contactos está cerrado. Cada uno de los conmutadores envía su propio mensaje al ordenador para desplazarse en diferente dirección: arriba, abajo, derecha o izquierda.

Algunas palancas de mando tienen también un botón para disparar los misiles. El botón está a un lado de la palanca y debe ser accionado con la otra mano. O bien, en la empuñadura de la palanca, los misiles son lanzados apretando el dedo pulgar.

Los microordenadores más baratos, en particular el Sinclair ZX81 y el Spectrum, no siempre ofrecen la posibilidad de conectar una palanca de mando. Por tanto se deberá o bien apretar la tecla de la dirección en que se desea que se desplace, o bien comprar una interface para la palanca.

La interface es un adaptador que permite conectar una palanca al ordenador. Algunas compañías independientes han fabricado interfaces para estas máquinas, pero incluso con este dispositivo es necesario escribir el programa del juego para introducir el control de la palanca y el del teclado.

Palanca de mando

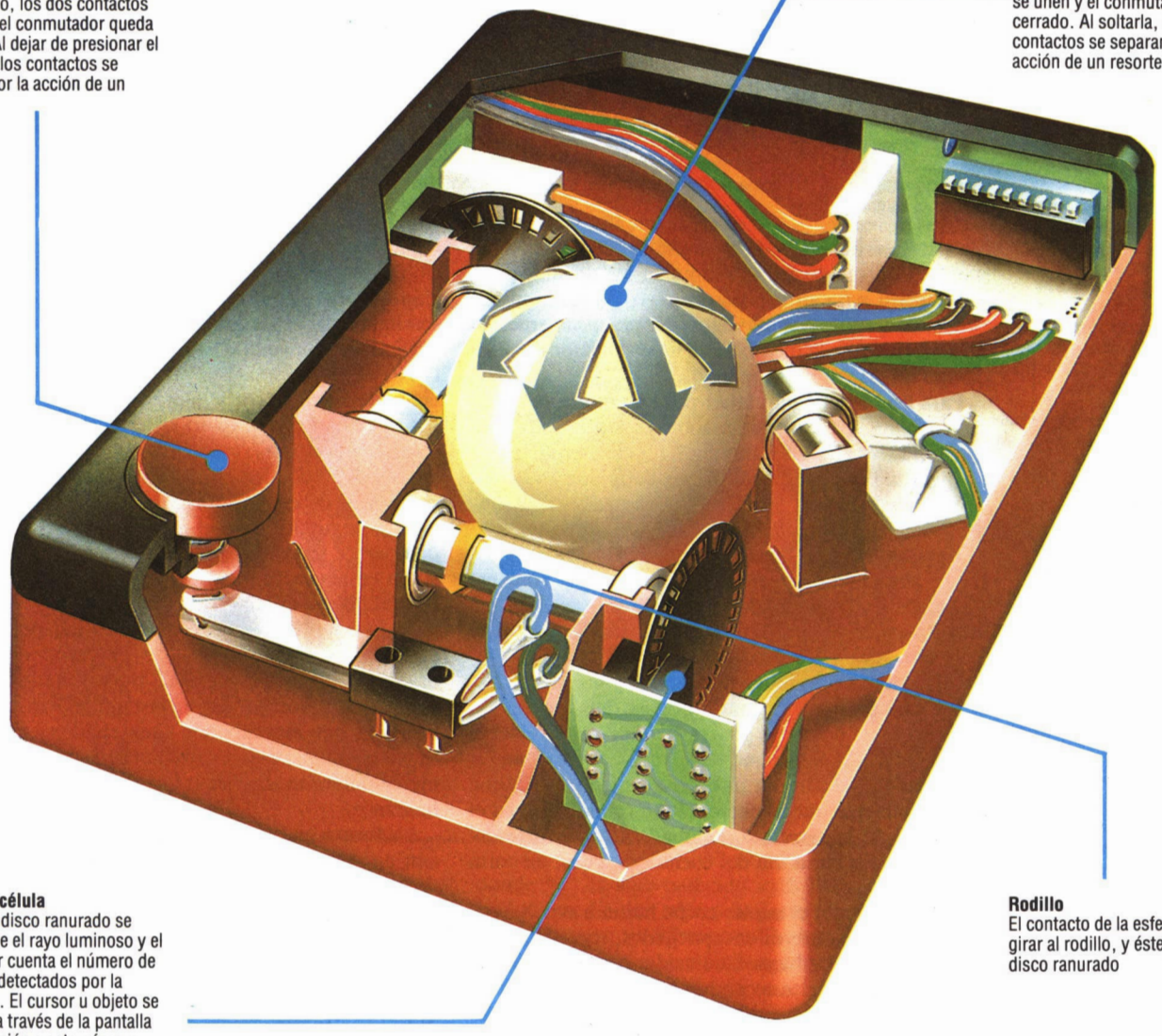


Pulsador de disparo
Se utiliza para el lanzamiento de los misiles o para hacer fuego con el rayo láser. En otros programas, el pulsador tiene la ventaja de efectuar el control con un solo mando

Potenciómetros
Son utilizados frecuentemente en dispositivos electrónicos en los que se tenga que variar el voltaje. El control del volumen o del tono en un aparato de hi-fi se realiza de la misma forma. Los potenciómetros tienen una serie de resistencias eléctricas y un contacto deslizante. El valor de la resistencia en el circuito cambia al desplazar el contacto. El ordenador mide el cambio de resistencia y traduce esta información en un movimiento del cursor en la pantalla. Un potenciómetro controla el movimiento vertical y el otro el horizontal

David Weeks

Apoyos basculantes
La palanca está sostenida por estos dos apoyos, montados perpendicularmente entre sí. Están conectados a los potenciómetros. Cuando se mueve la palanca, los contactos de los potenciómetros se deslizan y cambia la resistencia eléctrica



Pulsador de disparo
Al apretarlo, los dos contactos se unen y el conmutador queda cerrado. Al dejar de presionar el pulsador, los contactos se separan por la acción de un resorte

Bola
Al presionarla, los dos contactos se unen y el conmutador queda cerrado. Al soltarla, los contactos se separan por la acción de un resorte

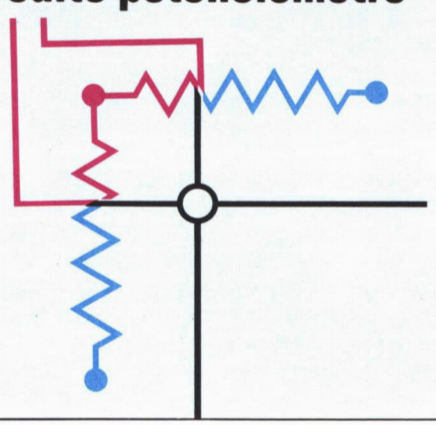
Luz y fotocélula
Al girar el disco ranurado se interrumpe el rayo luminoso y el ordenador cuenta el número de destellos detectados por la fotocélula. El cursor u objeto se desplaza a través de la pantalla en proporción a este número. Existen dos juegos de disco y fotocélula para controlar tanto el movimiento horizontal como el vertical en la pantalla

Rodillo
El contacto de la esfera hace girar al rodillo, y éste guía el disco ranurado

Bola de mando

Imagínese que está guiando el cursor de la pantalla a través de un laberinto. Tiene que ser capaz de hacer avanzar el cursor y de guiarlo a través de los pasajes a medida que se entrelazan y giran. La bola de mando está diseñada para este tipo de problemas. Es una esfera del tamaño de una bola de billar, que se hace girar con la palma de la mano. Al girar la bola, el objeto se mueve en la misma dirección, obteniéndose un control completo e inmediato. En el interior del aparato hay dos juegos de ruedas situadas perpendicularmente entre sí, que rozan con la bola. Al hacer girar ésta con la palma de la mano, una de las ruedas capta la parte vertical del movimiento y la otra la horizontal. El ordenador debe unir ambas señales para conseguir la trayectoria.

Circuito potenciómetro



La palanca de mando está conectada a dos resistencias variables (llamadas potenciómetros). La conexión mecánica mueve el cursor a lo largo de una u otra resistencia (representadas por las líneas en zigzag). Por tanto, la posición de la palanca determina la resistencia eléctrica de los dos potenciómetros. El ordenador verifica los voltajes y calcula la posición de la palanca. Luego, el ordenador convierte esta información en cambios de posición en la pantalla

Términos clave

Reenvío total

Cómo "vigila" el ordenador toda la información almacenada en su memoria, asegurándose de que nunca se olvide y tenga un acceso inmediato

En términos humanos, la memoria es el almacén de la mente, el lugar donde se acumulan los detalles de la experiencia para su uso posterior. Y en términos de ordenador, "memoria" significa prácticamente lo mismo, sólo que la memoria de un ordenador tiene un campo de acción más limitado.

Para un ser humano, tener poca memoria es un inconveniente. Para un ordenador, es un desastre. Sin memoria, el ordenador no tendría nada en qué basarse ni nada que le indicara lo que tiene que hacer, ya que también utiliza su memoria para almacenar los programas que lo guían.

En ambos casos, la palabra "memoria" implica dos cosas: almacenaje y reenvío de información. Almacenar datos sin la posibilidad de volverlos a extraer, no es muy útil, e intentar obtener información que no ha sido almacenada es obviamente inútil.

Los dos tipos de memoria son también similares desde otro punto de vista. Parece ser que la memoria humana es de dos tipos generales: de corta y de larga duración. Un hombre que debe cruzar una carretera, por ejemplo, recordará que debe esperar hasta que pase el vehículo que se acerca. Pero cuando esté en el otro lado de la carretera, se olvidará totalmente del automóvil. Su memoria del coche era, pues, de corta duración.

Sin embargo, si el mismo coche hubiera estado ocupado por dos hombres encapuchados, sentados en el asiento trasero, y fuera conducido por la mujer del hombre que cruza, éste podría recordar bien todo el incidente, ¡incluso el modelo y color del coche, y posiblemente el número de matrícula! Esto es memoria de larga duración.

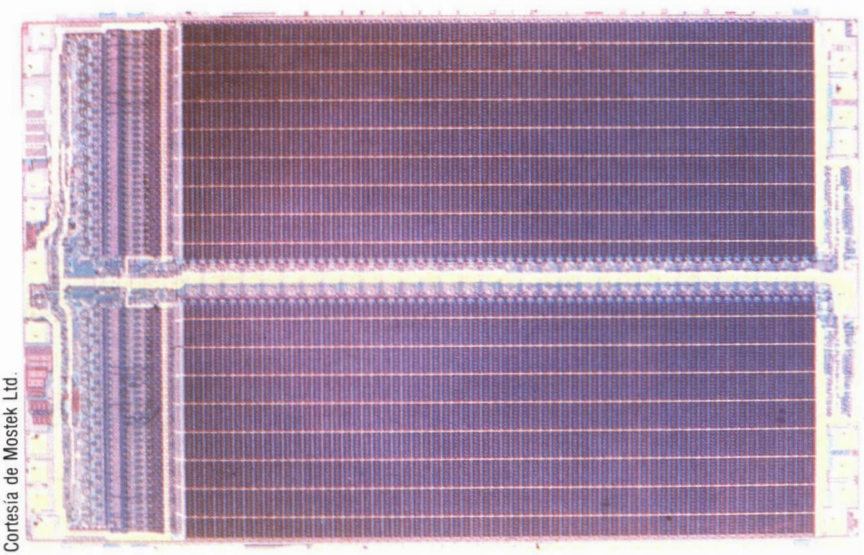
Se puede decir que también los ordenadores tienen memoria de corta y larga duración. La de larga duración o "no volátil" contiene los programas e información que el usuario quiere guardar. Estos programas se almacenan como grabaciones magnéticas en la superficie de una cinta de cassette, discos flexibles o en la ROM.

La memoria de corta duración o "volátil" es el chip RAM, situado en el interior del mismo ordenador, y sólo se utiliza temporalmente mientras el ordenador está trabajando. En el momento en que se corta el suministro de energía, aunque sea durante una fracción de segundo, todo el contenido de esta memoria desaparece instantáneamente.

Sin embargo, la mentada analogía con la memoria humana no es totalmente exacta. Para que el ordenador funcione, es necesario traspasar los programas y datos adecuados desde el almacenamiento de larga duración al de corta duración, para que así el ordenador pueda tener un acceso instantáneo a ellos. Y la *forma* en que los datos son almacenados y reenviados desde o hacia la memoria del ordenador es completamente diferente al proceso que se produce en los seres humanos.

La forma en que funciona la memoria humana es aún un misterio, puesto que los recuerdos de un incidente determinado no parece ser que se almacenen en un pequeño segmento identificable del cerebro. No tenemos que descifrar dónde está situado un tema determinado y reenviarlo al primer plano de la mente. Y cuando hemos terminado de recordar, no tenemos que preocuparnos de volverlo a situar en un punto determinado del cerebro.

Memoria RAM
El chip RAM (abajo) es uno de los adelantos recientes más importantes en la tecnología del ordenador. La RAM (Random Access Memory: memoria de acceso directo) es una de las variedades de la memoria totalmente electrónica, una categoría que incluye también la ROM (Read Only Memory: memoria de lectura solamente). Las cintas de cassette y los discos flexibles magnéticos son ejemplos de otro tipo de memoria: la electromagnética. La memoria RAM se fabrica con silicio, usando un proceso fotográfico y reacciones químicas, para crear miles de diminutos transistores. Cada "bit" de memoria necesita como mínimo un transistor en el circuito.
El tiempo requerido para "escribir" un simple bit en una de las 16 384 celdas de almacenamiento es de unos 200 nanosegundos (una quinta millonésima parte de segundo)



Cortesía de Mostek Ltd.

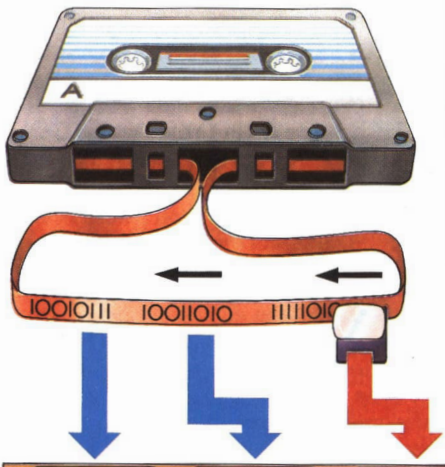
La memoria del ordenador: un caos organizado

En la memoria de un ordenador, lo que es vital es la *situación* de cada tema. El ordenador debe ser capaz de encontrar un byte de información determinado, ya sea parte de un programa o parte de los datos del mismo. El ordenador necesita también "tomar nota" de dónde pone la información.

La memoria humana se parece más a un buzón completamente lleno de información, pero desordenado. Los componentes de una información son introducidos sin orden, aparentemente al azar, mezclándose entre sí y empujados al interior del cerebro al reunir más y más imágenes y experiencia. De algún modo, el cerebro le da sentido y extrae lo que necesita, cuando lo necesita.

La memoria de un ordenador se parece más a un gigantesco casillero, estando cada casilla completamente separada de las otras. Todo está muy ordena-

Memoria

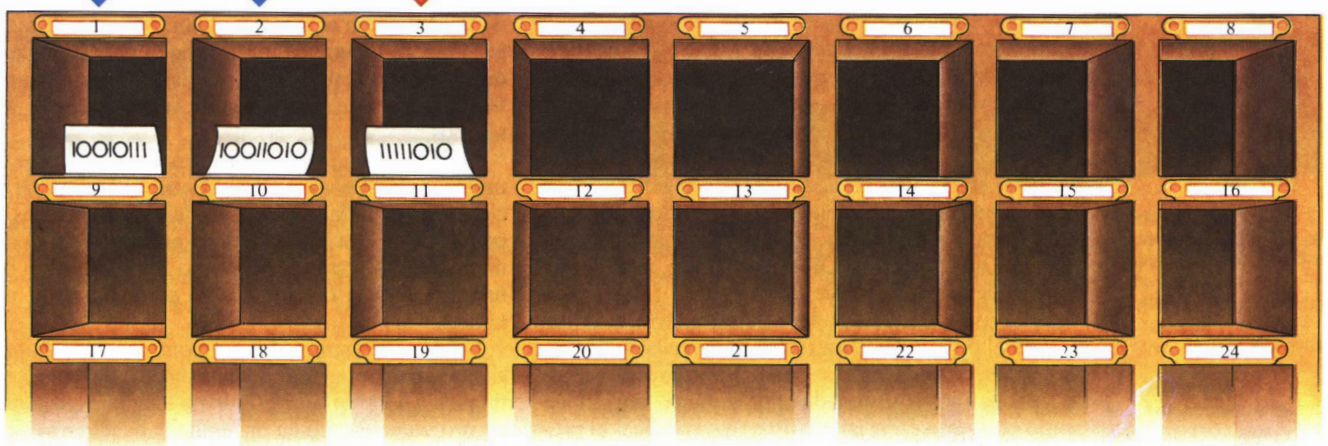


Los programas grabados en una cinta de cassette se almacenan "secuencialmente", con cada bit que compone un byte grabado uno tras otro. Al reproducir la cinta, el ordenador "lee" cada bit, pero los almacena en grupos de ocho (bytes) en cada celda de la memoria. El primer byte de la cinta es colocado en la primera celda disponible, el segundo byte en la próxima, y así sucesivamente. Cuando el ordenador necesita poner en marcha el programa, lo único que debe saber es la "dirección de comienzo".

El ordenador transfiere de forma secuencial el contenido de cada celda al interior de la unidad central de procesamiento, y los bytes hacen que ésta "ejecute" o realice las acciones pedidas por el programa.

Una parte de la memoria está ocupada por los programas del sistema responsables de aspectos fundamentales de su funcionamiento (verificación de qué teclas han sido pulsadas, colocación de los caracteres en la pantalla, etc.). Este software incorporado puede incluir también el lenguaje de programación BASIC. Estos programas internos ocupan espacio en la memoria y, por tanto, dejan menos sitio para el almacenamiento del software comercial o del usuario. Algunas versiones de BASIC, por ejemplo, son almacenadas en 16 Kbytes de memoria. Si el ordenador está provisto de 64 Kbytes de memoria, sólo quedan disponibles 48 Kbytes para otros programas

Cuando se introduce un programa a partir de una cassette, la primera posición de la memoria disponible (vacía) no será, por consiguiente, la primera posición en la RAM. Una de las primeras obligaciones del software base (sistema operativo) es saber y recordar dónde está la primera posición de la memoria disponible para el usuario. Una vez que el programa ha sido introducido en la RAM, el software base dice: "empezar por buscar en la memoria la posición x y continuar examinando cada posición sucesiva de la memoria, introducir el contenido de cada posición en la CPU y realizar lo que ésta diga". El orden original en el que ha sido introducido el programa por el programador es el mismo en el que se ha grabado la cinta. Cuando el programa es transferido desde la cinta hasta la memoria, es colocado en el mismo orden en las celdas. Para el ordenador, el efecto es el mismo que si hubiera sido introducido a través del teclado



Tony Lodge

do; cada casilla tiene un número (llamado su "dirección") y contiene sólo un byte, ni más ni menos. Así pues, el ordenador encuentra la información por el número de la casilla, no por lo que se halla almacenado en esa casilla.

Los ordenadores no tienen inteligencia, y no pueden organizar sus memorias por sí mismos. La única razón por la que un ordenador puede almacenar algo es porque alguien ha colocado la información en la casilla correcta y en el orden adecuado, y a su debido tiempo. ¿Cómo sucede esto en un ordenador doméstico normal?

Al conectar el ordenador, generalmente aparece en la pantalla un mensaje para indicarle que está en funcionamiento. En la mayoría de los casos, también le informa que puede empezar a escribir un programa. Este mensaje, y las ayudas que le permiten empezar a programar, están almacenados en una parte de la memoria interna del ordenador; es necesario almacenarlos en la memoria de larga duración (normalmente, en un chip ROM).

Esta porción de la memoria contiene los programas que verifican si se han pulsado las teclas, que "imprimen" las letras en la pantalla y realizan otros trabajos "básicos" esenciales. También contiene un programa especial que traduce las órdenes, por lo general escritas en BASIC, al lenguaje binario de unos y ceros, comprensible para el ordenador.

Cuando se conecta el ordenador, el mensaje que aparece en la pantalla suele decir: "x bytes libres", donde "x" es algo parecido a 15 797 u otro número cualquiera. Lo que representa este número es las casillas de la memoria que están libres para utilizar. Al ir pulsando las teclas, se van llenando las casillas, y aquí aparece otro punto importante relativo a la memoria del ordenador: el *orden* en que se almacena la información.

Al apretar una tecla, se envía un byte (que representa la letra digitada) a la memoria para su almacenamiento. Al teclear la letra "r", por ejemplo, se coloca esta letra en una casilla de la memoria en forma binaria.

Pero ¿a qué casilla va a parar esta "r"? Pues a la primera ranura de la memoria de corta duración que esté libre. Si se piensa en un bloque de casillas vacías colgado de una pared, la "r" iría a la casilla superior de la esquina izquierda.

Si ahora se aprieta la tecla "e", la serie adecuada de bits irá a la segunda casilla vacía, a la derecha de la "r". Al teclear en tercer lugar una "d", ésta se situará en la tercera casilla junto a la "e". Al mirar el casillero, en la fila superior aparecerán los códigos de la palabra "red".

El ordenador tiene un "contador" interno para calcular qué casilla se ha ocupado. Éste sabe dónde empezar, porque el programa del sistema le dice en qué punto comienza el área libre de la memoria. Cuando se almacena una letra, la cifra del contador se incrementa en una unidad para designar la casilla siguiente.

Mensaje comprendido

Si se pulsa una tecla, de inmediato entran en acción estratos ocultos de software, que decodifican las instrucciones, buscan en la memoria y exploran el teclado a la espera de la próxima orden

Un ordenador es un ensamblaje de metal, plástico y silicio que, si carece de algún programa en su memoria, es incapaz de realizar ninguna clase de tarea útil; algo así como un tocadiscos sin ningún disco en el plato. El proceso para lograr que el ordenador realice la tarea específica que usted necesita se conoce como "programación". Incluso un recién iniciado en programación podrá identificar dos fases diferentes para resolver un problema. En primer lugar, éste debe traducirse y escribirse de forma tal que el ordenador pueda comprenderlo. En segundo lugar, este programa debe alimentarse al ordenador y "ejecutarse". En un segundo momento, estas dos fases se subdividen en dos etapas: la primera le corresponde al propio programador, mientras que en la segunda etapa es el ordenador el que debe emprender las acciones (por lo general sin el conocimiento o la intervención del usuario).

Supongamos que desea escribir un programa para preparar una nómina. Lo primero que necesita es comprender perfectamente el problema. ¿Qué salida requiere usted del ordenador? ¿Qué información necesitará el ordenador para realizar los recibos de pago semanales? Esto puede suponer información acerca de salarios, horas trabajadas por semana, etc. El segundo elemento esencial consiste en especificar el proceso por el cual se ha de producir esta salida, por ejemplo: "¿Cómo se calculan las retenciones para impuestos y los descuentos para la Seguridad Social?"

Si se trata de una aplicación para una gran empresa, esta tarea la puede efectuar un experimentado "analista de sistemas", cuya especialidad es la de analizar la forma en que funciona una empresa y en escribirla de manera que pueda traducirse fácilmente en un programa. En el caso de programas domésticos o educativos, esta labor podrá realizarla el propio programador.

Si los ordenadores pudieran comprender el lenguaje corriente, todas estas "especificaciones de programa" podrían ejecutarse directamente; pero, por desgracia, los ordenadores todavía no lo comprenden. Muchos principiantes encuentran dificultades porque tratan de escribir el programa desde el comienzo hasta el final, del mismo modo en que se traduciría al francés un ensayo escrito en castellano. No obstante, algunos programadores muy experimentados dividen aún más esta etapa. Éstos podrían fraccionar la especificación de la nómina en cuatro "módulos": para la entrada de los datos de la semana, para cálculo, para el almacenamiento de los resultados acumulables (como "impuestos pagados este año") y para imprimir las hojas de salarios.

Cada módulo puede, entonces, dividirse en estructuras más pequeñas. Esto se conoce como "programación estructurada"; cada una de estas secciones más pequeñas es sencilla y puede expresarse en una o dos líneas del programa. Por último, todo el grupo de líneas (el listado del programa) se digita en el ordenador.

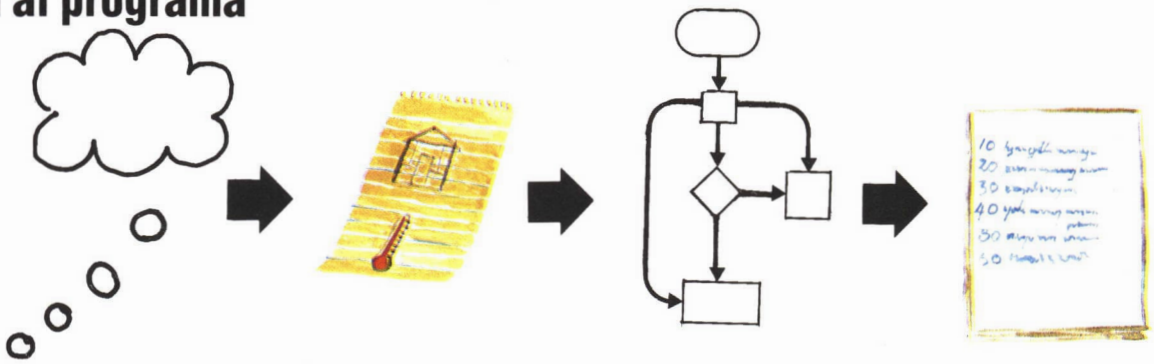
Un buen programador siempre conserva notas de cada etapa y éstas reflejan los muchos niveles que distinguen a un programa escrito en lenguaje corriente de un programa escrito en un lenguaje de alto nivel como el BASIC.

Lo que sucede a partir del momento en que se pulsa RUN queda completamente bajo el control del ordenador y, otra vez, vuelve a suponer muchas y diferentes etapas o estratos. Sin embargo, las operaciones internas del ordenador están "ocultas": lo único que el usuario percibe es que su programa le está solicitando alguna información complementaria y está produciendo la salida requerida.

Debido a que el microprocesador no puede comprender un lenguaje de alto nivel, la primera tarea con la que se enfrenta el ordenador consiste en traducir las

Del problema al programa

El origen de un programa para ordenador comienza al comprender que hay un problema que debe ser resuelto: en este caso, cómo mantener en un nivel constante la temperatura de un invernadero. Para obtener la respuesta, este problema ha de pasar a través de diversas etapas de procesamiento que resultan en un programa completo



Surge el problema...

La idea se escribe en borrador en una hoja de papel

Se confecciona un diagrama de flujo para analizar el problema y desarrollar la estructura del programa...

Luego se traduce a algún lenguaje de ordenador, por ejemplo, BASIC

instrucciones en código de lenguaje máquina. En los ordenadores personales esta tarea la realiza el interpretador que está almacenado permanentemente en la ROM de la máquina.

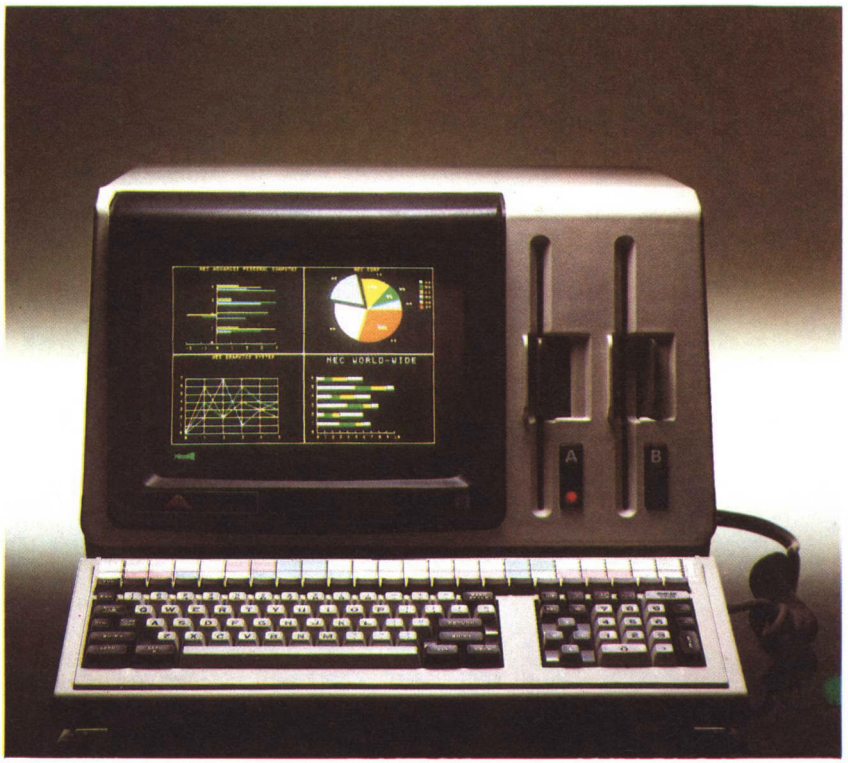
El interpretador es un sofisticado programa en código de lenguaje máquina que el microprocesador ejecuta directamente. Cuando se digita RUN, el interpretador comienza a examinar el programa del usuario, carácter por carácter. Compara todas las frases que encuentra con las de su propio diccionario. Si se encuentra con un carácter que no comprende (lo que puede suceder simplemente porque usted ha cometido un error de digitación), dejará de tratar de interpretar el programa e imprimirá en pantalla un mensaje indicando SYNTAX ERROR (error de sintaxis).

Si la palabra está incluida en el diccionario del interpretador (por ejemplo, PRINT), pasa inmediatamente a la parte del interpretador que sabe cómo tratar esta función. En este caso, la rutina examinará luego lo que venga a continuación de la palabra PRINT en el programa del usuario y preparará esta información para ser visualizada en forma de flujo de caracteres.

En este punto comienza a operar el siguiente nivel. En algún otro lugar de la memoria del ordenador hay una rutina que puede aceptar un flujo de caracteres, almacenarlos en otra zona de la memoria reservada para la pantalla y ordenarlos para que puedan ser convertidos en la clase de señales que necesita la pantalla de televisión o el monitor. Esto es algo que debe hacerse continuamente, incluso cuando el programa en sí mismo está comprometido sólo en cálculos.

Lo mismo puede decirse respecto al otro extremo del ordenador: el teclado. En el interior del ordenador, una rutina de programa escrita especialmente debe explorar el teclado para averiguar si han sido pulsadas algunas teclas y, en caso afirmativo, colocar los códigos adecuados en otra zona de la memoria para su utilización como la entrada del programa del usuario. Y, puesto que usted puede desear interrumpir en cualquier momento la realización del programa pulsando la tecla BREAK, el teclado debe ser explorado continuamente, aun mientras el programa se está ejecutando.

En realidad, el microprocesador que poseen la mayoría de los ordenadores personales no puede llevar a cabo más de una tarea a la vez, de modo que, efectivamente, ha de dividir su tiempo entre interpretar el programa del usuario y realizar sus propias funciones internas, tales como verificar el teclado y controlar la pantalla. Esta tarea la puede cumplir mediante un procedimiento "por interrupción"; un circuito electrónico especial interrumpe al microprocesador unas 50 veces por segundo y "le recuerda" que debe realizar sus tareas propias y otras funciones en la pantalla y el teclado, antes de continuar con lo que estaba haciendo.

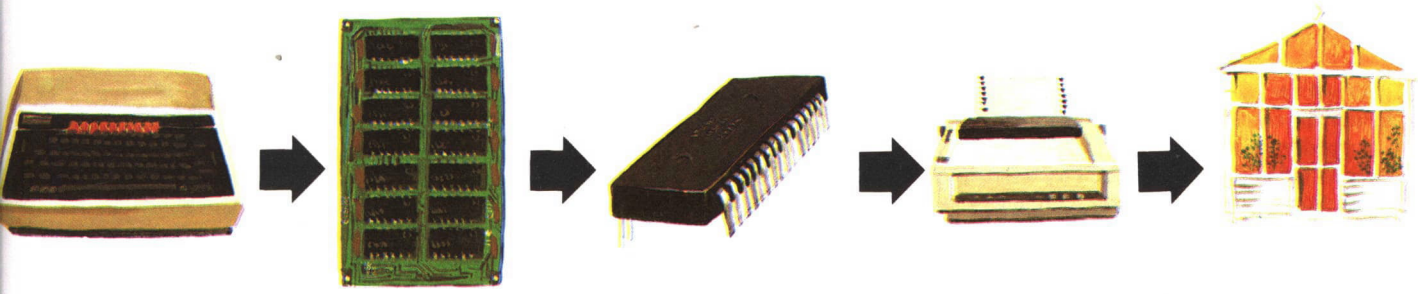


Así, pues, incluso cuando su programa haya sido digitado, el ordenador ha de llevar a cabo muchos niveles de procesamiento antes de producir los resultados. Si bien el proceso puede resultar complicado, es el ordenador quien se ocupa de la mayor parte del mismo.

La tendencia actual se orienta a facilitar la tarea del programador, haciendo recaer en el ordenador la realización de la mayor cantidad posible del trabajo de rutina. Los ordenadores de la próxima generación podrán escribir ellos mismos programas completos a partir de unas especificaciones en lenguaje corriente.

El software oculto

En todo ordenador existe una compleja jerarquía de software oculto que trabaja constantemente. Entre sus muchas tareas, controla y verifica cuándo ha sido pulsada una tecla y de cuál se trata, qué se está visualizando en pantalla, qué instrucciones se les están proporcionando a los periféricos, y el estado y contenido de la memoria RAM. Todas estas funciones se desempeñan de manera incesante mientras el usuario está concentrado en la siguiente función de su programa. El mismo principio de la jerarquía de software oculto rige tanto para los sofisticados ordenadores de gestión (como el de la fotografía) como para los ordenadores personales



El programa se introduce en la memoria del ordenador a través del teclado

El programa en BASIC pasa a través de una matriz de chips. Estos lo convierten en código de lenguaje máquina y lo retransmiten a la CPU

El cálculo se realiza en la CPU. La información resultante se transmite posteriormente a un periférico (p. ej., una impresora, un monitor, etc.)

En este caso la impresora produce una salida impresa o "copia impresa"

Si el programa es exacto, el problema ya está resuelto

Ian McKinnell

Procesador portátil
El tratamiento de textos se está convirtiendo rápidamente en una de las más difundidas aplicaciones del ordenador. Las nuevas máquinas se diseñan con configuraciones específicas para posibilitar esta función. Estas configuraciones incluyen pantallas de 80 columnas (para visualizar la anchura total de una carta mecanografiada), unidades de disco incorporadas (en ocasiones el precio incluye un disco para tratamiento de textos) y teclas de función programables que se emplean para manipular el texto. Algunas máquinas, como la Ajile que muestra la fotografía, son portátiles, ¡ideales para periodistas y ejecutivos de alto nivel!

El texto perfecto

Con un software adecuado, su micro se transforma en procesador de textos, editando y almacenando las palabras e, incluso, corrigiendo los errores ortográficos

El tratamiento de textos es una de las tareas más útiles que pueden realizarse con un microordenador. Pero la frase "tratamiento de textos" no explica, a nivel popular, el significado de esta poderosa herramienta. Ante esta expresión, la reacción normal suele ser preguntar: "¿Cómo se pueden procesar las palabras?"

Los anuncios del periódico ofrecen a menudo "procesadores de textos para su oficina". Lo que estos anuncios olvidaban informar era que el caro hardware de estos procesadores no era otra cosa que un microordenador adaptado específicamente para realizar programas de tratamiento de textos. Los procesadores de textos son menos flexibles que los microordenadores corrientes, porque sólo pueden realizar una tarea.

Quizá la clase de programa a la que alude la frase "tratamiento de textos" debería haberse expresado como "mecanografiado auxiliado por ordenador". Con la adición de una impresora, actualmente la mayoría de los ordenadores personales pueden ejecutar algún programa de tratamiento o edición de textos. Pero es necesario que el propietario de un ordenador trate por sí mismo de procesar textos, para que comprenda lo útiles que resultan este tipo de programas.

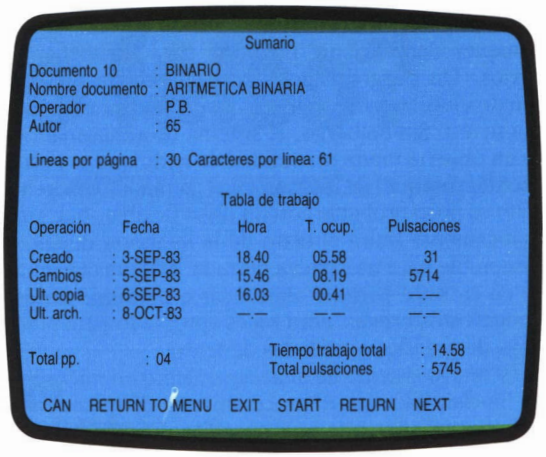
Utilizado como procesador de textos, un ordenador visualiza las palabras en pantalla tal como éstas son digitadas, del mismo modo en que, con una máquina de escribir, éstas se imprimen sobre un papel. Los microordenadores más grandes pueden visualizar 80 caracteres en la pantalla, que representan una "página". En los ordenadores más pequeños, el usuario necesita más paciencia. Debe arreglárselas con una pantalla mucho más estrecha y, en algunos modelos, se encuentra con la falta de letras minúsculas. También ha de tener presente que una máquina pequeña sólo

puede almacenar una cantidad limitada de texto.

El programa ofrece diversos y sofisticados medios auxiliares. Todos los programas de tratamiento de textos perciben el final de cada línea a medida que éste se acerca y automáticamente "dan la vuelta", bajando completa la última palabra y colocándola al comienzo de la línea siguiente. Esto significa que el mecanógrafo ya no ha de preocuparse por "retornar el carro" al final de cada línea. Por el contrario, podrá escribir de manera fluida y continua, puesto que el programa irá creando una línea nueva cada vez que sea necesario. No obstante, cuando el texto deba proseguir en punto y aparte, el mecanógrafo tendrá que pulsar la tecla RETURN.

En una máquina de escribir convencional, usted no tendrá otra alternativa que corregir los errores mecánicamente, por lo general borrando o tachando el error y volviendo a escribir encima, lo que resulta bastante incómodo. Cuando haya una o dos correcciones, las únicas opciones serán enviar una carta emborronada o escribirla de nuevo. El tratamiento de textos resuelve el problema. El cursor intermitente que aparece en pantalla le indica, como siempre, su ubicación en cada momento. Usted deberá desplazar el cursor sobre las palabras ya escritas hasta el punto donde haya una errata. Entonces podrá hacer que el error desaparezca y volver a digitar en la forma correcta.

Una vez usted esté enterado del caudal de ayuda que puede prestarle un programa de tratamiento de textos, se sentirá motivado para prepararlos con mayor detenimiento. Por ejemplo, puede utilizar la orden INSERT para agregar una palabra, una oración entera o un párrafo, con la misma facilidad que si se tratara de una sola letra. Esto estimula al usuario para reconsiderar lo que está redactando en la carta o documento. La instrucción para eliminar texto es igualmente sencilla. Una orden hace que las palabras y letras que se desea suprimir desaparezcan de la pantalla y el texto restante se cierre, devolviendo a la página su aspecto impecable. Muchos escritores y periodistas profesionales en la actualidad utilizan procesadores de textos y, en general, opinan que su trabajo ha mejorado cualitativa y cuantitativamente.

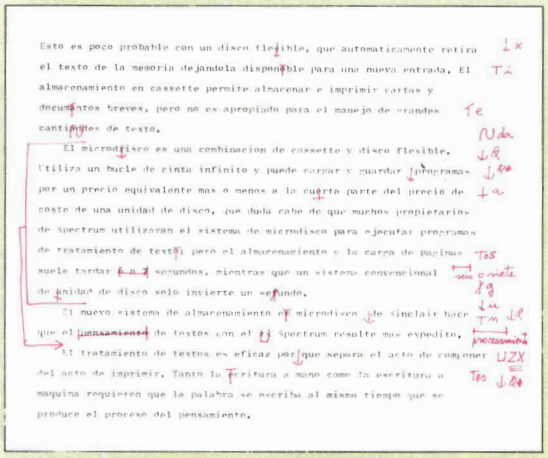


Estudiando el menú
La fotografía muestra el "menú" de un sofisticado paquete de programas para tratamiento de textos. El menú aparece en la pantalla apenas usted inserta el software, y le ofrece una guía sobre las diversas funciones de edición disponibles en el tratamiento de textos. Son ejemplos de funciones de edición: tabulación y determinación de márgenes, separación de las líneas, enumeración de palabras que integran el texto, reacomodación de párrafos y confección de un índice

Hasta los microordenadores más pequeños pueden ofrecer cierto potencial para tratamiento de textos. El Sinclair ZX81 puede operar con un programa sencillo para edición de textos, que le permite al usuario escribir una carta o un documento en pantalla y luego corregirlo. La frase "edición de textos" se aplica generalmente a un programa limitado para tratamiento de textos, que puede trabajar con una o dos páginas pero que es incapaz de tratar y almacenar documentos más extensos. La pequeña RAM del ZX81 y de los ordenadores de dimensiones similares limita drásticamente la cantidad de texto que puede visualizarse y con la cual es posible trabajar.

Una de las desventajas del ZX81 es que posee un teclado muy sensible al tacto, que impide una digitación rápida. Aunque el teclado del ZX Spectrum es bastante mejor, no llega a ser de la clase a la que están habituados los mecanógrafos. Si usted está estudiando la posibilidad de adquirir un micro para utilizarlo como procesador de textos, sería muy interesante que examinara los tipos de teclado, porque éstos tienen decisiva influencia en su comodidad de uso y en la velocidad de las pulsaciones.

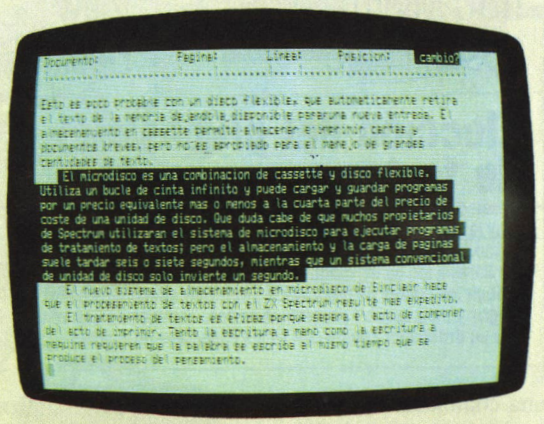
No obstante, un ordenador de 16 Kbytes o de 32 Kbytes puede resultar bastante útil para tratamiento


Esto es poco probable con un disco flexible, que automaticamente retira el texto de la memoria dejandola disponible para una nueva entrada. El almacenamiento en cassette permite almacenar e imprimir cartas y documentos breves, pero no es apropiado para el manejo de grandes cantidades de texto.

El microdisco es una combinacion de cassette y disco flexible. Utiliza un bucle de cinta infinito y puede cargar y guardar programas por un precio equivalente mas o menos a la cuarta parte del precio de coste de una unidad de disco. Que duda cabe de que muchos propietarios de Spectrum utilizaran el sistema de microdisco para ejecutar programas de tratamiento de textos; pero el almacenamiento y la carga de paginas suele tardar seis o siete segundos, mientras que un sistema convencional de unidad de disco solo invierte un segundo.

El nuevo sistema de almacenamiento en microdisco de Sinclair hace que el procesamiento de textos con el ZX Spectrum resulte mas expedito.

El tratamiento de textos es eficaz porque separa el acto de componer del acto de imprimir. Tanto la escritura a mano como la escritura a maquina requieren que la palabra se escriba al mismo tiempo que se produce el proceso del pensamiento.



Edición electrónica

La principal ventaja de un sistema de tratamiento de textos por ordenador respecto a la máquina de escribir, es la asombrosa adaptabilidad y velocidad que aquél ofrece. Mientras que los textos mecanografiados implican una laboriosa edición y corrección, mediante el tratamiento de textos puede hacerse lo mismo pero a velocidad electrónica. Y todas estas operaciones se realizan mientras el texto es visualizado en pantalla. Un sistema sofisticado puede buscar en el texto, cambiar palabras, trasladar líneas de un sitio a otro, revisar el texto automáticamente e, incluso, corregir su redacción y su ortografía. De hecho, un procesador de textos puede ahorrar tanto tiempo que gradualmente está comenzando a reemplazar a las máquinas de escribir convencionales. Compare las desordenadas correcciones del texto mecanografiado con la pulcra edición electrónica del procesador de textos

de textos. Después de comprar una impresora, el problema siguiente, en cuanto a hardware, consiste en disponer de suficiente memoria para almacenar sus textos. Un programa para tratamiento de textos se puede ejecutar utilizando una cassette para el almacenamiento. Sin embargo, el sistema de almacenamiento en cassette limita la cantidad de texto que es posible escribir porque la memoria se colma rápidamente. Esto es poco probable con un disco flexible, que automáticamente retira el texto de la memoria dejándola disponible para una nueva entrada. El almacenamiento en cassette permite almacenar e imprimir cartas y documentos breves, pero no es apropiado para el manejo de grandes cantidades de texto.

El nuevo sistema de almacenamiento en microdisco de Sinclair hace que el procesamiento de textos con el ZX Spectrum resulte más expedito. El microdisco es una combinación de cassette y disco flexible. Utiliza un bucle de cinta infinito y puede cargar y guardar programas por un precio equivalente más o menos a la cuarta parte del coste de una unidad de disco. Qué duda cabe de que muchos propietarios de Spectrum utilizarán el sistema de microdisco para ejecutar programas de tratamiento de textos; pero el almacenamiento y la carga de páginas suele tardar seis o siete segundos, mientras que un sistema convencional de unidad de disco sólo invierte un segundo.

El tratamiento de textos es eficaz porque separa el acto de componer del acto de imprimir. Tanto la escritura a mano como la escritura a máquina requieren que la palabra se escriba al mismo tiempo que se produce el proceso del pensamiento. En el caso del tratamiento de textos, en el papel no aparece ninguna palabra hasta que la composición en pantalla sea correcta. Pero llevar las palabras al papel exige una impresora, y el tipo de impresora barata apta para el listado de programas para ordenador es poco probable que pueda realizar una copia satisfactoria de ciertos documentos escritos como pueden ser una carta, un artículo o un poema.

Existe una gran competencia entre los fabricantes por producir sistemas baratos de tratamiento de textos, y ahora comienzan a ofrecerse algunos programas bastante económicos para ordenadores personales. Éstos almacenan las instrucciones de los programas para tratamiento de textos en un chip que puede enchufarse en la placa de circuito impreso del ordenador. Son muy útiles cuando no se dispone de una unidad de disco. Ofrecen la ventaja de que el programa puede cargarse rápidamente y utilizarse al instante, eliminando la espera que implica tener que cargarlo a través de un disco o de una cinta de cassette. Si la RAM del ordenador es suficientemente grande (a partir de 32 Kbytes), usted podrá escribir un texto de hasta 5 000 palabras y editarlo de la manera que le resulte más conveniente.

Si desea almacenar el texto editado después de haberlo impreso, necesitará guardarlo en una cinta de cassette, proceso que llevará algunos minutos. El software de tratamiento de textos barato no puede almacenar el texto que usted haya redactado. Si desea escribir una novela mediante un programa de tratamiento de textos, necesita conocer la capacidad del programa para manejar grandes cantidades de palabras.

Algunos sofisticados programas de tratamiento de textos pueden realizar funciones extras muy útiles. Una de las más populares es el diccionario automático o verificador de ortografía. Para utilizarlo necesitará un sistema de disco. El diccionario compara las palabras del texto con las que él tiene almacenadas. Señala las palabras que no reconoce y sugiere corregirlas.

La gran difusión del tratamiento de textos hace suponer que en el futuro será un ingenio esencial en la oficina y un dispositivo ideal para la correspondencia.

MICROORDENADOR	SOFTWARE	EN CASSETTE	EN DISCO	TECLADO MAQ. ESCRIBIR
Sinclair Spectrum	Wordprocessor	●	—	—
Commodore Vic 20	Vicwriter	●	—	●
Dragon 32	Gemini	●	—	●
New Brain	Propen	●	—	—
Commodore 64	Visawrite	—	●	●
Atari 800	Keyword	—	●	●
BBC Micro	View	—	●	●
Epson HX20	Deckmaster	●	—	●

Elección del procesador de textos

Si usted piensa utilizar su procesador de textos durante períodos muy largos, asegúrese de que podrá digitar cómodamente en el teclado de su ordenador. Los teclados de tipo membrana, similares a los de las máquinas de calcular, se desarrollaron para reducir los costos de fabricación y son más adecuados para juegos y para escribir programas cortos. También es una ventaja que su ordenador cuente con teclas especiales de función programables. Éstas se emplean con frecuencia en los paquetes de programas para tratamiento de textos más sofisticados, y reducen la cantidad de órdenes que usted debe digitar en su teclado. Obviamente, si desea copias impresas es indispensable comprar una impresora. Las impresoras varían notablemente en cuanto a calidad de impresión y velocidad de ejecución. Asegúrese de que su procesador de textos y su impresora sean compatibles para las diversas tareas que les asigne. Si lo que usted desea es escribir cartas, puede interesarle una impresora y un procesador de textos de precio razonable. Pero para textos más largos necesitaría una combinación más cara

Términos clave

¿Verdadero o falso?

Quizá los ordenadores aún no sepan "pensar"; pero saben seguir las leyes de la lógica

Acerca de la CPU (unidad central de proceso) suele decirse que es el corazón del ordenador. Es el lugar donde se efectúan todos los cálculos y donde se toman todas las decisiones lógicas. Pero, ¿cómo se efectúan estos cálculos y cómo se adoptan las decisiones?

Para llegar a comprenderlo, es preciso conocer los principios básicos de la aritmética binaria y familiarizarse con las puertas lógicas. En los ordenadores, estas puertas son circuitos eléctricos simples que pueden adoptar decisiones lógicas y establecer comparaciones lógicas. Esto parece más complicado de lo que en realidad es, y los principios pueden ilustrarse a partir de ejemplos tomados de la vida cotidiana.

Existen tres tipos de puertas fundamentales: la puerta AND (y), la puerta OR (o) y la puerta NOT (no), que se escriben con letras mayúsculas para diferenciarlas de las palabras normales en inglés "y", "o" y "no".

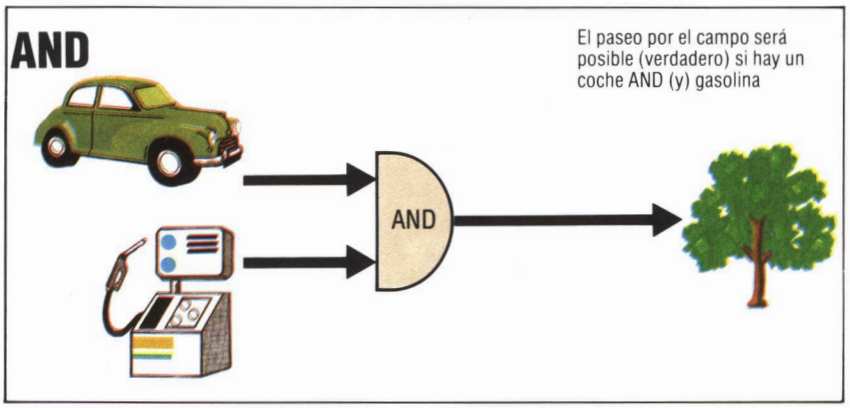
Conexiones lógicas

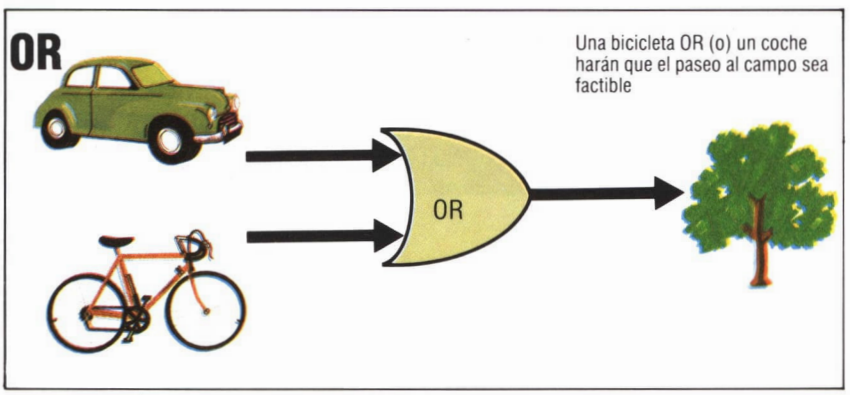
Una puerta AND es un circuito que produce una salida "verdadera" si todas las entradas son "verdaderas". Veamos qué significa esto. Supongamos que quisiera dar un paseo al campo. Si tiene un coche AND (y) gasolina, podrá dar ese paseo. No podría darlo si tuviera gasolina pero no poseyera un coche. Igualmente, tampoco sería factible dicho paseo si tuviera coche pero no gasolina.

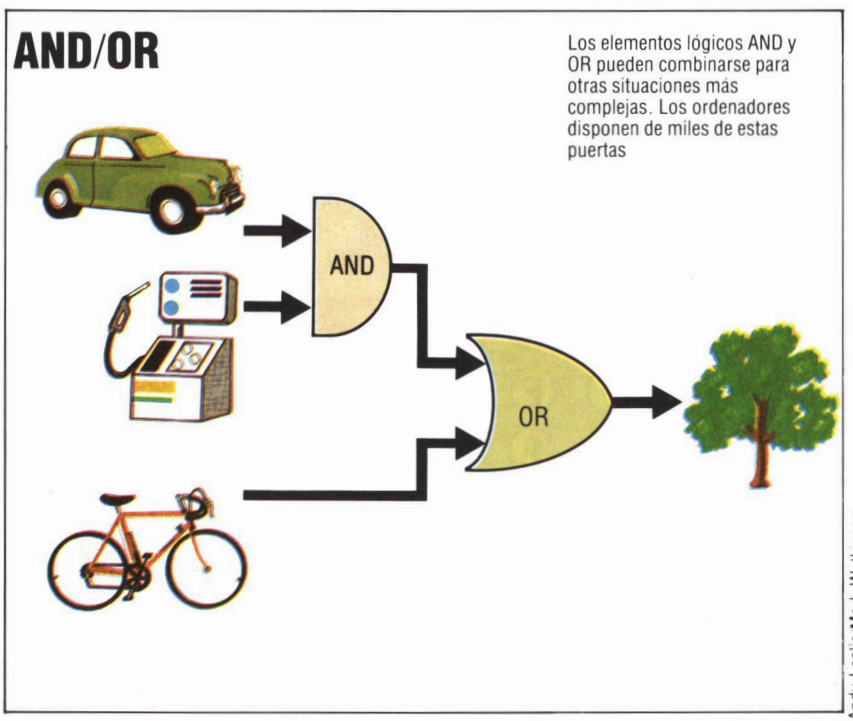
En este "circuito" AND hay dos condiciones de entrada y ambas han de ser "verdaderas". Para que pueda dar el paseo (la "salida") ha de ser verdad que posee un coche AND (y) ha de ser verdad que tiene gasolina. Entonces la salida se convierte en "verdadera": es verdad que puede dar un paseo por el campo. Más adelante veremos cómo este diagrama lógico puede expresarse como una ecuación lógica, y cómo puede representarse en una "tabla de verdad".

Imaginemos una situación ligeramente diferente. A alguien le agradaría dar un paseo por el campo. El paseo será posible si esa persona posee un coche OR (o) una bicicleta (esta vez damos por sentado que el coche tiene gasolina). Si posee un coche, podrá ir de paseo. Si tiene una bicicleta, podrá ir. Sólo en el caso de que ninguna de las condiciones de entrada fuese verdadera sería imposible el paseo; en el lenguaje de informática, la salida se convierte en falsa (es decir, no es verdad que esta persona pueda ir de paseo al campo).

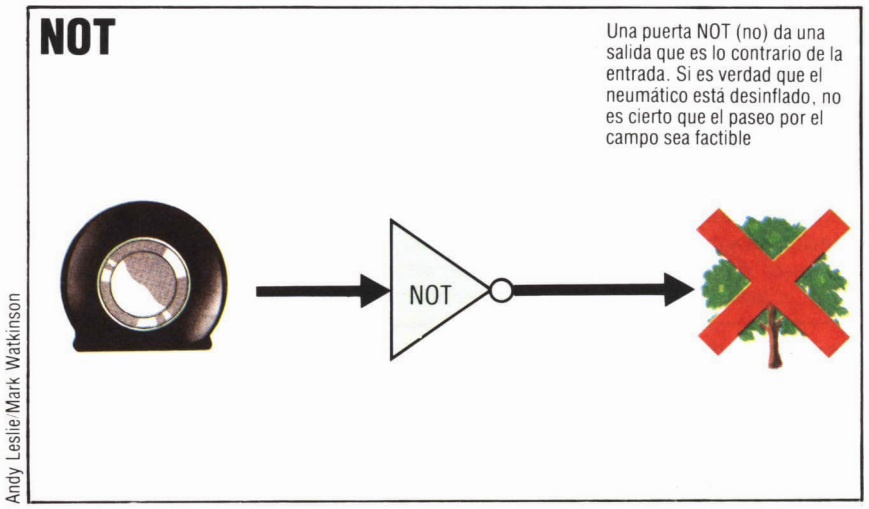
Nos queda por considerar otra puerta lógica esencial: la puerta NOT (no). Esta puerta simplemente da como salida lo contrario de la entrada. Si la entrada es verdadera, la salida será falsa. Si la entrada es falsa, la salida será verdadera. Extendiendo nuestra metáfora del paseo al campo, ya sea en coche o en bicicleta, debe ser falso que tenemos un neumático desinflado para dar dicho paseo. Si la entrada (neumático desinflado) fuera verdadera, la salida (paseo) sería falsa.







Andy Leslie/Mark Watkinson



Estos elementos lógicos pueden combinarse entre sí y lo hemos demostrado valiéndonos del ejemplo del paseo por el campo. Las combinaciones entre AND (y), OR (o) y NOT (no) permiten tomar todas las decisiones sobre la base de la lógica convencional. Es interesante elaborar las decisiones lógicas (puertas) que se necesitarían para otros problemas. Trate de elaborar, por ejemplo, lo que se requeriría para hacer una barbacoa en el jardín. Puede resultar bastante complicado. Para tener una barbacoa en el jardín (la salida verdadera) necesitaríamos varias condiciones de entrada: dinero OR (o) un talonario de cheques OR (o) una tarjeta de crédito (para comprar la comida y la bebida) AND (y) una tarde libre AND (y) buen tiempo AND (y) una parrilla AND (y) carbón.

Las tablas de verdad

Los símbolos que hemos usado en las ilustraciones son los mismos que se utilizan en los diagramas de los circuitos de ordenador. Para ver con qué facilidad se pueden realizar decisiones lógicas utilizando circuitos eléctricos, observemos la "tabla de verdad" para la ilustración AND. Si empleamos la letra *c* para representar la condición de entrada "tener un coche" y la letra *g* para la condición de entrada "tener gasolina", podemos representar la condición de salida "dar un paseo por el campo" empleando la letra *p*. Luego podremos utilizar la letra V para representar "verdadero" y la letra F para representar "falso". La tabla de verdad muestra todas las combinaciones posibles de condiciones de entrada y los efectos de la utilización de AND en la salida. Sería así:

COCHE	F	V	F	V	(c)
GASOLINA	F	F	V	V	(g)
PASEO	F	F	F	V	(p)

Tabla de verdad para dos entradas AND (y)

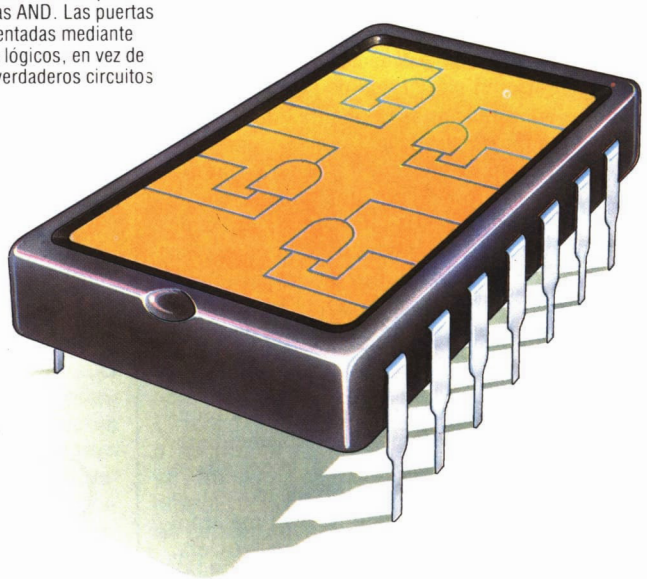
COCHE	0	1	0	1	(c)
GASOLINA	0	0	1	1	(g)
PASEO	0	0	0	1	(p)

La misma tabla de verdad utilizando 0 y 1 para "falso" y "verdadero"

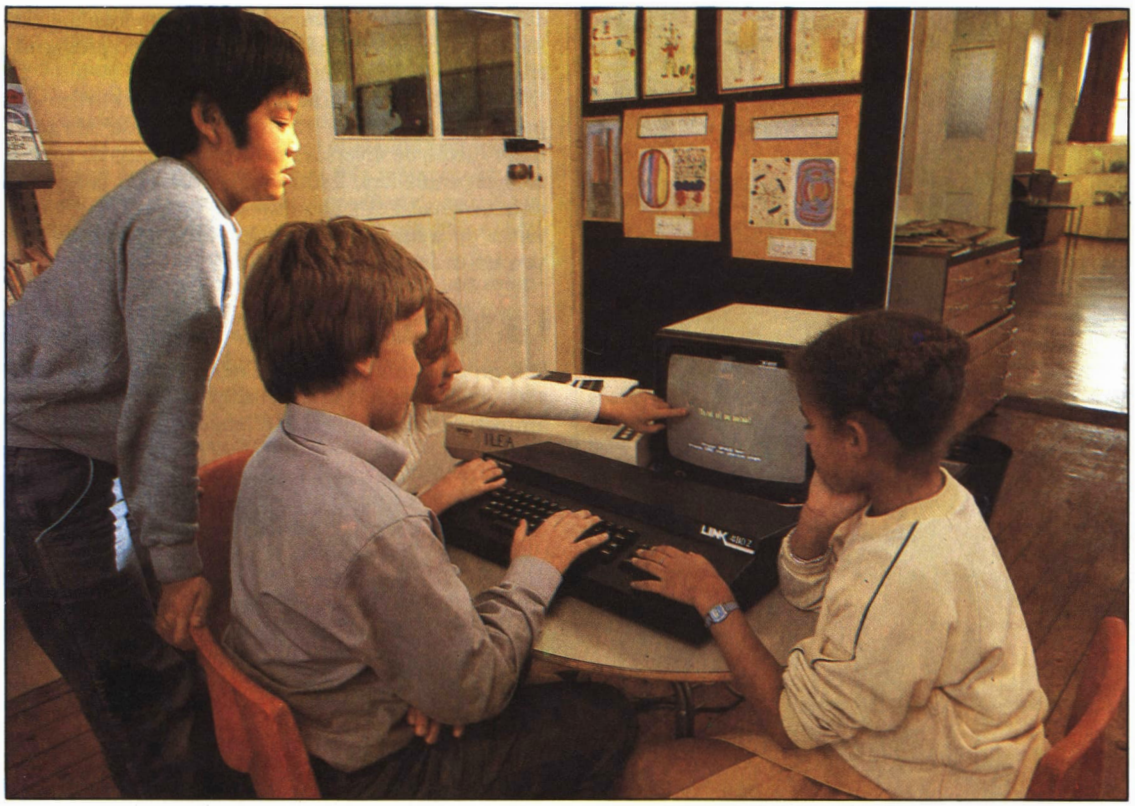
En los ordenadores utilizamos los dígitos binarios cero y uno para representar, respectivamente, falso y verdadero. El ordenador interpreta un voltaje positivo como un uno y un voltaje negativo como un cero. Un circuito AND puede montarse fácilmente utilizando transistores, de manera que si ambas entradas son voltajes positivos, la salida también será un voltaje positivo. Si una o ambas entradas son de un voltaje negativo, la salida del circuito será igualmente negativa.

Un circuito electrónico OR produce una salida de voltaje positiva si una o ambas entradas son positivas. Si ambas entradas son negativas, la salida también será negativa. En un circuito NOT, la entrada simplemente se invierte: si la entrada es positiva, la salida será negativa; si la entrada es negativa, la salida será positiva.

El chip 7408
Los grandes chips de silicio suelen contener miles de puertas para efectuar la lógica AND, OR y NOT. El pequeño chip 7408 que aquí vemos contiene todos los transistores y los sistemas de circuitos para cuatro puertas AND. Las puertas están representadas mediante los símbolos lógicos, en vez de mostrar los verdaderos circuitos



Escuela en la pantalla



Ian Dobbie

El ordenador en la enseñanza básica
Diversos países se están preparando para el futuro mediante la aplicación de una política educativa que tiende a familiarizar a los niños con los ordenadores ya durante sus primeros años de escolarización. No se trata tan sólo de familiarizar al niño con la nueva tecnología, sino también de utilizar el ordenador para enseñar una gran variedad de temas, desde biología hasta lenguas extranjeras. Un ordenador es un maestro paciente e ideal, ya que no introduce un nuevo concepto hasta que el niño no domine por completo el problema anterior, permitiendo que tanto el alumno de comprensión lenta como el más despierto aprendan al ritmo de sus propias posibilidades. Como medio auxiliar de aprendizaje, el ordenador ofrece una gama de aplicaciones más amplias: por el solo hecho de utilizarlo, el niño descubre cómo se analiza y se resuelve un problema

La era de los ordenadores ya ha comenzado en las aulas, y existe en el mercado una fascinante y variada gama de programas educativos para niños de todas las edades

A finales de 1983 había en España más de 4 000 ordenadores personales dedicados a la enseñanza, y se estima que dentro de pocos años la mayoría de los centros de educación primaria y secundaria, así como también los de formación profesional, contarán al menos con un microordenador. En la actualidad los ordenadores se utilizan no sólo para impartir "nociones de informática", sino también, en el campo educativo, para enseñar temas relacionados con las matemáticas y la alfabetización, ayudar a los niños de comprensión lenta y estudiar lenguas extranjeras.

En el mercado existen muchos programas educativos para ordenadores personales, pero los maestros suelen quejarse de su poca calidad. Ello se debe a que son muy pocos los programas que se han escrito respetando por igual tanto la disciplina educativa como la informática.

Es muy raro que un programador de ordenadores posea experiencia en el campo de la enseñanza; y los maestros, muchos de los cuales apenas iniciados en este campo, incurren a veces en los más garrafales errores de programación.

Aunque lo más probable es que el programa de un maestro sólo se aplique en su propia clase, los problemas surgen en cuanto dicho programa se envía a otra escuela. Por lo general, el programa en sí mismo, almacenado ya sea en cassette o en disco, no es suficiente, haciéndose imprescindible una buena documentación explicativa, pues sin ella es posible que los estudiantes sean incapaces de hacer funcionar el programa. Ante este problema, tal vez la respuesta del programador fuera: "¡Por supuesto, *se entiende* que se ha de digitar LOAD (cargar)!", pero a quien no sabe nada acerca de los ordenadores se le han de explicar uno por uno todos estos detalles.

Pasando a un plano más concreto, una buena programación requiere anticiparse a todos los errores que pueda cometer un principiante. Esto es muy importante para asegurar que el programa sea, efectivamente, un buen medio auxiliar de enseñanza. Una buena programación consiste en algo más que eliminar de un programa todos los posibles márgenes de error, hasta el punto de que sólo ejecute lo que debería hacer exclusivamente cuando se pulse la tecla adecuada. La buena programación también debe asegurar que el programa no hará nada que no deba realizar cuando se pulse una tecla equivocada. Ésta es la parte más complicada de la escritura de un programa. Éste ha de ser capaz de recuperarse de los más flagrantes errores en que pueda incurrir un niño y, al mismo tiempo, inculcarle la idea de que usar un ordenador es algo sencillo y divertido.

A pesar de estos problemas, existe una amplia gama de programas educativos tanto para la escuela como

para el hogar. Un ordenador es una maravillosa herramienta educativa y, para acertar en la elección del software más adecuado para sus hijos, le será de gran utilidad conocer las diversas maneras en que se puede emplear.

Un ordenador puede utilizarse para instruir al niño en prácticamente cualquier tema. Si el programa es bueno, lo más probable es que el niño se sienta fascinado y muy motivado para aprender.

La clase de programas educativos más generalizada se conoce como "de procedimiento correcto y ejercicios". En un programa de este tipo, al niño se le proporcionan ejemplos y luego se le invita a resolver problemas similares. Normalmente el programa lleva la cuenta de los fallos y los aciertos del niño. Incluso estimula al alumno cuando éste da una respuesta correcta y le sugiere amablemente "inténtalo otra vez" cuando la respuesta es equivocada.

Para decidir qué programas son los más adecuados para su hijo, deberá tener en consideración diversos factores: la edad del niño, el modelo de ordenador que usted haya adquirido y lo que su hijo esté estudiando en la escuela.

Si aún no ha comprado un ordenador personal, pero piensa que cuando lo haga una de las funciones a que lo destinará en su hogar será la educación, valdrá la pena averiguar qué tipo de ordenador usa su hijo en la escuela. Si usted está en condiciones de comprar un modelo similar, su hijo podrá realizar en casa los mismos programas educativos que utiliza en el colegio. A muchas escuelas les interesa proporcionar a los padres copias de los programas que se emplean en clase, y estos "deberes" pueden representar una considerable ayuda. Si usted ya ha adquirido un ordenador personal y éste no es compatible con los ordenadores de la escuela, no se preocupe; la experiencia que adquiera su hijo al digitar ordenadores diferentes también será muy valiosa.

Es natural que la mayoría de los programas educativos existentes hayan sido creados para las marcas de ordenadores más competitivas del mercado, pero muchos fabricantes están potenciando el papel docente del ordenador. Existe una gama particularmente amplia de programas educativos para el Apple, el Commodore PET, el Tandy, el BBC Micro, Sinclair y Texas Instruments; no obstante, algunos de los fabricantes de software incorporados recientemente a este mercado aún deben aportar una oferta de programas realmente más variada.

Los programas educativos para cualquier ordenador pueden conseguirse a través de la empresa fabricante o por medio de diversas firmas productoras de software independientes. Estas últimas escriben programas para ordenadores y anuncian sus productos en las revistas de informática y en los establecimientos que venden ordenadores personales.

La elección adecuada

Los programas destinados a niños menores de ocho años se centran en la enseñanza de las operaciones aritméticas básicas y en la alfabetización. Una de las gamas más interesantes de programas educativos para niños pequeños es la que produce Texas Instruments. El ordenador personal TI-99/4A, por ejemplo, fue muy bien acogido por los padres norteamericanos y británicos, pues contaba con una amplia gama de programas educativos de TI producida tanto por Texas Instruments como por Scott, Foresman & Co., de Estados Unidos. El 99/4A es un ordenador de 16 bits, lo que significa que probablemente los programas escritos en código de lenguaje máquina sean mucho mejores que los programas escritos para la mayoría de los ordenadores personales corrientes de 8 bits.

Esto lo demuestran algunos programas de TI tales como *Beginning grammar* (Iniciación a la gramática), *Addition and subtraction* (Suma y resta) y *Number magic* (La magia de los números). Estos programas van almacenados en un cartucho plástico que se introduce en el TI-99/4A, por lo cual a los niños les resultan muy manejables. No obstante, el hecho de que el software sea norteamericano representa un problema, y

El mejor compañero
El microordenador ayuda a resolver los "deberes", acompaña en los ratos de ocio y proporciona amigos. Pero para obtener un buen rendimiento de esta maravillosa máquina, es necesario acertar en la elección de los programas adecuados de acuerdo con la edad de los niños

algunos de los programas suponen un gran inconveniente para los maestros de otros países. Sin embargo, TI posee un magnífico ejemplo de LOGO, si bien realmente se encuadra en la segunda categoría de herramientas de descubrimiento.

Para niños de más edad se cuenta con un software muy rico y variado. Los programas para edades entre 8 y 11 años se caracterizan por su complejidad y calidad, y la mayoría de ellos tienen como objetivo reforzar los conocimientos básicos del alumno y poner a prueba sus aptitudes. A esta edad el niño comienza a interesarse por otros temas, como la música y las lenguas extranjeras, que pueden enseñarse, asimismo, por ordenador. Existen programas de esta naturaleza para la mayoría de los aparatos.

Para alumnos de enseñanza media existe una multitud de programas. Para hacer una selección de ellos y escoger sólo los mejores, lo más indicado es que hable con el profesor de su hijo.

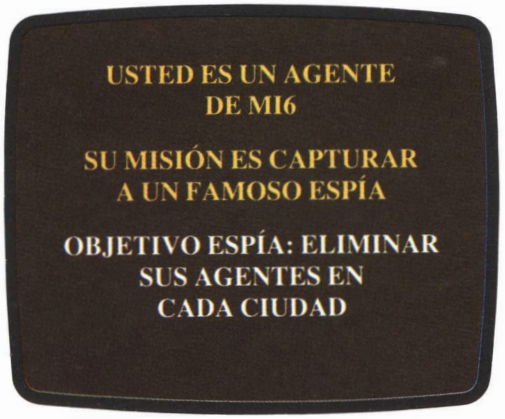
Es muy importante que el trabajo escolar que realice el niño en su casa siga la misma línea que el que lleva a cabo en la escuela; la mayoría de los profesores se mostrarán bien dispuestos para orientar y ayudar a los padres en la elección del tipo de software educativo más apropiado para sus hijos.

Existe aún otra categoría de programas educativos para ordenador, destinada fundamentalmente a los niños menores de 13 años. A esta edad los niños aún están descubriendo la forma de aprender y para ello resultan muy valiosos los programas que los introducen en el uso del ordenador como medio de descubrir el mundo por sí mismos. El programa más conocido de esta categoría es el LOGO, lenguaje del cual existen versiones para ordenadores de las firmas Atari, Tandy, Apple, Texas Instruments, Research Machines, Commodore e IBM. Por su parte, Sinclair y BBC han mostrado interés, también, por realizar sus versiones, pero aún no se encuentran en el mercado. A través de este programa, el niño entre 6 y 12 años de edad es estimulado para explorar el potencial gráfico del ordenador (y, a su vez, la geometría), mediante una "tortuga". El niño descubre cómo enseñarle a la tortuga a recordar procedimientos (programas) y, en determinados casos, puede ir progresando hasta llegar a dibujar en la pantalla todo un mundo de fantasía. Al utilizar estos programas, lo que el niño hace realmente es enseñarse a sí mismo las leyes básicas de las matemáticas y, en este sentido, se ha reconocido claramente la efectividad de este programa para ayudar a comprender conceptos matemáticos y espaciales.

Escoger un buen programa educativo no es una labor fácil, debido a la gran cantidad de software disponible. Es muy interesante asistir a alguna de las ferias de informática que se celebran periódicamente en el país. Con toda probabilidad, allí se encontrará con los propios fabricantes y programadores exhibiendo sus productos y, de este modo, tendrá la ocasión de conocer, probar y comparar programas.

Existe la opinión generalizada de que la actual escasez de buenos programas, que satisfagan tanto las exigencias relativas al ordenador como a la educación en sí misma, no ha de durar mucho tiempo más. Todos los meses aparecen programas nuevos que, con toda seguridad, serán un valioso estímulo para el desarrollo intelectual de sus hijos. Conviene tener presente a este respecto que, en un próximo futuro, la informática será indispensable en todos los campos de la actividad humana.

El Agente Secreto

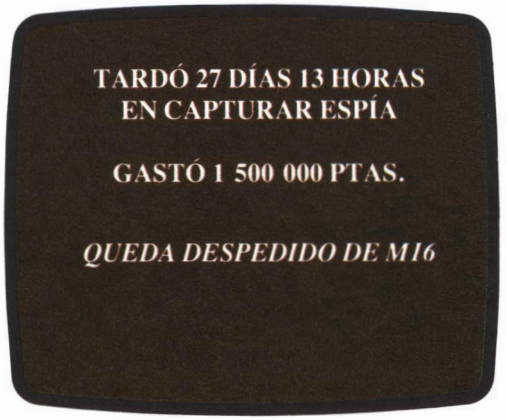


Actualmente existen muchos programas educativos en el mercado. El que muestran las fotografías se denomina *Agente Secreto* y ha sido realizado por Heinemann. Su misión consiste en capturar a un famoso espía enemigo que se halla en Europa, antes de que él elimine a todos sus agentes. Para poder atraparlo, se han de descifrar las claves acerca de su paradero. Comienza la búsqueda...

Al espía sólo se le puede capturar en una ciudad; pero él no permanece en el mismo sitio durante más de dos horas. Cuando cree saber dónde se halla éste, usted tiene la alternativa de dirigirse hasta allí en tren o en avión. Deberá decidir qué es más importante: llegar rápido o el precio a pagar

Cada vez que alguno de los agentes que trabajan para usted le envíe un mensaje, se encenderá una luz intermitente en el mapa que le indicará desde qué ciudad proviene. Si su agente fuera eliminado, el mensaje será interceptado antes de que usted lo reciba. En ese caso, tal vez le interese contratar los servicios de un nuevo agente; pero entonces habrá de pagar por él...

Los informadores estarán felices de venderle información, pero ésta se envía en clave, por lo cual quizá le interese entregársela a un especialista para que le ayude a descifrarla. Al final del juego habrá aprendido los nombres y la situación de las más importantes ciudades europeas, entenderá los anuncios de salidas y llegadas en estaciones y aeropuertos y será un experto en la confección de presupuestos



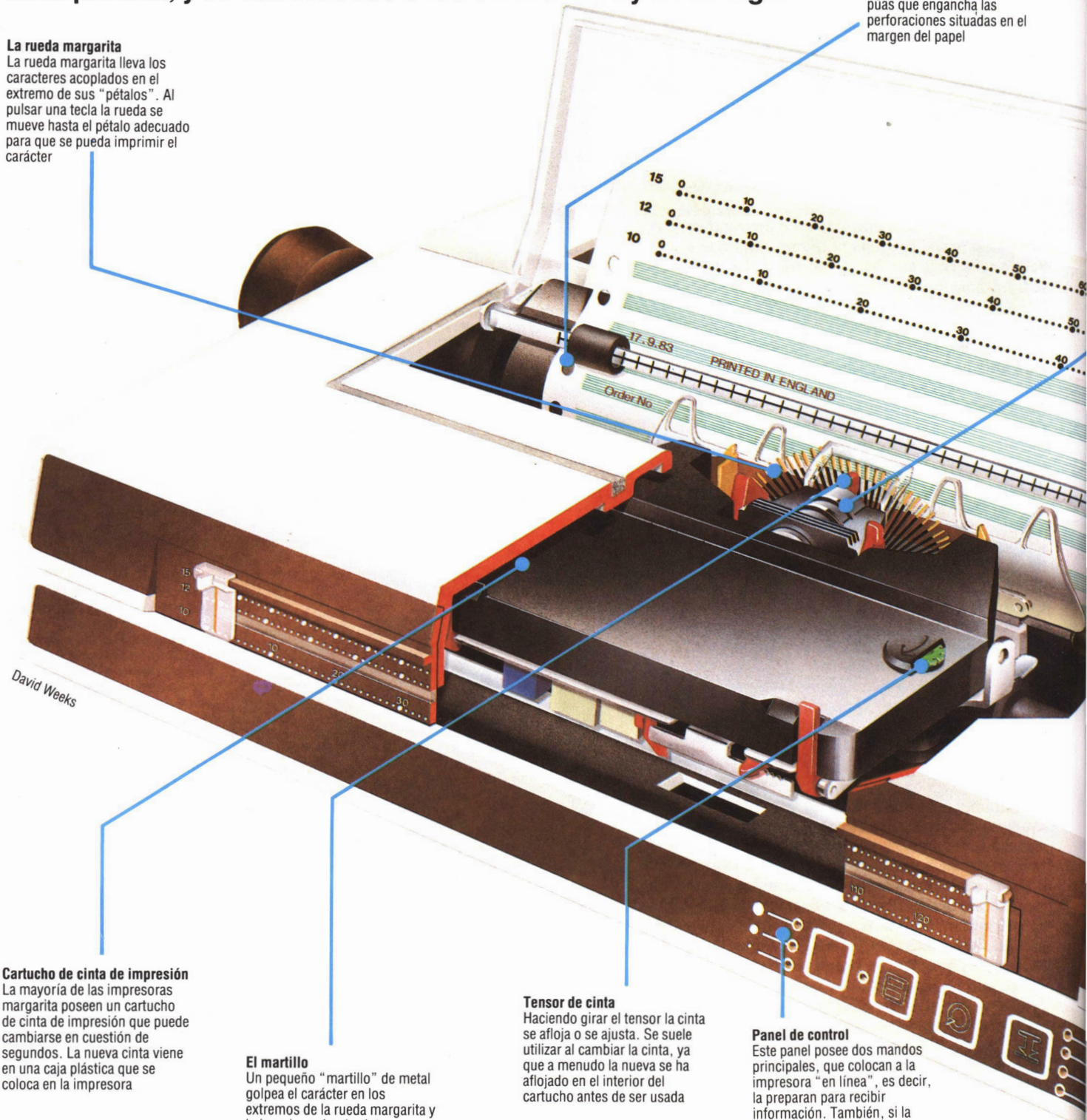
Ian McKinnell

Creadas para imprimir

La impresora margarita, la impresora de chorro de tinta y la impresora matricial son el más reciente avance en la tecnología de impresión, y se van introduciendo en la oficina y en el hogar

La rueda margarita
La rueda margarita lleva los caracteres acoplados en el extremo de sus "pétalos". Al pulsar una tecla la rueda se mueve hasta el pétalo adecuado para que se pueda imprimir el carácter

Guía del papel
La impresora se provee de papel mediante un rodillo rotatorio con púas que engancha las perforaciones situadas en el margen del papel



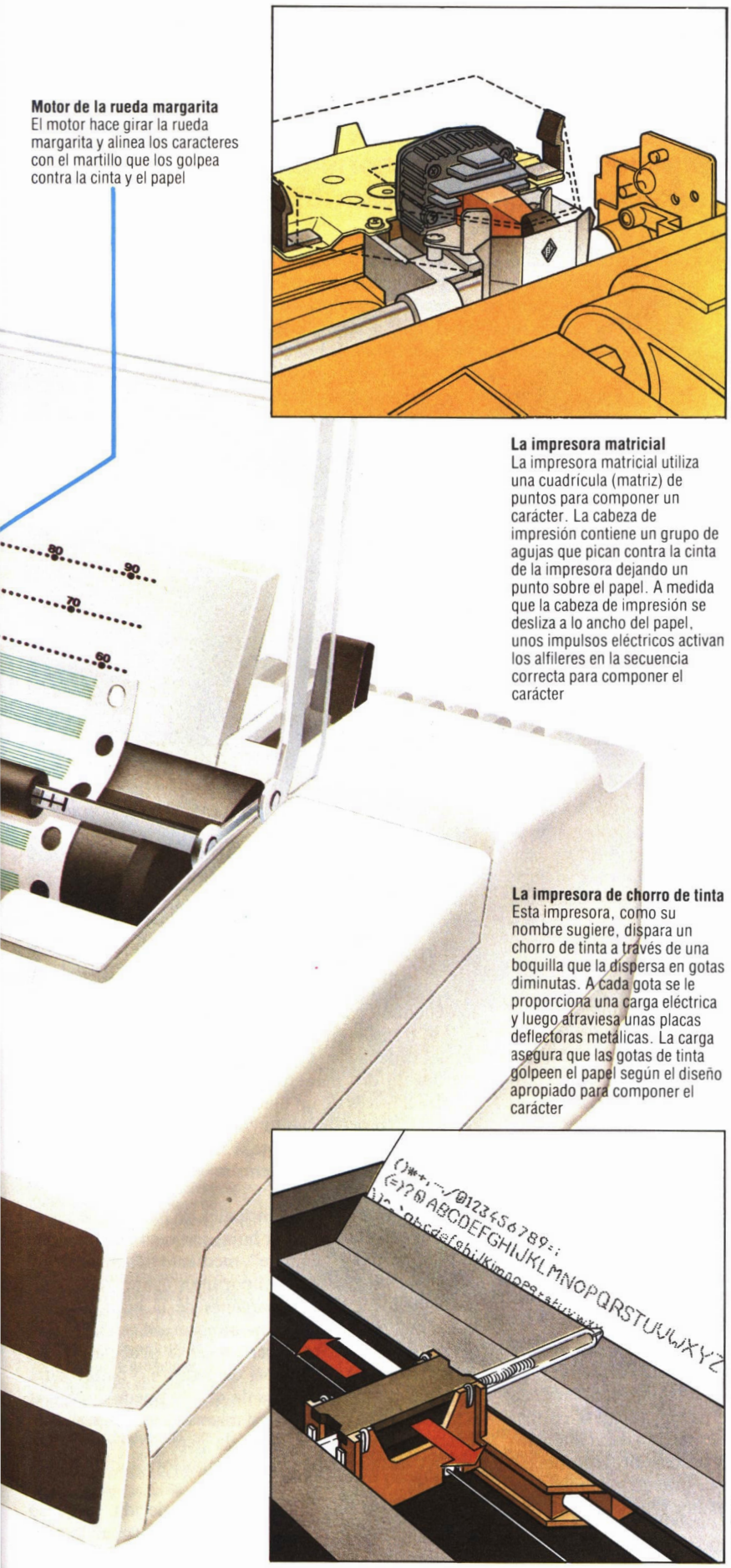
David Weeks

Cartucho de cinta de impresión
La mayoría de las impresoras margarita poseen un cartucho de cinta de impresión que puede cambiarse en cuestión de segundos. La nueva cinta viene en una caja plástica que se coloca en la impresora

El martillo
Un pequeño "martillo" de metal golpea el carácter en los extremos de la rueda margarita y lo impulsa sobre la cinta, estampando la impresión en el papel

Tensor de cinta
Haciendo girar el tensor la cinta se afloja o se ajusta. Se suele utilizar al cambiar la cinta, ya que a menudo la nueva se ha aflojado en el interior del cartucho antes de ser usada

Panel de control
Este panel posee dos mandos principales, que colocan a la impresora "en línea", es decir, la preparan para recibir información. También, si la impresora se detiene en mitad de una página, desplazan el papel hasta el inicio de la siguiente. Otro mando es el "avance de línea", que desplaza el papel hacia arriba una línea cada vez



Motor de la rueda margarita
El motor hace girar la rueda margarita y alinea los caracteres con el martillo que los golpea contra la cinta y el papel

Steve Cross

La impresora matricial
La impresora matricial utiliza una cuadrícula (matriz) de puntos para componer un carácter. La cabeza de impresión contiene un grupo de agujas que pican contra la cinta de la impresora dejando un punto sobre el papel. A medida que la cabeza de impresión se desliza a lo ancho del papel, unos impulsos eléctricos activan los alfileres en la secuencia correcta para componer el carácter

La impresora de chorro de tinta
Esta impresora, como su nombre sugiere, dispara un chorro de tinta a través de una boquilla que la dispersa en gotas diminutas. A cada gota se le proporciona una carga eléctrica y luego atraviesa unas placas deflectoras metálicas. La carga asegura que las gotas de tinta golpeen el papel según el diseño apropiado para componer el carácter

Steve Cross

Quizá hasta ahora no haya pensado seriamente en la posibilidad de usar una impresora. Después de todo, si está satisfecho utilizando su ordenador personal para jugar o para calcular los gastos de la familia, en realidad no necesita el texto impreso de lo que se visualiza en su monitor o en la pantalla de su televisor.

Pero cuando adquiera más experiencia en cuanto a la utilización de su ordenador personal, comprobará que trabajar seriamente careciendo de una impresora tiene unas limitaciones evidentes. Si le interesa escribir sus propios programas, querrá guardar copia de los listados de los mismos. Si utiliza su ordenador para llevar sus cuentas, necesitará un registro impreso de los cálculos.

La elección de la impresora adecuada a sus necesidades debe ser cuidadosa. Lo que haya de pagar por ella dependerá de la velocidad de la impresora en producir palabras y de la calidad de los resultados.

Elección de la impresora

Existen tres clases principales de impresoras para ordenadores personales: la matricial, la margarita y la térmica.

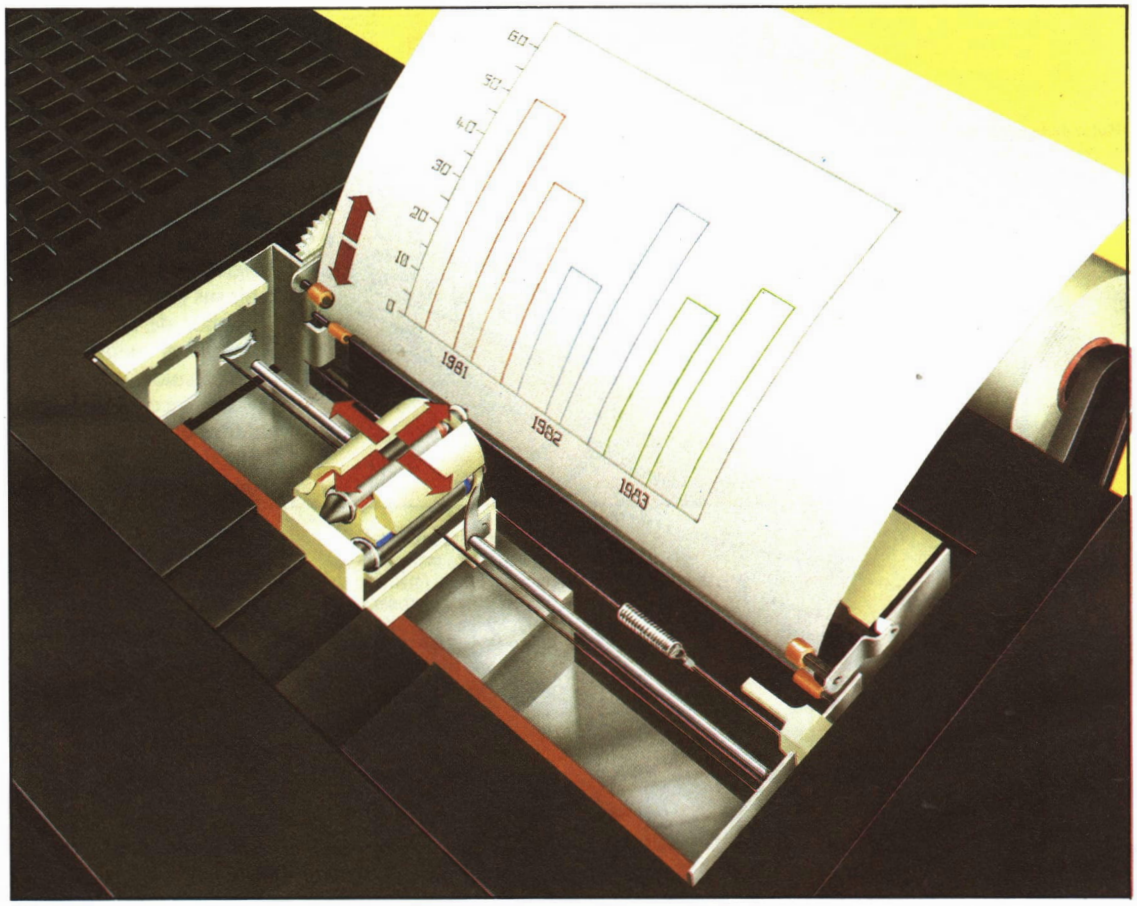
El método de impresión más difundido es el matricial. Éste consiste en una cabeza de impresión que contiene un grupo de agujas. Los caracteres se imprimen mediante las combinaciones de estas agujas al golpear contra la cinta. El método matricial tiene la ventaja de que trabaja muy rápido y de que las impresoras son relativamente económicas. Sin embargo, como las letras y los números se componen de una serie de puntos, la calidad de impresión suele ser muy pobre. Por otra parte, las impresoras de este tipo son bastante ruidosas.

Algunas impresoras matriciales han superado el problema de la baja calidad de impresión sobreimprimiendo los puntos dos o tres veces. En este caso, la cabeza de impresión se desplaza ligeramente para que los nuevos puntos encajen entre los puntos impresos previamente.

Las impresoras matriciales son aceptables si usted desea guardar borradores de lo que ha producido su ordenador. También pueden realizar gráficos y diagramas, ya que a la máquina le es posible imprimir patrones además de caracteres. Pero necesitará una impresora diferente si lo que desea es una impresión de gran calidad para, por ejemplo, enviar una carta al gerente de un banco.

Para obtener una calidad similar a la de las máquinas de escribir, ha de optar por una impresora margarita, así llamada porque utiliza una rueda con largos "pétalos" de aspecto parecido a los de dicha flor. En el extremo de cada pétalo hay una letra, un símbolo o un número. Para imprimirlos, la rueda gira para alinear cada pétalo con un pequeño "martillo" de metal que impulsa sobre la cinta y el rodillo el carácter situado en el extremo del pétalo. La rueda margarita puede ser plástica o metálica. También puede sustituirse por otra provista de un tipo de impresión diferente para obtener distintas clases de impresión, al igual que si se tratara de una máquina de escribir cuyos tipos están dispuestos en una bola.

El principal inconveniente de las impresoras margarita es que su funcionamiento es mucho más lento que el de las impresoras matriciales; además, suelen ser más caras. Tampoco sirven para diagramas y gráficos, ya que para producir las diversas formas de un gráfico se necesitarían varias ruedas.



Steve Cross

La impresora "esferográfica" Representa un reciente avance en cuanto a técnicas de impresión se refiere. La cabeza de impresión sostiene cuatro bolígrafos (plumas esferográficas) diseñados especialmente. Cuando se da la orden PRINT, el papel se mueve hacia arriba y hacia abajo para producir los trazos verticales del carácter, mientras los bolígrafos se mueven hacia los lados para crear los trazos horizontales. La ventaja de este sistema reside en que puede utilizarse para imprimir diagramas y gráficos en colores. Por otra parte, ofrece una mejor calidad de impresión que el procedimiento matricial, ya que sus caracteres se componen sencillamente a partir de los trazos del bolígrafo. En contrapartida, este sistema es comparativamente lento y los bolígrafos han de reemplazarse con regularidad si se imprimen de forma continuada textos muy prolongados

Las impresoras de chorro de tinta resultan aún más caras. Éstas disparan gotas de tinta para modelar el carácter que se ha de imprimir. La tinta es conducida a través de una boquilla que la dispersa en gotas diminutas. Estas gotas atraviesan luego un electrodo y reciben unas cargas eléctricas. Un par de placas metálicas desvían luego las gotas en distintas direcciones, para modelar la forma del carácter. ¡Las impresoras de chorro de tinta son tan rápidas que pueden imprimir alrededor de 20 metros de caracteres por segundo!

Un método alternativo, la impresora térmica, utiliza papel sensible al calor. La cabeza de impresión traspasa su calor al papel de modo tal que éste se vuelve negro en el área tocada, formando el carácter deseado. Las impresoras térmicas son muy silenciosas y bastante rápidas. Uno de los modelos más populares es el Apple Silent Type. El precio de las impresoras térmicas es bastante razonable; no obstante, con ellas necesitará utilizar un papel especial sensible al calor, más caro que el corriente, y, por otra parte, la calidad de impresión no es tan buena como la de las impresoras margarita.

La interface apropiada

Cuando decida comprarse una impresora, además de determinar exactamente para qué desea emplearla, en términos de calidad y velocidad de impresión, también habrá de asegurarse de que pueda utilizarla con su propio ordenador. La clavija conectora de la impresora debe ser compatible con el ordenador. El conector para enchufar la impresora está situado generalmente en la parte posterior del ordenador y se denomina "interface".

Los tres tipos de interface más corrientes son el Centronics, el IEEE488 y el RS232. El Centronics también se conoce como interface "paralela". Su ordenador ha de tener una abertura para al menos una de estas tres interfaces.

Sin embargo, la industria de la informática es notoriamente conocida por su incompatibilidad, y puede encontrarse con el hecho de que, aun utilizando la misma conexión, un ordenador y una impresora sean incompatibles. Esto se debe a que la interface debe trabajar a la misma velocidad tanto en el micro como en la impresora. La unidad para medir esta velocidad se denomina *baudio* y corresponde a la velocidad a la cual se transfieren a la impresora los bits de la memoria del ordenador. Los bits se pueden enviar a la impresora de dos maneras: transfiriéndolos uno detrás de otro a través de un único cable, como en la interface RS232 (procedimiento "en serie"), o bien transmitiéndolos simultáneamente a través de varios cables, como en las interfaces Centronics e IEEE488 (procedimiento "en paralelo").

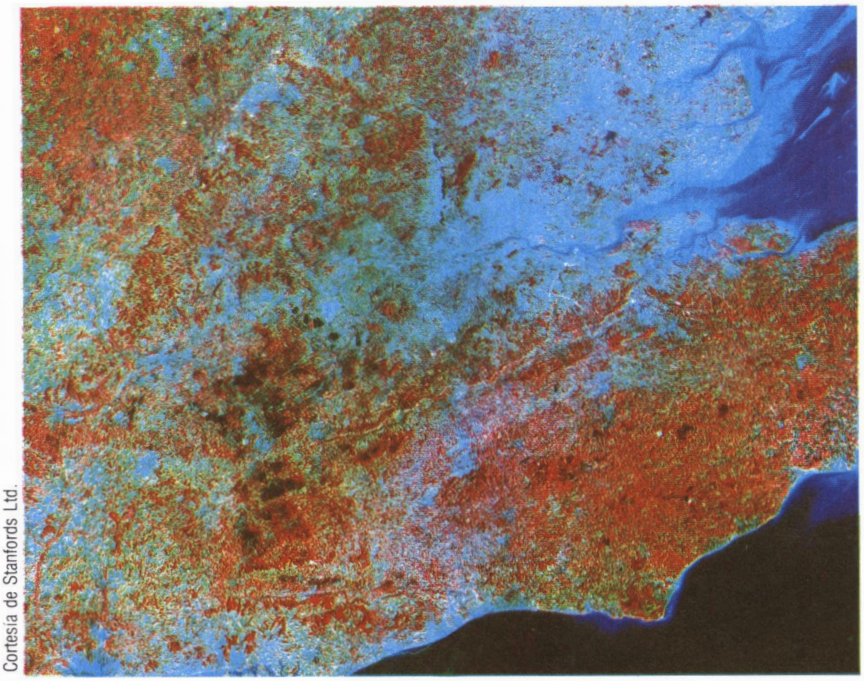
Existen dos formas principales de proveer de papel a las impresoras: hoja por hoja, como en una máquina de escribir, o por "arrastre", mediante dos ruedas dentadas que sujetan el papel por unos agujeros perforados situados a ambos márgenes de la hoja; se trata más o menos del procedimiento que utilizan las cámaras fotográficas para hacer correr la película de una exposición a otra. El papel de arrastre o para ruedas dentadas ofrece la ventaja de que se puede dejar que la impresora se alimente a sí misma. Sin embargo, este método no admite la clase de papel con membrete que ha de alimentarse a la impresora de una hoja cada vez.

Al comprar una impresora, debe decidir para qué desea utilizarla y luego adquirir la más adecuada para su ordenador.

Consultando al chip

Los "sistemas especializados" liberarán de la rutina a los expertos; se trata de ordenadores programados para analizar información compleja y responder preguntas relativas a la misma

La inteligencia artificial, es decir, la creación de ordenadores que piensen y tomen decisiones del modo en que lo hacen sus creadores humanos, es aún ciencia-ficción. La comprensión total del cerebro humano y de su funcionamiento es una tarea colosal, y a pesar de los avances que se están produciendo, las posibilidades de llegar a crear un ordenador inteligente, como el que aparece en la película *2001: una odisea del espacio*, seguirán siendo remotas durante muchos años.



Cortesía de Stanfords Ltd.

Panorama desde el espacio En julio de 1982 se lanzó el Landsat 4, que se desplaza en una órbita que cubre toda la superficie de la Tierra pasando por el mismo sitio una vez cada veinte días. Toda la información del sensor se envía en forma digital y, utilizando las técnicas informáticas, pueden resolverse objetos de sólo 40 metros de ancho y se pueden interpretar las características geográficas. En esta ilustración se ha procesado fotográficamente la información digital para mostrar aspectos de Londres y del sudeste de Inglaterra. Las aguas claras se ven en azul oscuro, y las aguas poco profundas, con sedimentos, en azul claro; las ciudades y los campos labrados se observan en azul grisáceo, los terrenos baldíos, en rojo amarronado; la cosecha de cereales tiene color verde y otras clases de vegetación se ven en rojo vivo

Pero si esta tarea pudiera limitarse, si el ordenador sólo hubiera de parecer "inteligente" en un campo muy restringido de la actividad humana, entonces sería mucho más sencillo reproducir, al menos, la apariencia de inteligencia.

Ésta es la teoría que sustenta a los sistemas especializados. El ideal perseguido concibe que un especialista en un campo determinado, por ejemplo, un geólogo o un experto cirujano, pudiera alimentar a un sistema de ordenador con sus conocimientos y con las reglas para aplicarlos. De esta manera, el programa de ordenador para manipular el conocimiento y las reglas quedaría a disposición de las personas no especializadas, quienes podrían digitar preguntas relativas a este campo del saber y recibir respuestas significativas.

Los sistemas especializados podrían ser de gran utilidad en diversos sentidos. Ya se ha desarrollado un programa para diagnosticar las causas del dolor de estómago interrogando a los pacientes acerca de sus síntomas. Otro programa se basa en el conocimiento humano de la geología para detectar los sitios donde existan mayores probabilidades de hallar molibdeno y otros minerales. Y un tercer programa deduce la probable estructura de moléculas orgánicas a partir de masas de información experimental no estructuradas. Normalmente todas estas tareas las asumirían científicos profesionales de altísima categoría cuya capacitación resulta sumamente onerosa. Gracias a los ordenadores, estas valiosas personas pueden ahora emprender trabajos más originales.

Pero los sistemas especializados tienen algo más que ofrecer aparte de la mera sustitución de los especialistas humanos. Una vez que el programa utiliza conocimiento especializado, el ordenador suele descubrir algunos hechos inesperados. Algunas veces la máquina detecta relaciones existentes entre un dato y otro que habían pasado inadvertidas para el hombre, y sugiere nuevas vías de investigación.

Existe, pues, la creencia generalizada de que los sistemas especializados suponen o, al menos, supondrán un importante avance en cuanto a las aplicaciones de los ordenadores. Si el programa a cargo del sistema funciona correctamente, el ordenador puede convertirse en un testigo especializado. Y muchos ordenadores pueden usar el mismo programa, traspasando el conocimiento especializado de una única persona a una gran cantidad de ordenadores igualmente especializados. El problema obvio con el que se enfrentan los investigadores es el de escribir un programa que *funcione* correctamente: un programa que sea tan "inteligente" como un especialista humano.

Creando el programa

El primer paso consiste en pensar de qué modo los expertos toman decisiones sobre la evidencia y las preguntas relativas a su especialidad. El pensamiento humano no es particularmente lógico, en especial si se lo compara con la forma en que trabajan los ordenadores, y depende en gran parte de la experiencia. Si a un especialista humano se le plantea una pregunta o un problema nuevos, los compara mentalmente con la gran cantidad de diversas situaciones con las que se ha enfrentado antes. Luego de comparar la nueva situación con aquellas que ya posee en su memoria, puede llegar a algunas conclusiones provisionales y emprender la acción adecuada.

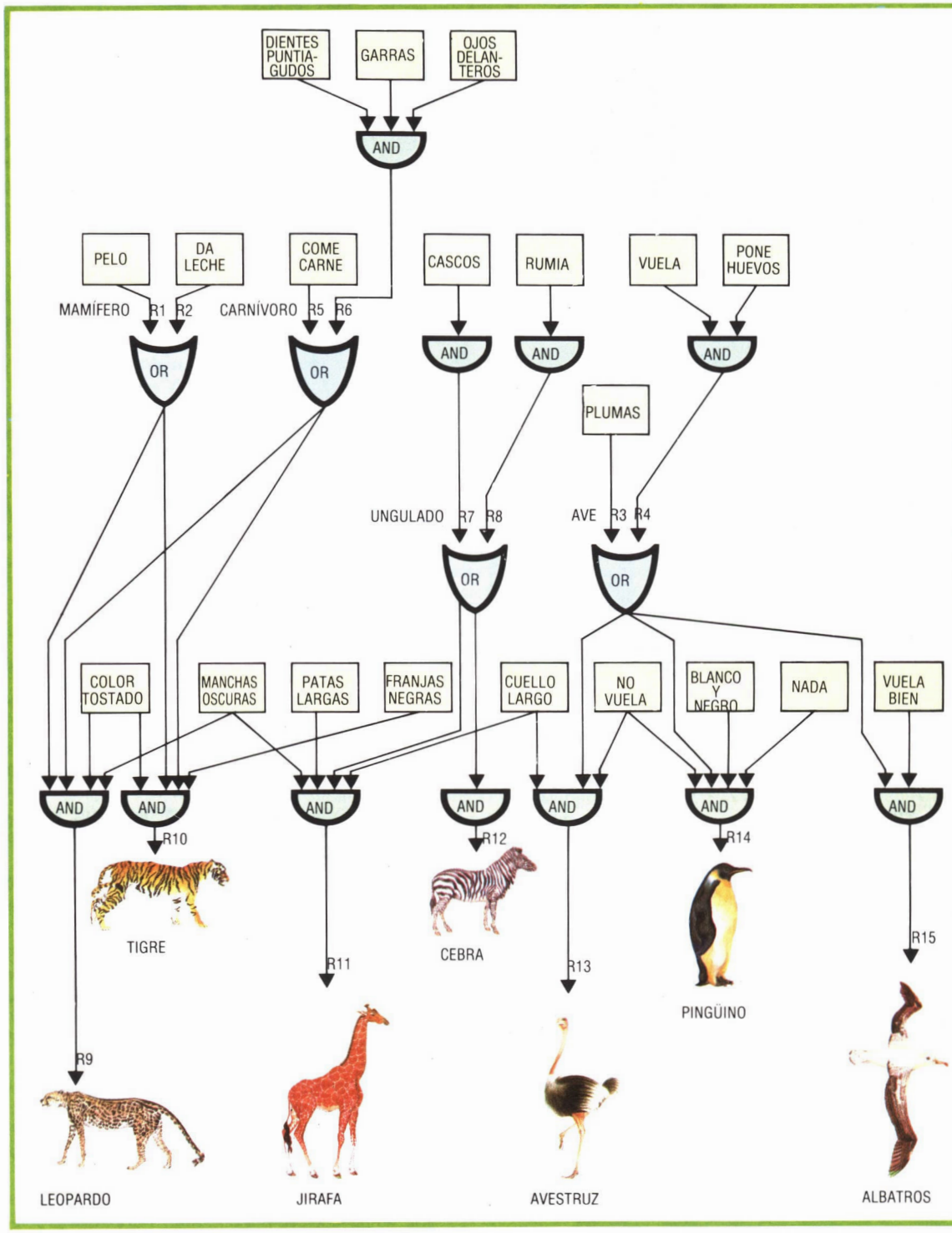
Pero representar el conocimiento profundamente detallado que posee, por ejemplo, un médico acerca de estas estructuras, implica almacenar y relacionar de formas muy complejas una enorme cantidad de reglas. Y son necesarias aún más modificaciones para que el ordenador imite el comportamiento humano. Los médicos nunca están absolutamente seguros de casi nada y al emitir una opinión sólo pueden decir que están "casi seguros" o "confían mucho" en ella. Basándose en sólo un par de síntomas, un médico sólo podría estar seguro de su diagnóstico en un 30 %.

De modo que en nuestro modelo de ordenador las conclusiones deben contar con unos valores de proba-

bilidad que se sitúan entre el 100 %, cuando sólo hay una conclusión posible, y el 1 %, cuando existen otras cien conclusiones igualmente posibles.

La mayoría de los sistemas especializados trabajan estableciendo un diálogo entre el interrogador y el ordenador. El interrogador ha de dar entrada a los detalles del problema que debe resolver el ordenador especializado; y en este sentido, el modo más sencillo de evitar problemas consiste en hacer que el ordenador interrogue al programador mediante una serie de preguntas objetivas (con varias respuestas posibles). Con esto se evita el riesgo de que el usuario dé entrada a palabras u oraciones que el ordenador no pueda comprender. Luego el ordenador compara la información de entrada con las reglas del conocimiento que tiene almacenado. La ruta que siga el ordenador en esta red de reglas depende de las respuestas que el usuario dé para sus preguntas, y a medida que el programa avanza a través de esta red, cada paso determina la siguiente pregunta del ordenador.

El sistema especializado más conocido en Gran Bretaña, el programa Mickie para diagnóstico médico, se ejecuta en un microordenador. Y están en proceso de fabricación otros productos comerciales que convertirán al ordenador personal en un especialista en diversos temas. Cierto es que aún no podemos hablar con nuestros ordenadores, pero sí formularles preguntas y recibir respuestas fiables.



Una red de animales

En este diagrama vemos cómo se organiza el "conocimiento" en un sistema especializado, de acuerdo a una serie de reglas sencillas. En este caso las reglas se numeran de R1 a R15 y tienen el fin de identificar a un animal a partir de la información que se tiene sobre él. Por ejemplo, supongamos que se da entrada a la información de que el animal posee dientes puntiagudos. El ordenador comienza por el recuadro superior izquierdo del diagrama y mira qué camino de la red puede tomar para llegar hasta uno de los animales incluidos en la parte inferior. La palabra AND (y) en el interior del círculo significa que las tres condiciones de los recuadros que se dirigen a ella han de ser verdaderas antes de que se pueda aplicar R6. El ordenador no puede seguir adelante hasta que no se le proporcione más información y requiere que se le respondan dos preguntas: "¿tiene garras ese animal?" y "¿tiene los ojos en la parte anterior de la cabeza (ojos delanteros)?" Si la respuesta es positiva en ambos casos, R6 permite que el ordenador continúe hacia abajo. Pero R6 conduce a una OR (o), lo que significa que sólo podrá seguir adelante si R6 o R5 son verdaderas. En este caso R6 es verdadera, de modo que el ordenador continúa. Aquí hay dos posibilidades AND, cada una de las cuales lleva a conclusiones diferentes. Para aplicar ya sea R9 o R10, y dar la respuesta LEOPARDO o TIGRE, el ordenador ha de formular más preguntas. En este caso ha de preguntar si el animal posee franjas negras o manchas oscuras. Con esta información ya podrá dar una respuesta. Éste es el diagrama clásico de la forma en que funciona el juego por ordenador "Animales"

Julie-Ann Chambers/Mark Watkinson

Cuando menos = más

Los ordenadores prefieren la sencillez, y por ello emplean un inteligente truco para efectuar una resta mediante una suma

Cortesía de the Science Photo Library

El Voyager 2
El espectacular Voyager se convirtió en el primer explorador espacial que fue más allá de nuestro sistema solar. Viajó a través del espacio, tomando fotografías y acumulando información, mientras el ordenador de a bordo convertía los datos en dígitos binarios. La información se enviaba a la Tierra a la asombrosa velocidad de 116 000 bits por segundo. En la Tierra era procesada por los ordenadores de la NASA, en Houston (Texas)

En la primera parte de esta serie descubrimos que se pueden emplear dígitos binarios para representar cualquier número decimal. Los números binarios tienen la desventaja de que son más largos que sus equivalentes decimales, pero son convenientes para el ordenador porque los ceros y los unos pueden representarse mediante voltajes negativos y voltajes positivos. También hemos visto que es muy sencillo sumar números binarios entre sí.

En el papel, los números binarios pueden restarse con igual sencillez que los números decimales, siguiendo las mismas reglas que se aplican a la resta en el sistema decimal. Sin embargo, hace tiempo que los diseñadores de ordenadores comprendieron que agregando circuitos (circuitos electrónicos para sumar) podían tanto sumar como restar sin necesidad de circuitos especiales de resta. Veamos cómo se realiza.

El complemento a dos

Un procedimiento para representar los números negativos en los ordenadores se denomina "complemento a dos". Con él, el proceso de la resta aparece como un aspecto más de la suma. Consideremos el siguiente problema aritmético:

$$16 - 12 = 4$$

o

$$16 + (-12) = 4$$

Aquí 12 se resta a 16, pero el proceso de sustracción puede considerarse, igualmente, como una adición: la suma de 16 y 12 negativo. En ambos casos la respuesta es la misma y la única diferencia es la utilización de signos aritméticos y de los paréntesis. Esta ligera modificación puede utilizarla el ordenador tanto para representar números negativos como para simplificar el problema de la resta.

Por razones de claridad, supongamos que la capacidad de nuestro ordenador sólo alcanza para manipular 5 dígitos. Por supuesto, los ordenadores reales pueden registrar números con miles de dígitos. Nuestro ordenador de 5 dígitos adopta un método de trabajo: el primer dígito por el lado izquierdo se considera separadamente de los otros 4. Si el dígito inicial es 1, representa 16 negativo; y si es 0, representa, por supuesto, un cero. Los cuatro dígitos restantes son positivos y responden a las convenciones binarias que hemos visto en el primer capítulo de este curso:

[ ]	[ ]	[ ]	[ ]	[ ]
−16 o 0	8	4	2	1 o 0

De manera que, por ejemplo, el número binario 01000 es el decimal 8 y 10000 es el decimal −16. Pero, ¿qué sucede con 10100? Éste incluye −16 y +4, dando −12.

¿Cuántos números pueden representarse con sólo 5 dígitos utilizando esta convención? El mayor número positivo es 01111 o decimal 15, y el mayor número negativo es 10000 o −16. Experimentando un poco verá que se pueden representar todos los números comprendidos entre −16 y +15.

Binario	Decimal
10000	−16
10001	−15
10010	−14
10011	−13
10100	−12
etc.	
11111	−1
00000	0
00001	1
00010	2
etc.	
01110	14
01111	15

Si aumentáramos el número de dígitos que pudiera manipular nuestro ordenador, ampliaríamos, por supuesto, la gama de números.

Ya en los primeros tiempos del desarrollo de la aritmética binaria para ordenadores, se descubrió un truco muy sencillo para hallar el complemento a dos, o forma negativa, de un número. Para llegar a este truco hay que seguir dos pasos.

Primero, invertir cada dígito. De modo que cada vez que encuentre un 1 ponga un 0, y cada vez que haya un 0 lo cambie por un 1. En segundo lugar, sumarle 1 al número invertido.

Siga el método tal como se expone en el ejemplo siguiente. Estamos utilizando +12, siendo 01100 su equivalente binario. (El 0 inicial sobre el lado izquierdo no es estrictamente necesario, puesto que 01100 es lo mismo que 1100. Pero, dado que nuestro ordenador posee 5 dígitos, debemos utilizarlos todos.)

	01100 (= +12)
Primer paso:	10011
Segundo paso:	00001 (+1)
	10100 (= −12)

Ahora veamos cómo aborda nuestro ordenador el problema de la resta, por ejemplo, 12 menos 4, utilizando el complemento a dos.

+12 es	01100
−4 es	11100 (usando el complemento a dos)
12 + (−4)	101000

Observe que ahora tenemos 6 dígitos. Puesto que la capacidad de nuestro ordenador sólo alcanza para registrar 5 dígitos, el primer dígito a la izquierda se denomina dígito de capacidad excedida y se ignora, ¡dejando 01000 u 8 decimal, que es la respuesta correcta! Veamos ahora un ejemplo algo más complicado: 4 menos 12.

+4 es	00100
−12 es	10100
4+(−12)	11000

Como ejemplo final, intentemos trabajar con dos números negativos al mismo tiempo: −3 −4 = = −3 + (−4) = −7

3 es	00011
por tanto, −3 es	11101 (usando el complemento a dos)
y −4 es	11100
	111001

Como podrá observar, nuevamente obtenemos un número de 6 dígitos. Una vez descartada la capacidad excedida, tenemos el número binario 11001 o −7 en decimal.

Estas restas utilizan sólo la suma y el truco del complemento a dos (que en sí mismo consiste sólo en la inversión de los dígitos y la suma).

La ventaja para el ordenador es que los dígitos binarios pueden invertirse fácilmente empleando una puerta NOT.

Una puerta NOT tiene una entrada y una salida. Es una puerta muy "díscola" porque cualquiera sea el valor que usted alimente, la salida será lo contrario. De manera que si la entrada es 0 la salida es 1, y si la entrada es 1 la salida es 0. Esta característica de "inversión" es exactamente lo que se necesita para el primer paso (la inversión) del truco del complemento a dos.

En un próximo capítulo de nuestra obra podremos ver cómo un ordenador tiene capacidad para sumar fácilmente utilizando una combinación de puertas lógicas.

Puntos y rayas
El código Morse es uno de los primeros ejemplos de la codificación binaria en la electrónica. En 1837 se instaló en Londres el primer telégrafo, con dos millas de cable que unían las estaciones de ferrocarril de Euston y Camden Town. Posteriormente, en ese mismo año, Samuel Morse mostró en los Estados Unidos su celebrado código para transmitir mensajes. Cada letra era una combinación de dos señales: puntos y rayas

Cuestiones de rutina

Programas dentro de otros: presentamos un nuevo aspecto del Basic, que hará que los programas sean claros y manejables

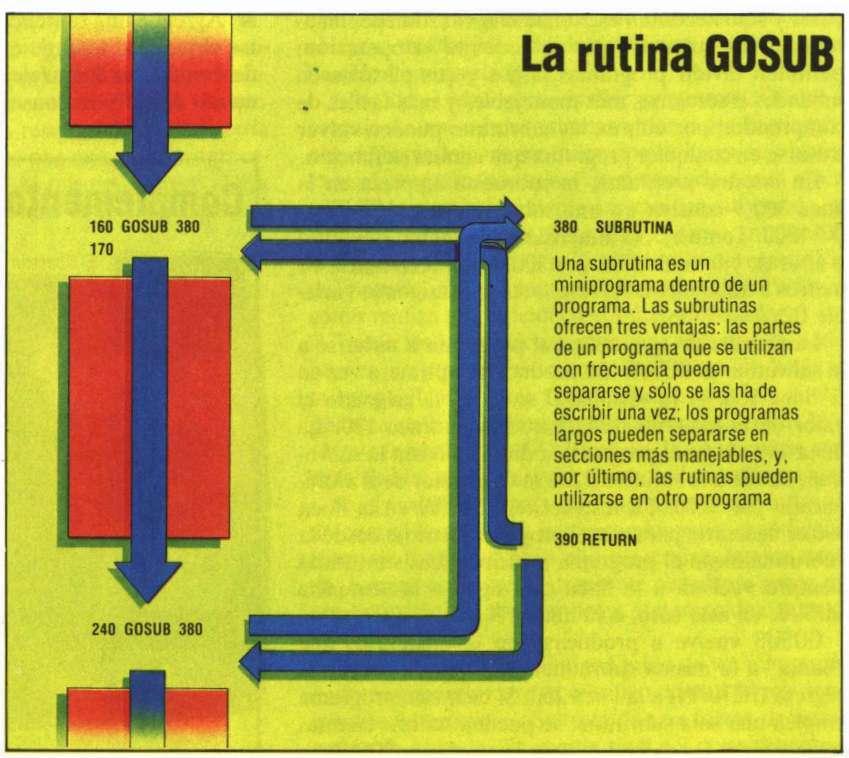
En anteriores entregas de nuestro curso de programación BASIC hemos digitado programas, los hemos ejecutado, hemos introducido modificaciones y luego borrado la memoria (mediante la orden NEW) cuando deseábamos dar entrada a nuevos programas. Cuando hemos necesitado volver a ejecutar otra vez el antiguo programa, ha sido necesario volver a digitarlo.

Para ahorrarnos este trabajo repetitivo, todas las versiones de BASIC se suministran con una orden que permite almacenar cualquier programa en cinta de cassette. El siguiente programa puede guardarse en cinta utilizando la orden SAVE seguida del nombre de archivo. El programa servirá para calcular el número de azulejos que se necesitan para revestir las paredes de una habitación.

```
10 REM ESTE PROGRAMA CALCULA EL NUMERO DE
   AZULEJOS
20 REM QUE SE NECESITAN PARA AZULEJAR UNA
   HABITACION
30 PRINT "DE ENTRADA A LA MEDIDA DE LOS LADOS
   DEL AZULEJO EN MM"
40 INPUT A1
50 REM LA LINEA 60 HALLA LA SUPERFICIE DEL
   AZULEJO
60 LET A2 = A1 * A1
70 PRINT "DE ENTRADA AL NUMERO DE PAREDES"
80 REM W ESTABLECE EL LIMITE DEL BUCLE
90 INPUT W
100 FOR X = 1 TO W
110 PRINT "LONGITUD DE LA PARED N.º"; X;
   "EN METROS"
120 REM D ES LA DIMENSION DE LA PARED
130 INPUT D
140 REM SE CONVIERTE EN MM
150 REM EN LA SUBRUTINA
160 GOSUB 380
170 REM LINEA 190 ESTABLECE L EN
180 REM LONGITUD DE LA PARED EN MM
190 LET L = D2
200 PRINT "ALTURA DE LA PARED N.º"; X;
   "EN METROS"
210 REM LINEAS 230 HASTA 250 ESTABLECEN H
220 REM EN LA ALTURA DE LA PARED EN MM
230 INPUT D
240 GOSUB 380
250 LET H = D2
260 REM LINEA 270 ESTABLECE QUE A3 ES LA
   SUPERFICIE DE LA PARED
270 LET A3 = L * H
280 REM S (SUBTOTAL) ES LA SUPERFICIE
   DE LA PARED DIVIDIDA
290 REM POR LA SUPERFICIE DEL AZULEJO
300 LET S = A3/A2
310 REM T (TOTAL) TIENE EL NUEVO SUBTOTAL
320 REM AGREGADO CADA VEZ QUE REALIZA
   EL BUCLE
330 LET T = T + S
340 NEXT X
350 REM PRINT EL TOTAL
360 PRINT T
370 END
380 LET D2 = D * 1000
390 RETURN
```

Habiendo digitado el programa, todo lo que ha de hacer para guardarlo en la cinta de cassette es utilizar la orden SAVE. Antes, por supuesto, debe preparar el ordenador de acuerdo a las instrucciones del manual. La orden SAVE es muy sencilla de utilizar. Basta con que digite SAVE seguido de un nombre de archivo entre comillas dobles. Un nombre de archivo es la denominación que se le da a un archivo y, en la terminología de informática, equivale a un programa o una serie de datos que pueden almacenarse o recuperarse cuando se desee. Lo mejor es utilizar un nombre de archivo que le recuerde la función del programa. Dado que nuestro programa calcula el número de azulejos que se necesitan para azulejar una habitación, podríamos denominarlo "AZULEJOS". Una vez preparada la grabadora de cassette, introduzca en ella una cinta virgen para retener el programa.

Las grabadoras de cassette con conector para control remoto pueden controlarse, por lo general, directamente mediante el ordenador. De lo contrario, coloque la grabadora en la modalidad *record* y luego dispóngala en la modalidad *pause*. Digite la orden SAVE, incluyendo el nombre de archivo. Ponga en funcionamiento la grabadora soltando el mando *pause* y luego pulse RETURN.



Tony Lodge

Para comprobar si el programa se ha grabado correctamente, borre la memoria del ordenador digitando NEW <CR>. Rebobine la cassette, coloque la grabadora en la modalidad *play* y luego vuelva a cargar el programa en el ordenador utilizando la orden LOAD. LOAD debe ir seguida del nombre de archivo del archivo deseado. Digite LOAD "AZULEJOS" y enseguida pulse RETURN.

Luego de que el programa haya sido cargado en el ordenador, un mensaje en la pantalla tal como READY u OK indicará que se ha completado la carga. Liste el programa (LIST) y verifique que se trata del mismo que ha digitado.

GOSUB

GOSUB es una sentencia que desvía el flujo de un programa a una subrutina. Una subrutina es como un miniprograma separado o un programa dentro de otro programa. En el programa que hemos empleado para ilustrar el tema, la subrutina es muy sencilla. Se incluye para mostrar el principio, puesto que podrían haberse elaborado fácilmente otras maneras de producir los mismos resultados sin recurrir a la utilización de una subrutina.

Nuestro programa calcula el número de piezas que se necesitan para azulejar un cuarto, averiguando la superficie de los azulejos empleados. Luego solicita que se dé entrada a la longitud y la altura de cada una de las paredes. Calcula la superficie de la pared después de haber convertido la longitud y la altura de metros a milímetros. El número de azulejos necesarios se calcula dividiendo la superficie de cada una de las paredes por la superficie de un azulejo y luego sumando los resultados. La conversión de la longitud y la altura de las paredes a milímetros se realiza en la subrutina, que simplemente multiplica la longitud o la altura (en metros) por 1 000 para hallar el equivalente en milímetros.

Las subrutinas ofrecen tres ventajas: las partes de los programas utilizadas con frecuencia se pueden separar y sólo necesitan escribirse una vez, independientemente de cuántas veces se requiera la operación; permiten dividir programas largos y complicados en unidades o secciones más manejables y más fáciles de comprender; por último, las subrutinas pueden volver a usarse en cualquier programa que admita su función.

En nuestro programa, la subrutina empieza en la línea 380 y consiste en una sola sentencia: LET D2 = D * 1000. Toma a D, la dimensión de la pared (longitud o altura), y la multiplica por 1000 para convertirla de metros a milímetros. Al resultado se le asigna la variable D2.

La instrucción que obliga al programa a dirigirse a la subrutina es GOSUB. Se produce por primera vez en la línea 160. A la variable D se le había asignado el valor de la longitud de la pared en la línea 130. La línea 160 obliga al programa a dirigirse hasta la subrutina, donde a la variable D2 se le da el valor de D multiplicado por 1000. La instrucción RETURN en la línea 390 es necesaria para que el programa retorne desde la subrutina hasta el programa principal. Las subrutinas siempre vuelven a la línea que sigue a la sentencia GOSUB; en este caso, a la línea 170.

GOSUB vuelve a producirse en la línea 240, que "llama" a la misma subrutina. Esta vez, la subrutina regresa (RETURN) a la línea 250. Si bien este programa emplea una sola subrutina, se pueden utilizar cuantas se necesiten. En todos los casos la sentencia GOSUB ha de incluir el número de línea de la subrutina apropiada. Observe que la sentencia END se produce en la línea 370, antes que la subrutina. END indica el final del programa principal y también sirve para detener el programa, evitando que siga ejecutando las subrutinas después de haber finalizado.

A pesar de que este programa es algo más largo que los anteriores, no por ello es más complicado. Pruebe con este programa, sígalo entero, línea por línea, y vea lo que sucede en cada etapa. Además de la orden GOSUB y de las subrutinas, este programa sólo introduce otro concepto: nombres de variables más largos.

Le sería de gran ayuda dibujar recuadros con los nombres de las variables en su interior y escribir, además, los valores para cada etapa.

Línea 300: LET S = A3/A2 algunas veces dará un número con fracción decimal. Intente ejecutar el programa dando entrada a 110 mm como medida de los azulejos, y utilizando una sola pared de 2,3 y 1,8 metros de longitud y altura respectivamente. Debe obtener como respuesta 342,149 azulejos. Como los azulejos únicamente se venden por unidades enteras, esta respuesta no es del todo apropiada. En una próxima ocasión estudiaremos una de las maneras de conseguir una respuesta adecuada en números enteros.

Ejercicios

■ Vea qué sucede si da entrada a 0 mm como medida del azulejo. Al final de la ejecución debería obtener un mensaje de error. ¿Y por qué? ¿Por qué no obtiene un mensaje de error similar cuando da una entrada de 0 metros para la longitud de una de las paredes? Una pista: no es igual multiplicar por cero que dividir por cero; ¡pruébelo en su calculadora!

■ El programa sólo funciona cuando se trata de azulejos cuadrados. Vea si puede modificar desde la línea 30 hasta la línea 60 para hallar la superficie de azulejos rectangulares (del mismo modo que, más adelante en el programa, hemos hallado la superficie de paredes rectangulares).

■ Agregue una sentencia en la línea 355 para aumentar el número total de azulejos en un 5 %, con el fin de compensar los azulejos que se estropeen. El aumento del 5 % se conseguirá multiplicando un número por 105/100.

Complementos al BASIC

END

El Spectrum no dispone de esta orden, lo que significa que se necesita otro procedimiento para pasar por alto las líneas 380 y 390. Este consiste en modificar la línea 370 para que quede: 370 GOTO 400, y agregando 400 PRINT "END"

LET

El Spectrum requiere que se definan todas las variables antes de que pueda realizar una operación aritmética. De modo que para que la línea 330 funcione correctamente, debe agregarse una nueva línea:
5 LET T = 0

GOSUB

En el Spectrum aparece como dos palabras separadas, si bien sólo se necesita una tecla

Descifrando el código

Digite su lenguaje de ordenador en el teclado y un programa que funciona dentro del micro lo convertirá rápidamente en su propio código de lenguaje máquina

A pesar de que parece que todos los microordenadores desempeñan funciones similares, cada modelo es exclusivo. Algunos se venden con programas ya incorporados, mientras que otros requieren que estos programas se les "lean" desde un disco o una cinta de cassette exteriores.

Algunas máquinas poseen un único programa global que permite tanto la entrada de programas como la utilización de órdenes directas como SAVE o LOAD. Para realizar estas dos funciones, otros modelos necesitan programas separados.

No obstante, la mayoría de los microordenadores más conocidos funcionan de acuerdo a principios similares. El movimiento de la información hacia y desde dispositivos de almacenamiento externos (discos y cintas) a la pantalla se controla, en ambos casos, mediante el teclado. Asimismo, todas las máquinas pueden comunicarse con otros dispositivos externos, como impresoras, trazadores de gráficos e instrumentos científicos. Y la mayoría de los micros admiten que sus usuarios escriban programas en lenguajes similares al corriente, como el BASIC.

Cuando digita en el teclado de un ordenador un programa en BASIC, otro programa denominado *sistema operativo* pasa lo que ha digitado tanto a la pantalla como a un programa interpretador de BASIC. Esto significa que en el interior del ordenador se están llevando tres programas simultáneamente: el sistema operativo, el interpretador de BASIC y su propio programa.

Cuando usted ejecuta su programa, estos tres programas deben actuar al mismo tiempo. Cada instrucción en BASIC de su programa es traducida por el interpretador, y las instrucciones resultantes en código de lenguaje máquina pasan, una por una, al microprocesador para que las ejecute. Al mismo tiempo, el sistema operativo está explorando el teclado para detectar la entrada de datos y poder visualizarlos en pantalla.

Si una de las instrucciones de su programa solicita que algo se imprima o se grabe en disco, por ejemplo, el interpretador le requeriría al sistema operativo que realice esta tarea.

La ilusión de que en el mismo momento se están produciendo diversas acciones se crea por la increíble velocidad con que trabaja el microprocesador. Éste procesa las instrucciones del sistema operativo y del interpretador con tanta rapidez que ambas pueden ejecutarse simultáneamente.

Algunas máquinas trabajan a velocidad aún mayor, permitiendo la llegada de datos para "interrumpir" el procesamiento normal. De esta forma, no es necesario que el sistema operativo verifique las diversas fuentes de entradas de datos externos, como el teclado o las unidades de disco.

Se denomina *editor* a un tipo de procesador de textos menos sofisticado. La calidad de los editores suele variar considerablemente; no obstante, es probable que el editor incorporado del interpretador de BASIC sea de similar calidad.

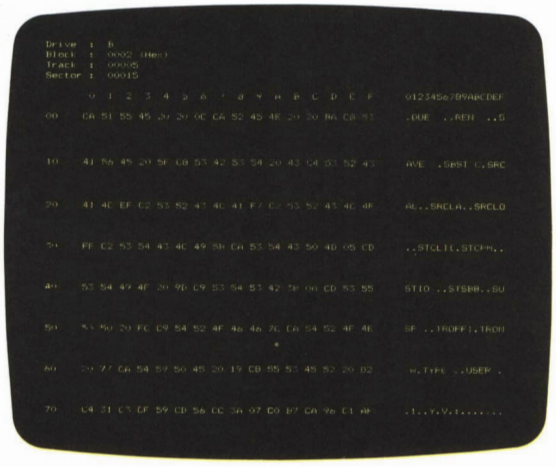
Para ejecutar sus programas en BASIC, en lugar de emplear un interpretador puede utilizar un compilador. Mientras que el interpretador ha de traducir una instrucción cada vez que se le proporcione, el compilador traduce todo su programa, completo, al código de lenguaje máquina del ordenador una sola vez y para siempre. Los programas que han sido "compilados" se ejecutan mucho más rápidamente que el software que ha sido "interpretado".

El BASIC es ideal para escribir programas. Ofrece la ventaja de que es muy similar al lenguaje corriente y muy apropiado para ser utilizado por los principiantes. Pero el programador más experimentado podrá ejecutar sus programas con mayor rapidez utilizando un lenguaje ensamblador. Éste no se parece en absoluto al lenguaje común y, por lo tanto, el programador ha de poseer un conocimiento bastante detallado de cómo realiza sus funciones el microprocesador.

Todas las instrucciones que se le dan al ordenador poseen un equivalente directo en código de lenguaje máquina. Un lenguaje ensamblador está constituido por una serie de instrucciones abreviadas, como MVI *(Move Immediate)* o JZ *(Jump on Zero)*. Éstas se usan para ayudar al programador a recordar sus funciones.

Si el usuario domina un lenguaje ensamblador, lo único que le quedaría por aprender sería el código de lenguaje máquina, pero ello no tendría mucho sentido, a menos que verdaderamente necesitara ganar mínimas fracciones de segundo en el tiempo de ejecución de un programa.

Los códigos de lenguaje máquina de los microordenadores generalmente se escriben de una forma denominada *hexadecimal*. Se trata de una forma de numeración de base 16. Se cuenta de 0 a 9 normalmente, y



El sistema operativo de disco
Cuando los programas se almacenan en un disco flexible, la información se distribuye al azar alrededor de la superficie del disco. El sistema operativo de disco es un programa que automáticamente lleva el registro de la localización de cada byte de información. La ilustración muestra la información almacenada en una sección pequeña de un disco. En el bloque izquierdo, ésta se representa en forma hexadecimal y, en el bloque de la derecha, figura el carácter equivalente. Los códigos que no corresponden a caracteres que han de imprimirse aparecen en forma de puntos

luego se continúa utilizando las letras del alfabeto desde la A a la F para los números del 10 al 15. Cada dirección de memoria comprende ocho dígitos binarios (bits) y éstos se pueden representar mediante un par de dígitos hexadecimales.

Por ejemplo, el número binario 01011101 se ha de dividir primero en dos mitades: 0101 y 1101. Éstas se traducirían en los números decimales 5 y 13, que en el sistema hexadecimal equivalen a 5 y D. Así, al programar en código de lenguaje máquina, 01011101 es 5D. Éste constituye el procedimiento más lento de desarrollar programas, pero probablemente sea la manera de obtener los tiempos de ejecución más rápidos. Lo que sigue corresponde a extractos tomados de programas típicos:

```
BASIC
100 INPUT "Entre las horas trabajadas"; HORAS
200 PAGA = HORAS * TASA
```

La primera línea visualiza en pantalla un mensaje que invita al usuario a dar entrada a las horas trabajadas. Acepta la entrada y luego, en la segunda línea, la multiplica por la tasa de la paga (introducida previamente) para dar la cifra global de la paga.

```
Lenguaje ensamblador
MVL C, 01
CALL 05
```

La primera de las instrucciones anteriores desplaza el valor "1" a una parte de la memoria del microprocesador denominada "registro C". La segunda instrucción entrega el control al sistema operativo. Entonces el sistema operativo vuelve a entregar el control a su programa. Lo que sigue es la traducción directa de las instrucciones en lenguaje ensamblador que citamos arriba:

```
Código de lenguaje máquina
0E01
CD0500
```

Una traducción en código de lenguaje máquina de las dos sentencias en BASIC comprendería muchas órdenes. Evidentemente, es preferible que este trabajo lo realice un compilador o un interpretador, en lugar de escribirlo en lenguaje ensamblador o en código de lenguaje máquina.

Algunas órdenes del sistema operativo son tan enigmáticas como los programas en lenguaje ensamblador. He aquí algunos ejemplos tomados del CP/M (*Control Program for Microcomputers;* programa de control para microordenadores):

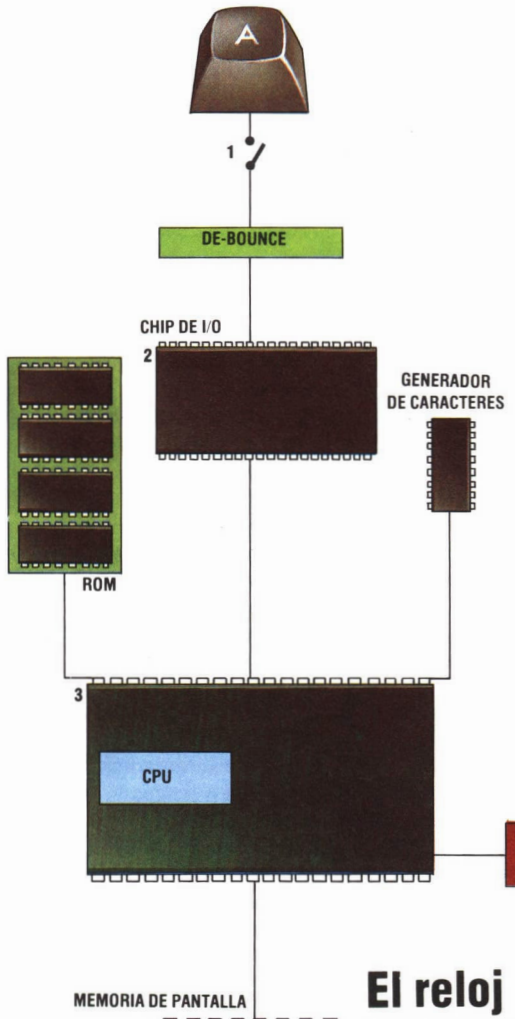
```
DIR * . BAS
```

Esto significa "liste todos los ficheros de la unidad de disco cuyo sufijo sea BAS". DIR es la abreviatura de "directorio".

A pesar de esta forma más bien extraña de elaborar las órdenes, el CP/M está instalado en gran número de máquinas. Ofrece enormes ventajas para los escritores de software profesionales. Los programas se pueden transportar fácilmente de una máquina a otra, con la única condición de que estén escritos de manera tal que entreguen al CP/M el control de la manipulación de los discos, el teclado, la impresora y la pantalla.

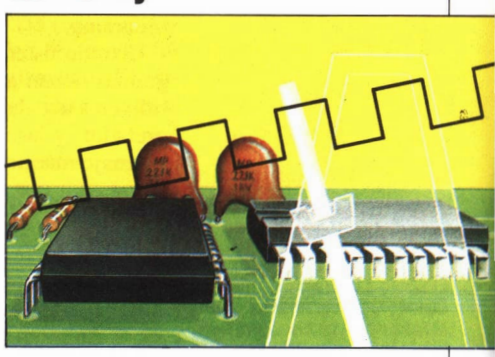
Los fabricantes que presenten una máquina nueva pueden beneficiarse del software de base ya existente, y los interesados pueden adquirir su ordenador en la confianza de que con toda seguridad podrán satisfacer sus necesidades en cuanto a software.

De la tecla a la pantalla



(1) Normalmente, al pulsar una tecla del ordenador aparece de inmediato una letra en el monitor. El proceso, sin embargo, es más complicado de lo que parece. Al pulsar una tecla se envía una señal a través del chip de I/O **(2)** hacia la CPU. La CPU **(3)** ejecuta un pequeño programa en la ROM, que detecta cuál es la tecla que se ha pulsado y coloca un código correspondiente en la memoria que normalmente se utiliza como almacenamiento de la imagen en pantalla, la RAM de pantalla **(4)**.
Mientras tanto, el sistema de circuitos de video **(5)** explora constantemente la RAM de pantalla y, tomando los patrones de bit relacionados con cada letra de una ROM "generadora de caracteres", envía una señal al televisor o al monitor, que entonces dibuja la letra en la pantalla **(6)** en el lugar correcto y, en su caso, en uno de los colores disponibles. Todo este proceso se lleva a cabo en apenas unas millonésimas de segundo

El reloj



La sincronización interna del ordenador es verdaderamente decisiva. Por ejemplo, si la CPU lee un byte de la memoria debe establecer las líneas de dirección de la localización del byte. La CPU ha de estar segura de que el chip de la memoria ha tenido tiempo suficiente para colocar el byte de información en el bus de datos. Sólo entonces aceptará y "leerá" el byte en la memoria interna de la CPU. En los microordenadores personales, estos relojes hacen tictac cuatro millones de veces por segundo

Términos clave

Puertas y sumadores

Los números binarios, compuestos de ceros y unos, se pueden sumar entre sí mediante la sencilla lógica del AND, OR y NOT

En un artículo anterior hemos visto cómo los relativamente sencillos circuitos de transistor se pueden utilizar para tomar decisiones lógicas como AND, OR y NOT. Lo sorprendente es que estas mismas "puertas lógicas" también constituyen los bloques de nes aritméticas en el interior del ordenador. Lógicamente, las entradas de las puertas son, bien de voltaje cero, para representar "falso", o bien de voltaje positivo, para representar "verdadero". La ausencia de voltaje se suele simbolizar con un cero (0) y el voltaje positivo con un uno (1). Cuando las puertas lógicas se utilizan para la aritmética, se emplean los mismos ceros y unos, pero entonces representan literalmente los unos y los ceros que se suman.

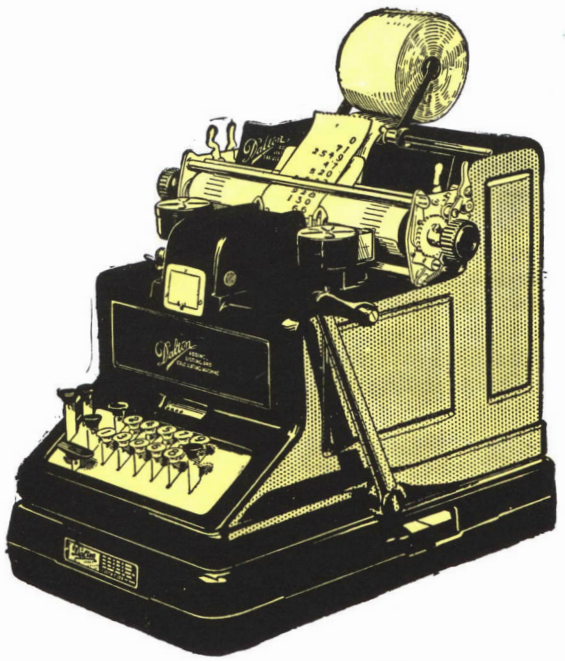
Si deseamos sumar dos dígitos binarios, sólo habrá dos entradas en el circuito de suma, y únicamente podrá haber cuatro combinaciones de entrada: 0 + 0, 0 + 1, 1 + 0 y 1 + 1. Estudiando la aritmética binaria hemos aprendido que 0 más 0 es igual a 0 (como en aritmética decimal). Sabemos también que 0 más 1 (o 1 más 0) es igual a 1 (igual que en aritmética decimal). La diferencia, respecto a la aritmética que hemos aprendido en la escuela, es que en binario 1 más 1 es igual a 0 y llevamos 1. La demostración aritmética de estas cuatro sumas sería la siguiente:

X		Y		Z
0	+	0	=	0
0	+	1	=	1
1	+	0	=	1
1	+	1	=	10

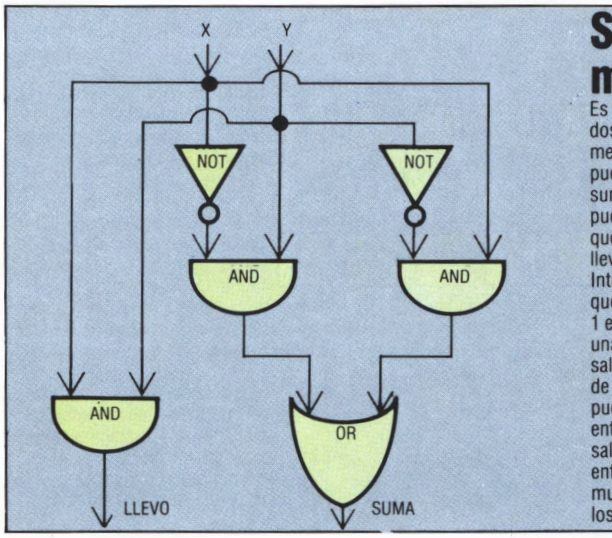
Si hubiéramos de utilizar una puerta OR para efectuar la suma, obtendríamos una salida falsa (0) si ambas entradas fueran falsas (0), y una salida verdadera (1) si alguna de las entradas fuera verdadera (0 + 1 o 1 + 0).

De momento, para sumar dos dígitos binarios parecería perfectamente adecuado utilizar una puerta OR simple. Pero veamos. Si ambas entradas fueran verdaderas, la salida de una puerta OR simple también sería verdadera, pero en aritmética binaria esa respuesta estaría equivocada. La respuesta correcta sería 0 y llevo 1. Una puerta OR simple daría una respuesta correcta para tres de las cuatro combinaciones de entrada posibles, pero tres de cuatro no es un nivel satisfactorio.

Lo que se necesita es un circuito que dé una respuesta de 0 si ambas entradas son 0, y una respuesta de 1 si una de las entradas es 0 y la otra es 1, y una respuesta de 0 si ambas entradas son 1 (como en la tabla de verdad anterior). Esto no es tan difícil como parece. Si tenemos dos puertas AND, con las dos entradas yendo a ambas puertas, pero invirtiendo una de las entradas a través de una puerta NOT por una de las puertas AND, y si la otra entrada se invierte a través de otra puerta NOT que va hacia la otra puerta AND (véase la ilustración), obtendríamos una situación en la cual un 0 en ambas entradas dará una salida falsa en las dos puertas AND, y, del mismo modo, un 1 en ambas entradas dará una salida falsa para ambas puertas. Por otra parte, un 0 en una entrada y un 1 en la otra darán dos entradas verdaderas para una de las puertas AND. Una de estas puertas producirá, por tanto, una salida verdadera. Si las dos puertas AND tuviesen conectadas sus salidas con una puerta OR, la salida de la puerta OR sólo sería verdadera si una, y sólo una, de las dos entradas fuera verdadera.

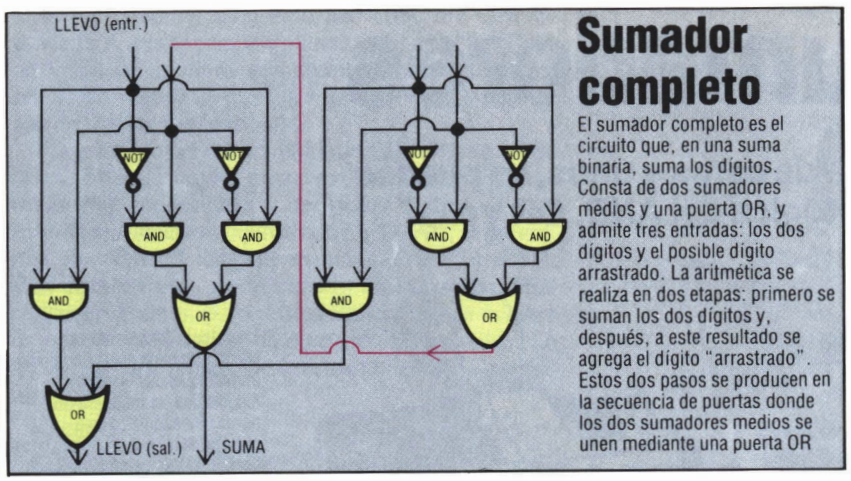


La calculadora de mesa
Hasta que hace poco tiempo se inventara la calculadora electrónica, la máquina de sumar mecánica (o caja registradora) era un objeto muy común en las tiendas y oficinas. Exceptuando unos pocos refinamientos, esta máquina no sufrió modificaciones esenciales durante trescientos años, funcionando mediante una serie de engranajes y ruedecillas. Muy pronto se advirtió el potencial comercial de la calculadora. Pascal inventó la primera máquina de sumar para que su padre, que era inspector de hacienda, la utilizara en su despacho. Cuando Leibniz desarrolló la multiplicación y la división, la calculadora se introdujo en el mundo empresarial



Sumador medio
Es un dispositivo para sumar dos números binarios mediante una combinación de puertas lógicas. Se llama sumador medio porque no puede arrastrar los dígitos que a menudo hay que llevarse al sumar números. Intente sumar 1 y 1. Recuerde que una puerta NOT invierte el 1 en 0 y el 0 en 1. Para que una puerta AND produzca una salida 1, las dos entradas han de ser 1. La salida de una puerta OR será 1 si una entrada o ambas son 1. La salida sólo será 0 si ambas entradas son 0. La ilustración muestra el camino que siguen los dígitos

Mark Watkinson



Sumador completo

El sumador completo es el circuito que, en una suma binaria, suma los dígitos. Consta de dos sumadores medios y una puerta OR, y admite tres entradas: los dos dígitos y el posible dígito arrastrado. La aritmética se realiza en dos etapas: primero se suman los dos dígitos y, después, a este resultado se agrega el dígito "arrastrado". Estos dos pasos se producen en la secuencia de puertas donde los dos sumadores medios se unen mediante una puerta OR

Mark Watkinson

Este circuito está prácticamente allí. El único problema es que una entrada de dos unos, aunque da una "suma" correcta de 0, es incapaz de producir una señal "llevo". Sin embargo, una puerta AND adicional, conectada en paralelo con las dos entradas, producirá una señal "llevo" cuando, y sólo cuando, ambas entradas sean verdaderas. Lo que sigue es la tabla de verdad para el circuito de la ilustración, denominado *sumador medio*:

X (entrada uno)	Y (entrada dos)	LL (salida "llevo")	S (salida "suma")
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	0

Se denomina *sumador medio* porque, en cierto sentido, sólo es adecuado a medias. Es apto para el caso de que todo lo que deseemos hacer sea sumar una única columna de dos dígitos binarios. No obstante, por lo general desearemos sumar dos bytes de datos, y cada byte contiene ocho dígitos. El sumador que se cuidara de la columna de dígitos binarios situada más sobre la derecha, bien podría ser un sumador medio. Sin embargo, todos los demás sumadores a su izquierda habrían de aceptar tres entradas: los dos dígitos de "su" columna y el que se lleve desde la columna situada inmediatamente a su derecha. Observemos esta suma:

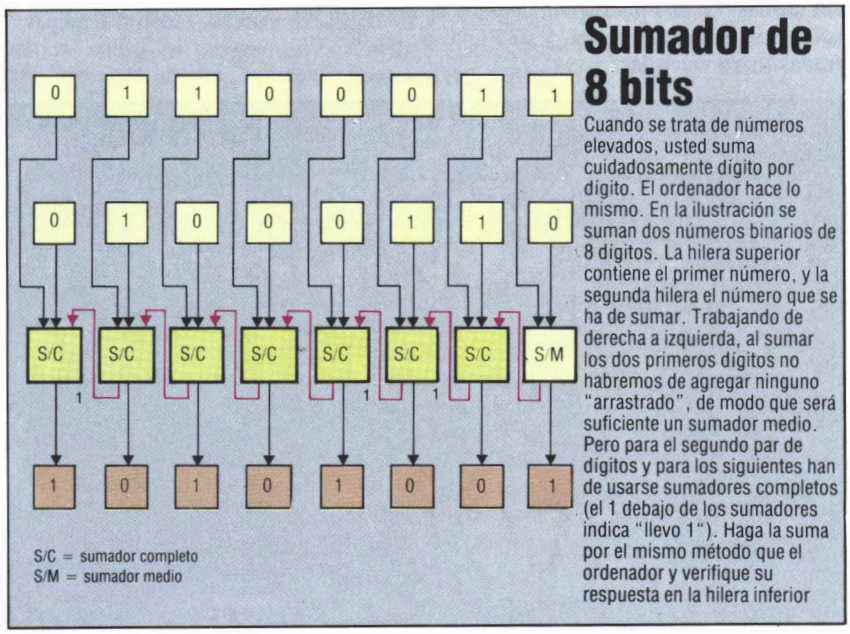
```
  011
+ 111
 1010
```

Al sumar la columna de "los unos" decimos 1 más 1 es 0, llevamos 1 y escribimos un 0 bajo la columna de "los unos". Cuando sumamos la columna de "los dos", decimos 1 más 1 es 0, llevo 1, más lo que nos llevamos de "los unos", que es 1, llevo 1. Escribimos un 1 bajo la columna de "los dos" y nos llevamos 1 a la columna de "los cuatros". Aquí decimos 0 más 1 igual a 1, más lo que me llevaba de la columna de "los dos", que es 0, me llevo 1. Colocamos el 0 en la columna de "los cuatros" y llevamos 1, que escribimos bajo la columna de "los ochos". En otras palabras, la tabla de verdad para un *sumador completo* capaz de manipular los dígitos que se llevan, además de los dos dígitos binarios, sería la siguiente:

X	Y	LL (ent.)	LL (sal.)	S
0	0	0	0	0
1	0	0	0	1
0	1	0	0	1
0	0	1	0	1
0	1	1	1	0
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1

Un sumador completo se puede obtener a partir de la utilización de dos sumadores medios y una puerta OR adicional. La "salida llevo" de cada sumador completo se conecta directamente con la "entrada llevo" del sumador a su izquierda, y de esta forma se pueden encadenar tantos sumadores completos como sea necesario.

En los microordenadores modernos, la mayoría de las sumas y otras operaciones aritméticas se efectúan con grandes cantidades de circuitos sumadores conceptualmente idénticos a los que hemos descrito anteriormente. En su mayoría, estos circuitos sumadores están incluidos, formando sólo una parte de él, en el sistema de circuitos de la CPU. Antes de que llegara la época de la integración a gran escala, que culminó en el microprocesador, comúnmente se utilizaban circuitos integrados más sencillos que apenas contenían unas pocas puertas. Estos circuitos se denominan chips TTL. (TTL significa *Transistor-Transistor Logic*: lógica de transistor a transistor, que alude a la forma en que efectúan la conmutación lógica los transistores que están acoplados entre sí directamente.) El interior de una CPU típica consiste en un único chip de silicio que tiene incorporadas pequeñas zonas de memoria RAM y ROM, grandes cantidades de circuitos de conmutación y una parte denominada ALU o *Arithmetic and Logic Unit* (unidad aritmético lógica). La ALU es la parte de la CPU que contiene todas las puertas lógicas y los sumadores necesarios para que el ordenador efectúe cálculos y tome decisiones lógicas.



Sumador de 8 bits

Cuando se trata de números elevados, usted suma cuidadosamente dígito por dígito. El ordenador hace lo mismo. En la ilustración se suman dos números binarios de 8 dígitos. La hilera superior contiene el primer número, y la segunda hilera el número que se ha de sumar. Trabajando de derecha a izquierda, al sumar los dos primeros dígitos no habremos de agregar ninguno "arrastrado", de modo que será suficiente un sumador medio. Pero para el segundo par de dígitos y para los siguientes han de usarse sumadores completos (el 1 debajo de los sumadores indica "llevo 1"). Haga la suma por el mismo método que el ordenador y verifique su respuesta en la hilera inferior

Navidad en Basic

Introducimos nuevas órdenes con las cuales se puede manipular la información y escribimos un programa para calcular cuántos días faltan para la Navidad

En este programa pasamos revista a todos los temas que hemos incluido hasta ahora en nuestro curso de programación; asimismo, introducimos algunas sentencias de BASIC nuevas y eficaces. La finalidad del programa es calcular el número de días que faltan para que llegue la Navidad.

Si observa el listado del programa, verá que empieza con una lista de las variables utilizadas. Ciertamente, esta práctica no es esencial, pero sí es aconsejable para que sus programas resulten más fáciles de comprender cuando se encuentre con ellos algún tiempo después. Algunas versiones de BASIC permiten dar a las variables nombres largos, por ejemplo DIA, en lugar de una única letra, como hemos venido utilizando hasta ahora. Si usted es afortunado y posee un BASIC que admite variables de nombre largo, escoja para ellas palabras que posean algún significado. DIA, MES$ o AÑO son mucho mejor que D, M$ o A. Si no tiene alternativa posible porque su BASIC no admite variables de nombre largo, conseguirá casi la misma facilidad de lectura si procede al listado de las variables al inicio del programa.

Al ejecutar el programa, lo primero que aparecerá en la pantalla serán las sentencias PRINT que comienzan a partir de la línea 230. Éstas describen brevemente lo que hará el programa y a continuación preparan al usuario para que digite la fecha en la forma indicada, usando comas para separar día, mes y año.

La primera sentencia desconocida aparecerá en la línea 300. Es una sentencia DIMension (dimensión). Se utiliza para determinar el número de ítems o de elementos de que consta la matriz denominada X. Una matriz, también llamada variable subíndice, es como una variable común; la diferencia estriba en que su caja contiene diversos compartimientos. En la línea 300 estamos creando una variable denominada X que dentro de su caja posee 13 compartimientos. En un próximo apartado retomaremos el tema de las matrices y la sentencia DIM con más detalle.

```
310 INPUT D, M$, A
```

Esta línea es una sentencia INPUT normal, sólo que espera tres entradas. D es una variable numérica y contendrá la fecha de hoy. A, otra variable numérica, corresponderá al año. M$ es algo diferente. Es una "variable alfanumérica", característica que se indica mediante el signo $ (dólar). Una variable alfanumérica acepta tanto caracteres del teclado como números. Si, por ejemplo, digitamos 23, ENERO, 1984, a la variable D se le asignará el valor 23, a la variable M$ la serie de caracteres ENERO y a la variable A el valor 1984.

```
330 GOSUB 550 REM RUTINA 'N.º DEL MES'
```

Esta sentencia indica que el programa debe bifurcarse hasta la subrutina que comienza en la línea 560. Observe también que en la misma línea se ha incluido una REMark (observación). Si en la línea hay espacio, no siempre es necesario colocar las REM en una nueva línea. Esta subrutina en particular la utiliza el programa una sola vez y, en rigor, podría igualmente haberse incorporado en el programa principal. Al convertirla en subrutina no hemos hecho más que separar esta parte del resto del programa.

Originalmente, cuando este programa se escribió por primera vez, para el mes se utilizó un número y esta parte del programa no era necesaria. Luego se decidió permitir que se diera entrada al mes digitándolo como una palabra completa. Para convertir el mes escrito en palabras a su equivalente en número, se escribió por separado el programa extra que ahora constituye esta subrutina. El único cambio que hubo que introducir en el programa principal (original) fue la adición de una sentencia GOSUB. Esta subrutina ilustra con qué sencillez se pueden estructurar los programas en bloques y enlazar entre sí utilizando las sentencias GOSUB y RETURN.

La subrutina en sí misma es muy sencilla, pero ilustra lo inteligente que es el BASIC para manipular series de caracteres. Supongamos que hemos dado entrada a ENERO como la parte mes de la sentencia INPUT. A la variable M$ se le tendría que asignar, en tal caso, la serie de caracteres ENERO. Una vez que se ha inicializado M = 0, la segunda línea de la subrutina es:

```
560 IF M$ = "ENERO" THEN LET M = 1
```

Esta sentencia compara el contenido de M$ con los caracteres comprendidos entre las comillas dobles. Si son los mismos (como en este caso), la línea continúa para establecer el valor de la variable numérica M en 1. No confunda la variable M con la variable M$. Son distintas. Sólo una de ellas puede contener una variable alfanumérica: ¡la que posee el signo $! Después de verificar si M$ es la misma que ENERO, el programa se dirige a la línea siguiente y verifica si el contenido de M$ es el mismo que FEBRERO. Si no lo es, M no se establece en 2. El valor de la variable M sólo se establece cuando la correspondencia es correcta y ese valor es el mismo que el número del mes: 1 para enero, 3 para marzo y así sucesivamente.

Al llegar a la línea 680 el BASIC retorna (RETURN) al programa principal, a la línea siguiente a la sentencia GOSUB. Se trata de la línea 340, de donde si

M = 0 (nos hemos equivocado al digitar el mes) se vuelve a la línea 230.

Entre las líneas 350 y 370 hay un bucle FOR-NEXT. Éste incrementa el valor de I, comenzando por 1 y siguiendo hasta 13. La variable I se utiliza como el subíndice de la matriz X de la línea 360. Se la debe estudiar con suma atención.

```
360 READ X(I)
```

READ es una nueva sentencia que hasta ahora no habíamos visto nunca. READ se utiliza siempre con su correspondiente sentencia DATA. La sentencia DATA para esta línea está en la línea 510:

```
510 DATA 31, 28, 31, 30, 31, 30, 31, 31, 30,
    31, 30, 25, 0
```

Estos números, a excepción de los dos últimos, corresponden a la cantidad de días de que consta cada mes. Las dos líneas equivalen a 13 sentencias LET separadas.

```
LET X(1) = 31
LET X(2) = 28
LET X(3) = 31
LET X(4) = 30
LET X(5) = 31
LET X(6) = 30
LET X(7) = 31
LET X(8) = 31
LET X(9) = 30
LET X(10) = 31
LET X(11) = 30
LET X(12) = 25
LET X(13) = 0
```

Antes de retomar este programa, consideremos un programa más pequeño y mucho más simple. Es el siguiente:

```
10 READ A, B, C
20 LET D = A + B + C
30 PRINT D
40 DATA 5, 10, 20
```

Aquí, la sentencia READ de la línea 10 lee el primer ítem de datos (DATA) de la línea 40 y "escribe" su valor en la primera variable. En otras palabras, asigna valor 5 a la variable A. Luego READ lee el siguiente ítem de datos y lo coloca en la variable que viene a continuación.

Este programa hace que A = 5, B = 10 y C = 20. Luego suma estos valores y asigna el resultado a la variable D. Después este resultado, 35, se imprime (PRINT) en la línea 30.

Volvamos al programa "navideño". La primera vez que se realiza el bucle que comienza en la línea 350, el valor de I se establece en 1. Por lo tanto, la línea 360 equivale a READ X(1). El correspondiente ítem de datos de la línea 510 es 31 (el primer ítem). En consecuencia, X(1) se establece en 31.

La segunda vez que se realiza el bucle, I se convierte en 2, de manera que la línea 360 equivale a READ X(2). El siguiente dato de la línea DATA es 28. Esto significa que X(2) se establece en 28. De este modo, los 13 "compartimientos" de la variable subíndice X se llenan con el número de días de cada mes, excepto el duodécimo, que sólo tiene 25 días, y el decimotercero, cuyo número es 0.

```
390 GOSUB 750 REM RUTINA 'AÑO BISIESTO'
```

```
100  REM   LISTA DE VARIABLES
110  REM
120  REM   D = FECHA DE HOY
130  REM   M$ = NOMBRE DEL MES
140  REM   A = AÑO
150  REM   I = INDICE 1
160  REM   X( ) = (CON INDICE) NUMERO DE DIAS DE CADA MES
170  REM   R = DIAS QUE FALTAN
180  REM   M = NUMERO DEL MES
190  REM   L = INDICE 2
200  REM   Z = VALOR ENTERO DE A/4
210  REM
220  REM
230  PRINT "ESTE PROGRAMA CALCULA"
240  PRINT "CUANTOS DIAS FALTAN PARA NAVIDAD"
250  PRINT "PARA FECHAS HASTA EL"
255  PRINT "24 DE DICIEMBRE"
260  PRINT
270  PRINT "DIGITE DIA DE HOY, MES, AÑO"
280  PRINT "P. EJ. 12, JULIO, 1984"
290  PRINT
300  DIM X(13)
310  INPUT D, M$, A
320  REM
330  GOSUB 550 REM RUTINA 'N. DEL MES'
340  IF M = 0 THEN GOTO 230    REM EN CASO DE MES
     EQUIVOCADO
350  FOR I = 1 TO 13
360  READ X(I)
370  NEXT I
380  REM
390  GOSUB 750 REM RUTINA 'AÑO BISIESTO'
400  REM
410  LET R = X(M) - D
420  FOR L = 1 TO 12
430  LET M = M + 1
440  LET R = R + X(M)
450  NEXT L
460  REM
470  IF R = 1 THEN GOTO 500
480  PRINT "FALTAN ";R;" DIAS HASTA NAVIDAD"
490  GOTO 520
500  PRINT "FALTA 1 DIA PARA NAVIDAD"
510  DATA 31,28,31,30,31,30,31,31,30,31,30,25,0
520  END
530  REM
540  REM
550  LET M = 0
560  IF M$ = "ENERO" THEN LET M = 1
570  IF M$ = "FEBRERO" THEN LET M = 2
580  IF M$ = "MARZO" THEN LET M = 3
590  IF M$ = "ABRIL" THEN LET M = 4
600  IF M$ = "MAYO" THEN LET M = 5
610  IF M$ = "JUNIO" THEN LET M = 6
620  IF M$ = "JULIO" THEN LET M = 7
630  IF M$ = "AGOSTO" THEN LET M = 8
640  IF M$ = "SEPTIEMBRE" THEN LET M = 9
650  IF M$ = "OCTUBRE" THEN LET M = 10
660  IF M$ = "NOVIEMBRE" THEN LET M = 11
670  IF M$ = "DICIEMBRE" THEN LET M = 12
680  RETURN
690  REM
700  REM
710  REM
720  REM NOTA: ESTA SUBRUTINA NO VERIFICA
730  REM          LOS AÑOS MULTIPLOS DE 100 (SIGLOS)
740  REM
750  LET A = A / 4
760  LET Z = INT (A)
770  IF A - Z = 0 THEN GOTO 790
780  RETURN
790  LET X(2) = X(2) + 1
800  RETURN
```

Esta línea dirige el programa hasta una subrutina que verifica si el año al que se ha dado entrada es bisiesto o no. Veamos cómo funciona.

```
750 LET A = A/4
760 LET Z = INT(A)
770 IF A - Z = 0 THEN GOTO 790
780 RETURN
790 LET X(2) = X(2) + 1
800 RETURN
```

Son años bisiestos los que se pueden dividir por 4. Si se trata de un siglo, también debe ser divisible por 400 para que se considere como un año bisiesto. En aras de conseguir la mayor simplicidad posible, no hemos intentado verificar el siglo, sino sólo la divisibilidad por 4.

La línea 750 establece el valor de A en el antiguo valor de A (del año) dividido por 4. La nueva A será un número entero si el año es exactamente divisible por 4. De lo contrario, tendrá una fracción decimal.

La línea 760 utiliza la función INT para hallar el valor "entero" de A. Si el número al que se ha dado entrada para el año era 1985, el nuevo valor de A = 496,25 será redondeado por la función INT privándole de los decimales. Esto se indica con INT(A), donde en vez de A podemos poner directamente un número. De modo que LET Z = INT (496,25) establecería el valor de Z en 496.

En la línea 770 se resta el valor de Z al valor de A y se verifica si el resultado es 0. De ser así, significa que el año es un año bisiesto (puesto que ahora A no posee fracción decimal). En este caso, el programa se bifurca hacia la línea 790 mediante GOTO. La línea 790 le suma 1 al segundo ítem de la matriz (el segundo ítem era 28, el número de días de un mes de febrero normal).

Si el resultado de la resta de la línea 770 no era 0, X(2) se deja tal como está y la subrutina retorna (RETURN) al programa principal, a la línea 400.

La línea 400 es otra REM utilizada para espaciar el programa y facilitar su lectura. La siguiente línea que verdaderamente hace algo es la 410, donde R es la variable que alberga el número de días que faltan. Aquí X(M) se establece en el número de días del mes al que se ha dado entrada, y se restan los días digitados. Si, por ejemplo, hemos dado entrada a 12, FEBRERO, 1984, D sería igual a 12 y M sería 2. Por lo tanto, X(M) sería igual a X(2) y el segundo ítem de la matriz X es 29 (se le ha sumado 1 porque 1984 es un año bisiesto). En consecuencia, R se establecerá en 29 - 12, es decir, 17, el número de días que faltan del mes en curso, febrero.

En la línea 420 empieza otro bucle. Éste está diseñado para incrementar el valor de M. ¿Comprende por qué decimos FOR L = M TO 12? Si M fuera 2, porque hubiésemos dado entrada al mes de FEBRERO, la línea 430 la incrementará a 3. Luego la línea 440 establece a R, el número de días que faltan, en la antigua R más X(M). Ésta ahora equivale a X(3), ya que M se ha incrementado en 1. El valor de X(3) es 31, el número de días del mes de marzo. La línea 440, por lo tanto, establece el nuevo valor de R en 17 + 31 (17 era el resultado de haber restado 12 a 29). La próxima vez que se efectúe el bucle, M pasará a valer 4 y se le sumará al antiguo valor de R el número de días de abril, X(4). De manera que la variable R se convertirá en 17 + 31 + 30.

El último circuito a través del bucle se produce después de haberse alcanzado el límite máximo de 12 (en la línea 420). La línea 430 le suma 1 al valor de M por última vez, estableciéndola en 13. El valor de X(13) es el último ítem de la sentencia DATA, 0. Hemos agotado la suma de todos los días posibles de meses enteros. Finaliza el bucle y pasa a la línea:

```
470 IF R = 1 THEN GOTO 500
```

Esta línea simplemente verifica si no falta nada más que un día para Navidad, de modo que obtengamos en pantalla una oración correcta desde el punto de vista gramatical. Si R no es 1 es porque falta más de un día, con lo cual la sentencia PRINT de la línea 480 será gramaticalmente correcta.

Y esto es todo. La versión de BASIC que hemos empleado debería funcionar en la mayoría de los ordenadores (ver el recuadro "Complementos al BASIC"), excepto, quizá, para la subrutina "año bisiesto". El BASIC es muy incoherente en cuanto a la forma de utilizar LET. Si líneas como IF M$ = "SEPTIEMBRE" THEN LET M = 9 no funcionan en su ordenador, se le ofrece la opción de reescribir la subrutina como sigue:

```
560 IF M$ = "ENERO" THEN GOTO 900
570 IF M$ = "FEBRERO" THEN GOTO 910
580 IF M$ = "MARZO" THEN GOTO 290


900 LET M = 1
905 RETURN
910 LET M = 2
915 RETURN
920 LET M = 3
925 RETURN
(... y así sucesivamente)
```

Esta solución es, sin duda, poco elegante y, con tantos GOTO y RETURN, es más difícil de seguir. Sin embargo, demuestra que por lo general existen diversas maneras de solucionar cualquier problema.

Complementos al BASIC

DIM

Si el programa se ha de ejecutar en un ordenador BBC, Dragon u Oric, se debe modificar la línea 300 para que diga: 300 DIM X(14)

INPUT

En el Spectrum no se puede dar entrada a una serie de valores separados mediante comas. Por tanto, se debe modificar la línea 310 de la siguiente manera: 310 INPUT D, y agregar las líneas 312 INPUT M$ y 314 INPUT A. Como a la fecha no se le da entrada en tres etapas, se deben modificar las sentencias PRINT para que digan: 280 PRINT "¿DIA?", agregando 311 PRINT "¿MES?" y 313 PRINT "¿AÑO?"

REM

Cuando las sentencias REM se colocan al final de otra sentencia, se deben separar mediante dos puntos (:) si se utiliza un BBC, un TI 99/4A o un Spectrum. Por ejemplo, la línea 330 dirá: 330 GOSUB 550: REM RUTINA 'N. DEL MES'

END

El Spectrum no dispone de sentencia END, de modo que se ha de modificar la línea 520 para que se lea: 520 GOTO 1000, agregando la línea 1000 REM FIN DEL PROGRAMA

Grabando

El gran temor del programador es perder la obra de arte que ha creado en la pantalla. Las cassettes ofrecen la solución

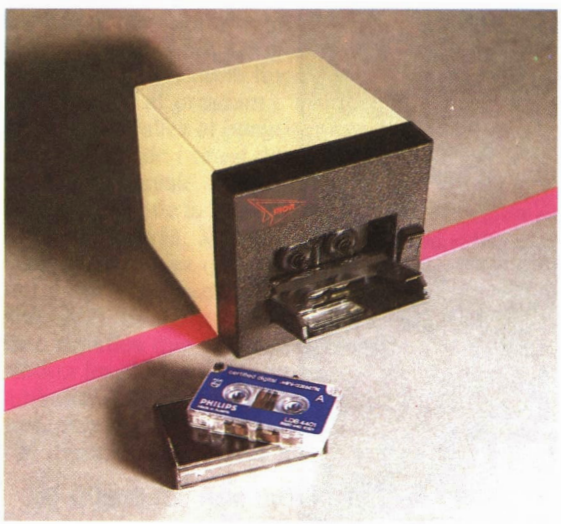
Altavoz
En la mayoría de las grabadoras de cassette el altavoz queda desconectado cuando se acopla el aparato a un ordenador o a un equipo de alta fidelidad

Cabeza de borrado
Estando en modalidad de grabación, elimina de la cinta cualquier señal grabada previamente

Contador de vueltas
Un dispositivo esencial si se piensa almacenar diversos programas en una misma cinta

Cabeza de grabación/reproducción
Esta cabeza de doble función graba las señales sonoras en la cinta magnética y las reproduce

La Hobbit
La Hobbit de la Ikon es una grabadora de cassette exclusiva, es decir, diseñada exclusivamente para almacenar programas de ordenador. Es superior a la grabadora de cassette normal, porque la Hobbit está totalmente controlada por un software. No es necesario pulsar "wind", "rewind", "play" ni "record", puesto que la Hobbit realiza todas estas funciones por sí misma. Si desea cargar (LOAD) un programa, deberá digitar el nombre del mismo y la Hobbit buscará en su propio catálogo hasta localizarlo. Luego rastrea la cinta hasta colocarla en la posición correcta



Ian McKinnell

Eje de arrastre
Eje de gran precisión que rota a una velocidad cuidadosamente controlada para hacer pasar por la cabeza de grabación/reproducción 1 7/8 de pulgada de cinta por segundo

Cuando se digita un programa en un ordenador personal, ya sea tomándolo de un libro, de una revista o confeccionándolo uno mismo, la información se almacena en la RAM. Mientras el ordenador está encendido, el programa está a salvo; pero desaparece en el momento en que usted desconecta la máquina. Esto puede suponerle un gran contratiempo: ¡toda una sesión de programación perdida para siempre! La próxima vez que desee utilizar ese programa habrá de digitarlo todo de nuevo. Para evitar este problema, los fabricantes de ordenadores personales suelen incorporar un procedimiento en virtud del cual se puede almacenar con carácter más permanente el contenido de la memoria del ordenador.

Volumen
Regula el volumen de reproducción y ha de determinarse con sumo cuidado, ya que de lo contrario se corre el riesgo de que las cassettes no se carguen correctamente

El procedimiento más común de almacenamiento es el de la cinta de cassette. Escogido originalmente por su gran disponibilidad y su bajo precio, en la actualidad casi todos los ordenadores personales que se venden incorporan este sistema. La forma en que cada ordenador almacena su información varía ligeramente; por ejemplo, un programa creado y almacenado en un ordenador Commodore no se puede cargar en un Spectrum. Sin embargo, el procedimiento que se utiliza para convertir el programa en una forma almacenable, es casi universal.

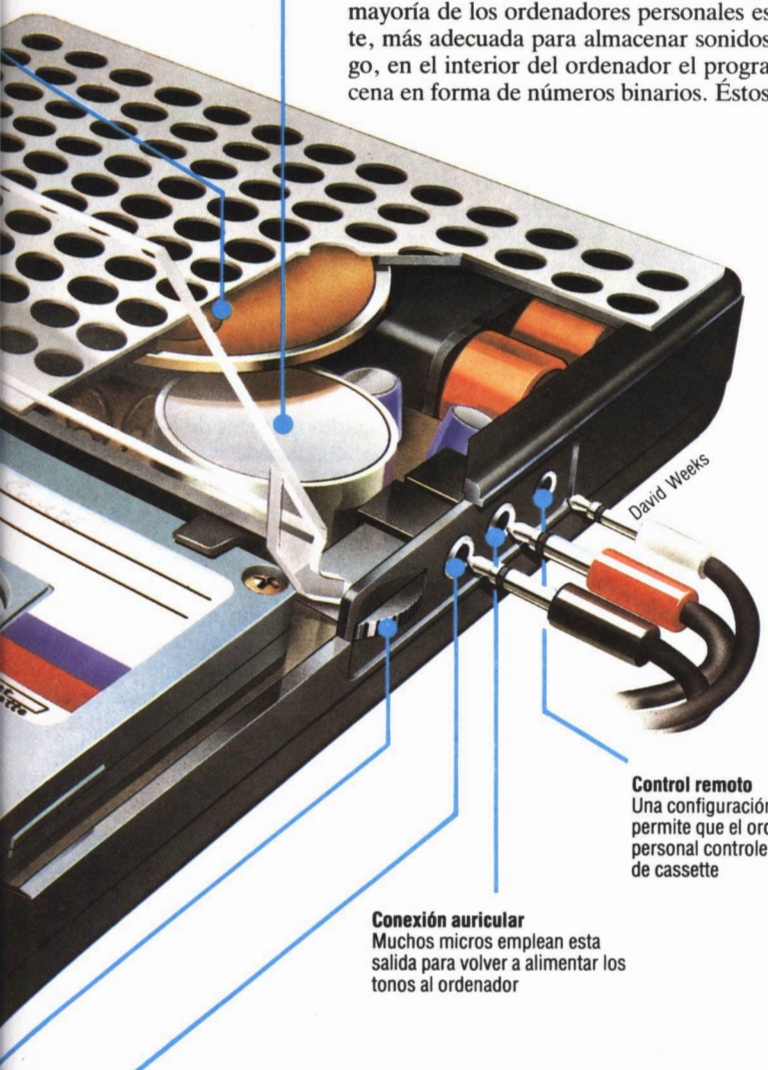
La clase de grabadoras de cassette que utilizan la mayoría de los ordenadores personales es, obviamente, más adecuada para almacenar sonidos; sin embargo, en el interior del ordenador el programa se almacena en forma de números binarios. Éstos han de convertirse en sonidos para que el ordenador pueda reconocer la diferencia entre un bit que "se enciende" y un bit que "se apaga": los ceros y los unos binarios. Para hacerlo, el procedimiento más sencillo consiste en crear un sonido que represente un 1 y otro que represente un 0.

Por lo general, estos sonidos se escogen de manera que sean un tono de 2 400 ciclos para el 1 y un tono de 1 200 ciclos para el 0.

Cuando se digita en el ordenador la orden SAVE, lo primero que se grabará en la cinta son algunos segundos de tono constante. Esto se hace para que al reproducirle la cinta al ordenador en algún otro momento éste pueda percibir la diferencia entre la cinta virgen y la sección que aloja el programa. La primera información real que se graba es la serie de tonos que representan a los caracteres del nombre que le hemos dado al programa. Cada carácter está compuesto de un byte (ocho bits en total), de modo que para representar un carácter son necesarios ocho tonos. Sin embargo, para indicar el comienzo y el final de cada byte el ordenador normalmente coloca un tono extra a ambos extremos. Éstos se denominan *bits de principio y final* y su valor siempre es el mismo: ya 1 o 0, según de qué ordenador se trate.

El propio programa se almacena de forma bastante similar, excepto en ocasiones que se lo divide en segmentos. Normalmente estos segmentos son de 256 bytes cada uno y suelen incluir una información adicional que le permite al ordenador asegurarse de que está recargando la información correcta. El sistema empleado aquí es muy sencillo y se denomina *suma de comprobación*. El primer byte contiene el número de bytes de que consta el segmento, y el último byte contiene un número calculado especialmente que representa el total de todos los bytes sumados. Cuando el ordenador lee la cassette verifica si las cifras que figuran en la cinta coinciden con las que él mismo ha calculado y, en caso de que no coincidan, informa del error al usuario.

Algunos sistemas de cassette, como el del modelo BBC Micro, amplían esta comprobación hasta el extremo de otorgar un nombre y un número a cada segmento del programa. En el caso de que se produzca un error, no hay problema alguno, pues se puede rebobinar la cinta unas pulgadas y volver a intentarlo. En notorio contraste, otros sistemas de cassette ni siquiera muestran el nombre del programa que se está cargando.



Motor
El motor acciona el eje de arrastre a una velocidad constante y también hace girar los tambores para bobinar y rebobinar la cinta

Control remoto
Una configuración muy útil que permite que el ordenador personal controle a la grabadora de cassette

Conexión auricular
Muchos micros emplean esta salida para volver a alimentar los tonos al ordenador

Conexión micrófono
Utilizada a menudo para la entrada de datos del ordenador en la grabadora. No obstante, sólo debe emplearse si no se dispone de conexiones para DIN y auxiliares. Cuando se utiliza esta entrada es necesario realizar un cuidadoso ajuste de los mandos de tono y volumen

La velocidad baudio

La velocidad a la cual se producen y se graban los tonos en la cinta se suele denominar, incorrectamente, *velocidad baudio*. El nombre proviene del código Baudot, que se utilizaba en las primeras versiones del telégrafo eléctrico, y que realmente se refiere al número de veces por segundo que cambia la señal. Una medida más exacta sería el número de bits que se graban por segundo. A mayor velocidad (entre 300 y 1 200 bits por segundo), más rápidamente se almacenarán los programas en cinta y menos tiempo se tardará en volver a cargarlos en el ordenador. Lamentablemente, la mayor velocidad de almacenamiento de los tonos incide en la fiabilidad; una velocidad de 1 200 bits por segundo es a la vez fiable y rápida. Algunos sistemas ofrecen dos velocidades, por lo general una velocidad lenta superfiable de 300 bits por segundo y una velocidad rápida de 1 200 o 2 400 bits por segundo. Para evitar imprevistos, los programas valiosos se pueden guardar en dos copias, una de cada tipo.

La cinta de cassette ha de ser de buena calidad; no hay ningún inconveniente en usar las cintas de sonido normales en vez de las cassettes especiales, siempre y cuando sean de una marca prestigiosa y su longitud no exceda de C-60.

La capacidad aproximada de una determinada longitud de cinta se establece dividiendo por 10 la velocidad de la interface para cassette. El resultado de esta división corresponderá a la cantidad de bytes que se pueden almacenar en la cinta por segundo; una C-60 de 30 minutos por cada lado, en la que la interface trabaje a 1 200 bits por segundo, podría retener 432 Kbytes de programa.

Términos clave

Almacenamiento seguro

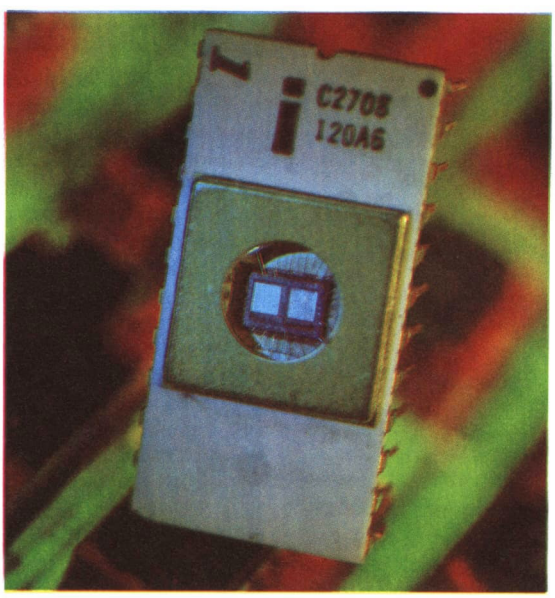
El ordenador puede almacenar en su memoria miles de bytes de información y recordar dónde está localizado cada uno de ellos

La memoria del ordenador se puede describir en términos de almacenamiento a corto plazo y almacenamiento a largo plazo. La de largo plazo no pierde la información almacenada y la puede retener durante largos períodos, aun después de haberse apagado el ordenador. En esta categoría entran las cintas magnéticas y los discos flexibles.

Los ordenadores también necesitan una memoria rápida a corto plazo para el almacenamiento temporal de programas y resultados.

Otra forma de describir la memoria del ordenador consiste en considerarla en términos de memoria interna o memoria externa. La memoria interna está localizada dentro del ordenador y por lo general es totalmente "electrónica", mientras que la memoria externa es periférica o exterior al ordenador. La memoria externa por lo general es parcialmente mecánica, y comprende dispositivos tales como cassettes, unidades de disco flexible e incluso tarjetas de papel perforadas.

Generalmente la memoria electrónica interna se denomina memoria principal, mientras que la memoria externa se considera como una memoria secundaria o



EPROM
El problema de las memorias ROM normales es que poseen un contenido "incorporado", determinado en una etapa de la fabricación y que no se puede modificar. Las unidades EPROM (Erasable Programmable Read Only Memory: memoria de lectura sólo susceptible de borrar y programar) son mucho más flexibles. Una vez programadas, se pueden volver a programar borrando los contenidos y volviendo a "escribir" en ellas. Las unidades EPROM incorporan una "ventana" de sílice que permite que penetren los rayos ultravioletas, haciendo que se descarguen los condensadores que almacenan los bits en la EPROM. En ausencia de luz ultravioleta, los condensadores conservan su carga indefinidamente y se retienen los contenidos de la memoria

de apoyo. En la actualidad existen dos variedades principales de memoria interna: RAM y ROM.

Tanto la RAM como la ROM son dispositivos totalmente electrónicos, fabricados en forma de chips de silicio y empaquetados en estuches plásticos rectangulares con cables paralelos de hojalata o chapados en plata. Existe mucha similitud en cuanto a la forma en que son seleccionados y "dirigidos" por la CPU del ordenador, pero de ello nos ocuparemos más adelante.

La principal diferencia funcional es que los chips de memoria ROM se utilizan para almacenar programas con carácter permanente. El patrón de unos y ceros de cada dirección de memoria se fija y se determina durante el proceso de fabricación y, en consecuencia, no se puede modificar. Las ROM son las "bibliotecas de referencia" del mundo de la informática. El ordenador puede consultar el contenido de la ROM, pero no puede "escribir" nada en ella.

ROM corresponde a las siglas de Read Only Memory (memoria de lectura solamente); la palabra lectura describe lo que hace el ordenador cuando "accede a" o recupera información de la memoria. Existen diversos tipos de ROM que difieren ligeramente entre sí; algunos de ellos permiten eliminar o borrar especialmente el programa interno y se pueden volver a programar. No obstante, una ROM bastante clásica es la 2364 de Intel. Este chip es una ROM de 65 536 bits, organizados en 8 Kbytes de 8 bits. Esto significa que los 64 Kbits están agrupados en bytes de 8 bits y que cada localización "dirigible" accede a o lee un byte entero. En matemáticas, 1 K = 2^{10} (dos a la décima potencia) o 1 024, de modo que 64 K = 64 × 1 024 o 65 536.

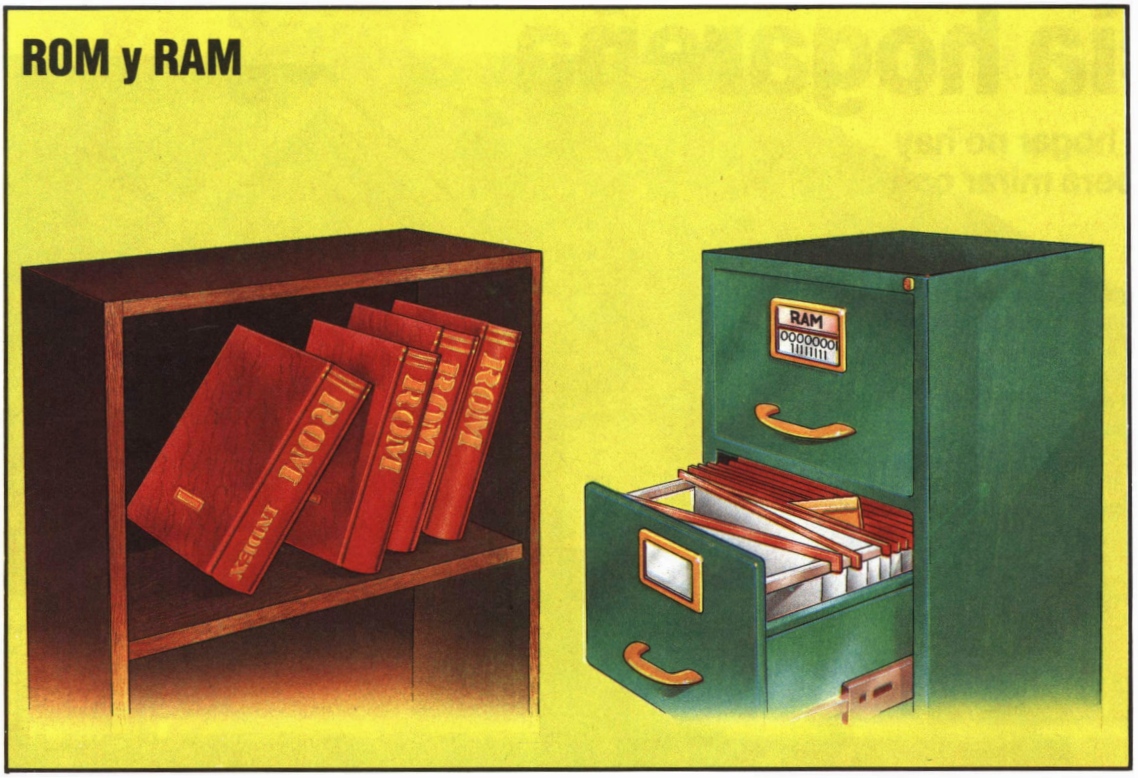
El ordenador, por lo tanto, ha de ser capaz de seleccionar cualquiera de las 8 192 (8 K) localizaciones de dirección. Una atenta lectura de las especificaciones del chip 2364 revela que posee 28 patillas, con una reservada para la fuente de alimentación eléctrica de +5 voltios y otra para la conexión a tierra. Esto deja un total de 26 patillas. Cada byte contiene ocho bits, de modo que cuando se lee un byte del chip, los ocho dígitos de ese byte han de traspasarse mediante cables desde el chip a la CPU. En consecuencia, hay ocho cables para trasladar los bits del byte que se le está leyendo a la CPU. Estos cables se denominan *bus de datos*. Ocho de las patillas del chip se dedican a esta función, una para cada uno de los bits del byte.

Esto deja un total de 18 patillas. Una patilla no es necesaria y no se conecta. Se conserva porque es más fácil fabricar chips con un número par de patillas. Cuatro patillas se utilizan para "seleccionar" el chip de diversas maneras. Éstas son la patilla de "habilitación de salida", la patilla de "habilitación del chip" y las dos patillas de "selección de chip". Estas patillas reciben las señales del ordenador para permitir que el chip sepa cuándo es requerido.

Las 13 patillas restantes son las patillas "de dirección". Cada patilla está conectada a un cable de "bus de direcciones", y éste transporta la dirección del byte requerido, codificada en forma binaria. Trece dígitos binarios pueden dar 2^{13} u 8 192 combinaciones exclusivas de unos y ceros, de modo que las 13 líneas de dirección alcanzan exactamente para seleccionar de forma exclusiva cada uno y la totalidad de los 8 192 bytes almacenados en la ROM.

Las unidades RAM son algo así como las pizarras del mundo de la informática. En ellas los programas y los datos se almacenan con carácter provisional, mientras el ordenador está en funcionamiento, y en ellas también "se escriben" temporalmente los resultados y

ROM y RAM



La memoria a largo plazo
La ROM (Read Only Memory: memoria de lectura solamente) se puede comparar con un libro, dado que se trata de un lugar donde se almacena información con carácter permanente. Tal como no se pueden alterar las palabras de una página impresa, del mismo modo no es posible modificar o suprimir los datos de la ROM

La memoria a corto plazo
La RAM (Random Access Memory: memoria de acceso directo) se parece más a un sistema de archivo que a un libro, puesto que la información se puede modificar y los datos no son permanentes; cuando se apaga el ordenador, la RAM se borra por completo

otros datos. En líneas generales, la memoria RAM es más compleja interiormente que la ROM, porque cada uno de los bits de cada byte de la RAM debe ser capaz de modificarse cada vez que se escribe algo en ellos. Un chip de RAM bastante típico es el Intel 2114. Cada chip de RAM 2114 retiene 4 096 bits de memoria y éstos están organizados en 1 024 *nibbles* (medios bytes) de cuatro bits. Esto significa que cada dirección de localización producirá una salida de 4 bits de datos. Por lo tanto, se necesitarán dos de estos chips para producir un Kbyte completo de información. Cada chip 2114 posee sólo 18 patillas, dos de las cuales se emplean para la toma de tierra y la fuente de alimentación eléctrica. Cuatro se utilizan para las líneas de datos de entrada/salida. Una se usa para la señal de selección del chip (la señal que le indica al chip cuándo se lo requiere o cuándo ha sido "seleccionado") y otra se utiliza para indicarle al chip, una vez que ya ha sido seleccionado, si se está escribiendo en él o si se lo está leyendo. Las diez patillas restantes se emplean para el bus de direcciones. Diez líneas de dirección pueden identificar 2^{10} localizaciones exclusivas, o sea 1 024. Si un ordenador dispusiera de 64 Kbytes de RAM, y si se utilizaran chips Intel 2114, se requeriría un total de 128 chips de RAM, puesto que se necesitarían dos chips para cada Kbyte completo. En la actualidad es más común usar chips de RAM de mayor densidad, que en el mismo espacio alojan más memoria. Utilizando chips de RAM más modernos, como el 4164, se pueden obtener 64 Kbytes de RAM con sólo ocho chips.

Año tras año los chips RAM y ROM van siendo más baratos y más compactos, y en la actualidad es posible que un solo chip contenga 128 Kbits. No obstante, el progreso en cuanto al almacenamiento de densidades aún mayores en un solo chip ya no se produce tan rápidamente como antes. El sistema de circuitos contenido en el diminuto trozo de silicio está llegando a ser tan minúsculo, que las técnicas ópticas que se emplean para "grabar" los circuitos apenas llegan a ser suficientes para ese trabajo. Los chips de memoria de "gran densidad" del futuro probablemente se fabricarán mediante procedimientos de grabación por haces de electrones o por rayos X.

En líneas generales, existen dos tipos de memoria RAM en uso, conocidos como RAM estática y RAM dinámica. Ambos tipos poseen sus ventajas y sus inconvenientes, pero actualmente se utiliza más la RAM dinámica que la estática. Ambas pierden el contenido de la memoria apenas se desconecta la alimentación eléctrica, pero la memoria dinámica requiere que se le "refresquen" los contenidos cada algunos milisegundos. Cada bit de la memoria se ha de refrescar o reescribir sin entorpecer la capacidad de la CPU para acceder a la información que contengan. Esto significa que se debe diseñar un sistema de circuitos de sincronización específico y muy crítico, lo que dificulta la labor del diseñador de circuitos.

La memoria dinámica presenta dos claras ventajas sobre la memoria estática. La dinámica sólo requiere un transistor por bit, frente a los tres transistores que normalmente necesita la memoria estática para cada bit. Esto permite almacenar más memoria en chips más pequeños. La mayoría de los chips de RAM dinámica poseen sólo 16 patillas. La otra ventaja de las unidades RAM dinámicas es que consumen menos energía que las estáticas. Por lo tanto, generan menos calor y requieren fuentes de alimentación eléctrica más pequeñas y más baratas.

La ventaja de la RAM estática radica en la simplicidad del diseño de su circuito. Una vez escritos los contenidos de la memoria, permanecen en ella sin que sea necesario refrescarlos. Cada celda de memoria de un bit requiere tres transistores, de modo que resulta difícil conseguir las elevadas densidades que permite la RAM dinámica. Asimismo, la RAM estática consume más energía, y el calor extra generado dificulta la labor del sistema de refrigeración del ordenador, lo que puede hacer necesaria la utilización de un ventilador, con el consiguiente encarecimiento del diseño.

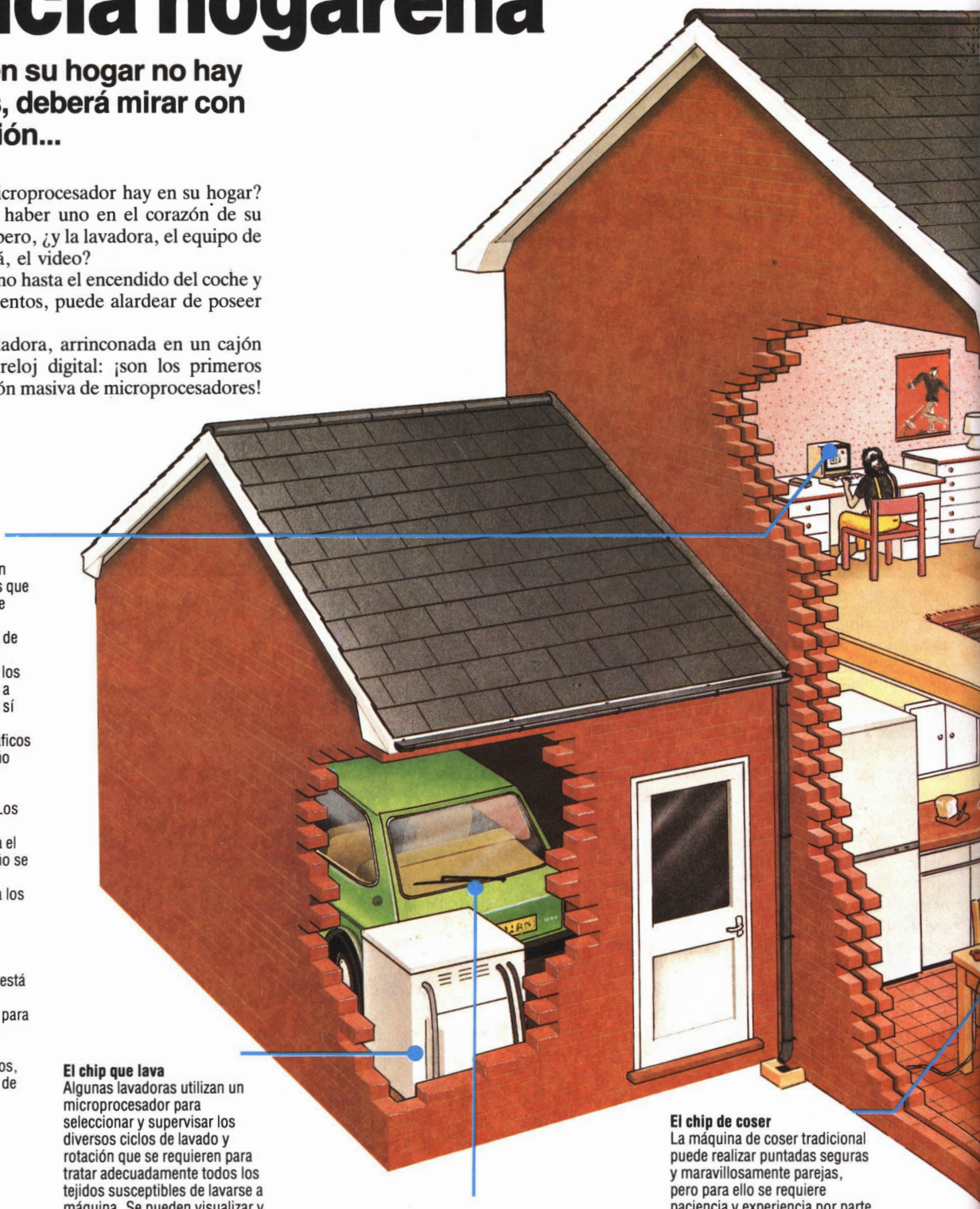
Ciencia hogareña

Si cree que en su hogar no hay ordenadores, deberá mirar con mayor atención...

¿Cuántos chips de microprocesador hay en su hogar? Por supuesto, puede haber uno en el corazón de su ordenador personal; pero, ¿y la lavadora, el equipo de alta fidelidad o, quizá, el video?

Todo, desde el horno hasta el encendido del coche y su tablero de instrumentos, puede alardear de poseer un chip.

No olvide la calculadora, arrinconada en un cajón del escritorio, o su reloj digital: ¡son los primeros ejemplos de fabricación masiva de microprocesadores!

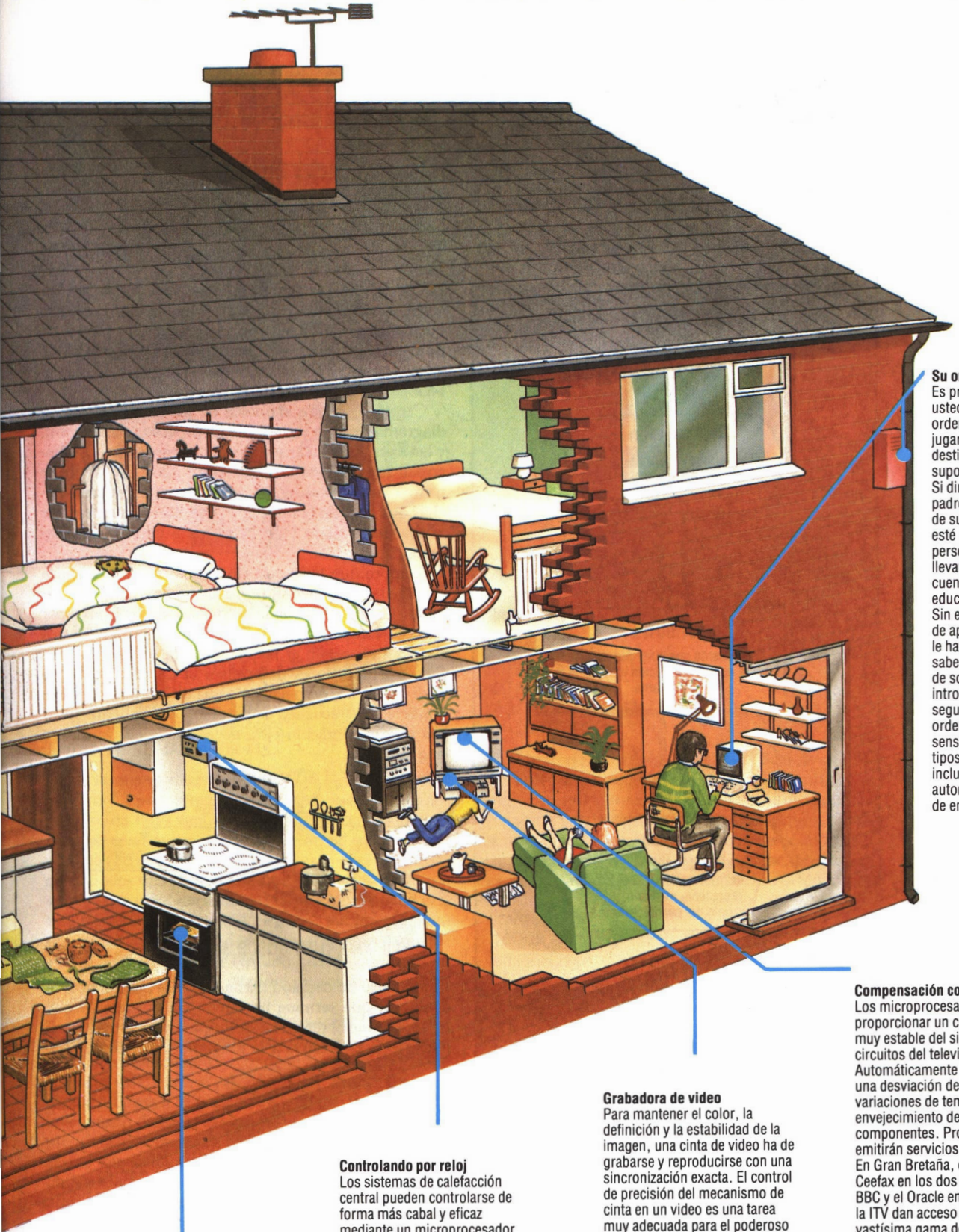


Juego para los niños
En el hogar, los niños suelen utilizar los ordenadores más que los adultos, familiarizándose con ellos con la misma naturalidad que si se tratara de un televisor. Conociendo la tortuga y el lenguaje LOGO, los niños aprenden a explorar y a satisfacer su curiosidad por sí mismos. Incluso un LOGO sencillo, que sólo posea gráficos tortuga, puede ayudar al niño más pequeño del hogar. También existe software educativo de gran calidad. Los juegos pueden estimular y educar, pero con frecuencia el interés y el talento de un niño se desarrollan y se refuerzan mucho más enfrentándolo a los problemas que supone programar realmente un ordenador. Actualmente el LOGO destinado a muchos ordenadores personales se está extendiendo y abaratando y ofrece un enorme potencial para favorecer la mejor forma de abordar la solución de un problema en muchos campos, no sólo en la programación de ordenadores

El chip que lava
Algunas lavadoras utilizan un microprocesador para seleccionar y supervisar los diversos ciclos de lavado y rotación que se requieren para tratar adecuadamente todos los tejidos susceptibles de lavarse a máquina. Se pueden visualizar y seleccionar al tacto las mejores combinaciones de acciones de lavado y temperaturas, niveles de agua, proceso de aclarado y velocidades de centrifugación. Puesto que no existen otros componentes móviles más que el tambor, el período de vida útil de la máquina es muy largo y los costos de mantenimiento se reducen considerablemente

El micro móvil
Los coches están empezando a incorporar ordenadores para mejorar el rendimiento y proporcionar nuevas prestaciones. Éstos pueden calcular el consumo de gasolina, controlar la velocidad, actuar como alarma antirrobo e incluso hablar con el conductor para advertirle que la presión del aceite es muy baja o que se está agotando la batería

El chip de coser
La máquina de coser tradicional puede realizar puntadas seguras y maravillosamente parejas, pero para ello se requiere paciencia y experiencia por parte del usuario. Una máquina de coser controlada por microprocesador puede ayudar a crear con muy poco esfuerzo un complicado patrón de bordado o una puntada delicada. Además de la selección de clases de puntadas preestablecidas, estas máquinas de coser se pueden programar para producir otras costuras, diferentes de las almacenadas en la memoria incorporada, aun cuando estén desconectadas



Su ordenador personal
Es probable que al principio usted haya utilizado su ordenador personal sólo para jugar; pero ¿qüé le parecería destinarlo a actividades que supongan un mayor desafío? Si dirige una empresa o si es un padre interesado en la educación de sus hijos, es probable que ya esté utilizando su ordenador personal de muchas maneras: llevando el estado de sus cuentas o aprovechando el valor educativo del ordenador.
Sin embargo, existen una serie de aplicaciones que tal vez no se le hayan ocurrido. Le interesará saber, por ejemplo, que el grado de sofisticación que puede introducir en un sistema de seguridad es ilimitado. El ordenador puede supervisar sensores ocultos de diversos tipos, disparar alarmas, e incluso telefonear automáticamente a los servicios de emergencia

Compensación completa
Los microprocesadores pueden proporcionar un control de color muy estable del sistema de circuitos del televisor. Automáticamente compensan una desviación de sintonía, las variaciones de temperatura y el envejecimiento de los componentes. Pronto se emitirán servicios de teletexto. En Gran Bretaña, el servicio Ceefax en los dos canales de la BBC y el Oracle en los canales de la ITV dan acceso a una vastísima gama de informaciones que se hallan almacenadas en una base de datos. Esta información se actualiza constantemente, de modo que en cualquier momento y de forma instantánea se pueden conocer las últimas novedades en cuanto a la actualidad, tiempo, deportes e incluso las cotizaciones de la bolsa

Grabadora de video
Para mantener el color, la definición y la estabilidad de la imagen, una cinta de video ha de grabarse y reproducirse con una sincronización exacta. El control de precisión del mecanismo de cinta en un video es una tarea muy adecuada para el poderoso microchip con el que cuentan algunos modelos. También puede encargarse de grabar programas de televisión cuando usted no esté en casa. Basta con programar el sintonizador automático con la hora y los canales que le interesen

Controlando por reloj
Los sistemas de calefacción central pueden controlarse de forma más cabal y eficaz mediante un microprocesador que a través de los métodos convencionales. El reloj electrónico del chip permite programar convenientemente las diversas necesidades de calefacción durante los días laborables y los fines de semana. Zonas separadas como los dormitorios o el garaje también pueden tener sus propios programas de control de tiempo y temperatura. Un control como éste ahorra energía y reduce las facturas

No más comidas quemadas
Un horno informatizado puede ayudarle a preparar platos perfectamente cocinados mediante el control exacto del tiempo y de la temperatura. Mientras el plato se cocina según la temperatura y el tiempo programados, la siguiente receta aparece en pantalla a través del adaptador para teletexto del ordenador personal

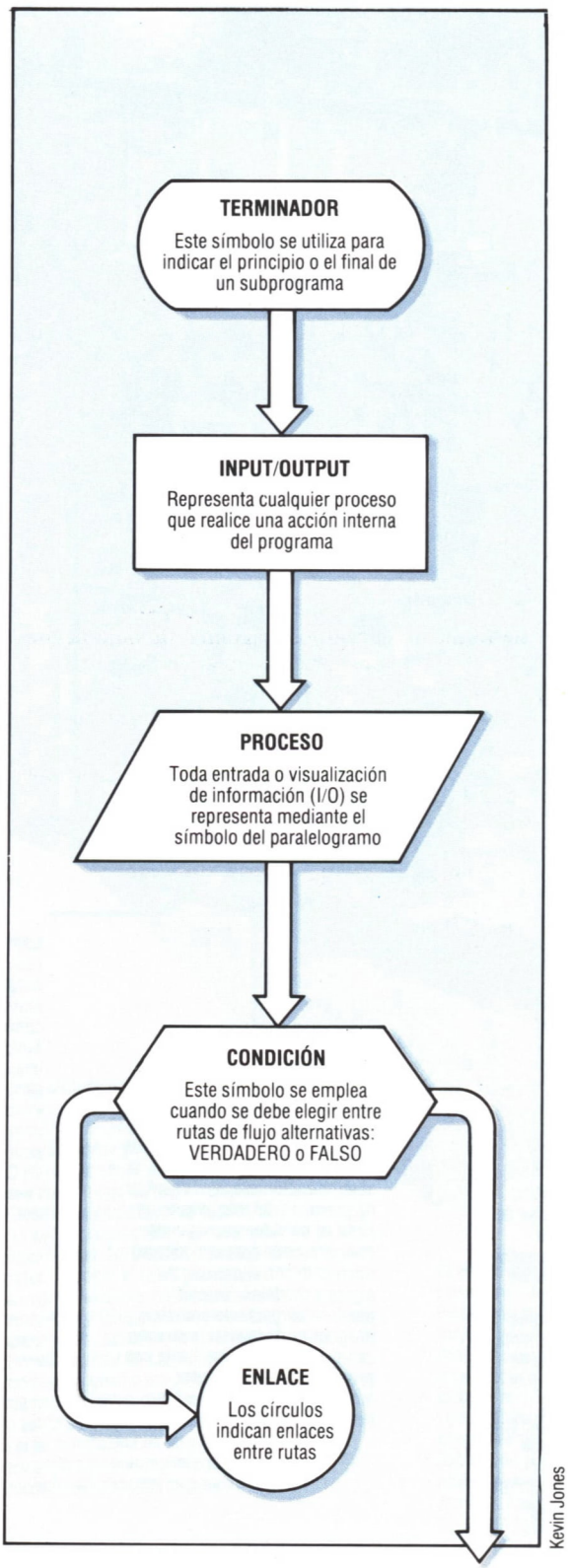
Diagramando la ruta

La cuidadosa utilización de diagramas de flujo lleva a crear programas eficaces y bien organizados

El flujo de información
El objetivo real de un diagrama de flujo es indicar de manera simple y concisa el flujo de información y el control a través de un programa de ordenador.

Los puntos de "condición" son decisivos; hacen que el control se transfiera a un punto diferente del que sigue en la secuencia. Una representación gráfica sencilla de esta transferencia de control es mucho más fácil de comprender que la misma sentencia escrita: realmente, ¡una imagen puede valer más que mil palabras!

El símbolo "CONDICIÓN" puede representarse ya sea mediante un hexágono aplanado, como en la ilustración, o bien mediante una forma de rombo alargado

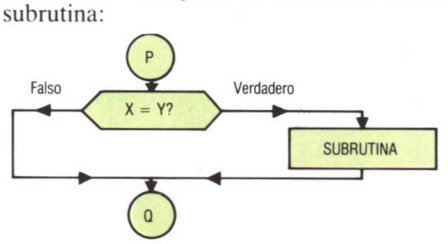


Kevin Jones

Un problema puede representarse de forma gráfica sencilla, dibujando diagramas que ilustren los pasos que requiere el procesamiento y los caminos o rutas de flujo que los conectan entre sí. Estos "diagramas de flujo" son útiles como medio para comprender un problema y para trabajar en su solución.

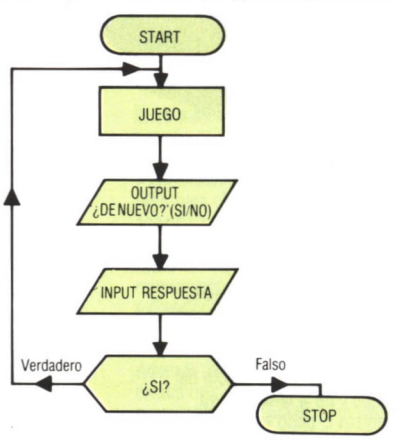
Cada uno de los símbolos de los recuadros en un diagrama de flujo representa un proceso o una acción, y las líneas que enlazan estos recuadros de acción describen los posibles caminos entre ellos. El "flujo de tráfico" es unidireccional, por lo cual se utilizan flechas que indican la dirección, que normalmente es de arriba hacia abajo y de izquierda a derecha del diagrama.

Siempre que haya que elegir una opción, se utiliza un "recuadro de decisión" en forma de rombo o hexágono. Como en el caso anterior, el control fluye por un solo camino, pero puede discurrir en una de dos direcciones, según el resultado de la condición. Si la finalidad de ésta es determinar si un proceso ha de realizarse o no, entonces sólo una de las rutas de salida contiene un "recuadro de proceso". El siguiente ejemplo es una condición para decidir si se deriva o no a una subrutina:




```
120 IF X=Y THEN GOSUB 300
```

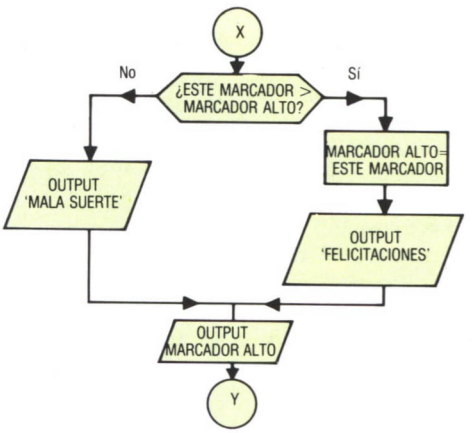
El recuadro de decisión también se usa para indicar la condición que termina un bucle. En el ejemplo que sigue, el control retorna al principio de un programa al dar respuesta positiva a la pregunta '¿DE NUEVO?':




```
90 REM**COMIENZA EL JUEGO**
100
800
```

```
810 PRINT "¿DE NUEVO? (SI/NO)";
820 INPUT R$
830 IF R$ = "Y" THEN GOTO 100
840 END
```

Es posible que deseemos adoptar una decisión que de por resultado seguir uno de dos cursos de acción claramente diferenciados. En el siguiente ejemplo comparamos el marcador de un jugador con otro marcador anterior más alto:




```
1200 IF ESTE MARCADOR > MARCADOR ALTO THEN
     GOTO 1230
1210 PRINT "MALA SUERTE. DEBE ABANDONAR";
1220 GOTO 1250
1230 LET MARCADOR ALTO = ESTE MARCADOR
1240 PRINT "¡FELICITACIONES! UN NUEVO RECORD";
1250 PRINT MARCADOR ALTO
```

Observe que el valor de MARCADOR ALTO se imprime en ambos casos, y que en el proceso los dos caminos posibles de flujo vuelven a unirse para convertirse en la única entrada para esta operación de salida.

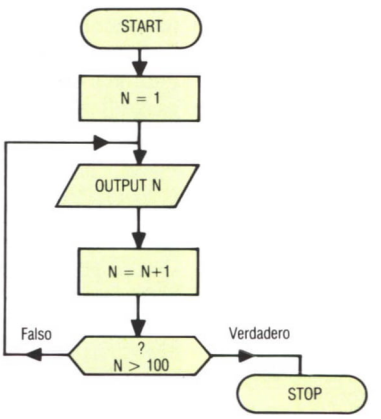
Todas las decisiones se toman como el resultado de condiciones similares a ésta, que dan un resultado positivo o negativo, verdadero o falso. Como puede ver, este proceso de toma de decisiones, puramente binario, niega la posibilidad de una respuesta "quizá". Puede utilizar cualquier término que desee, ¡pero no olvide designar los dos caminos de salida coherentemente!

Todos los lenguajes de programación poseen una sentencia de decisión intrínseca que, si se satisface la condición "verdadero", conduce a una bifurcación condicionada; pero, si el resultado es "falso", deja que el control pase a la sentencia siguiente. En el caso de una versión de BASIC que permita sólo un IF-THEN simple, debemos imitar la bifurcación condicionada mediante una sentencia GOTO, como en la línea 1200 del último ejemplo. La sentencia de la línea 1210 sólo se ejecutará si el resultado de la condición de la línea 1200 es falso.

¿Y qué sucede con la segunda utilización de GOTO en la línea 1220? Como puede observar, el empleo de GOTO al final de la condición, para solucionar el problema del destino de la bifurcación condicionada, nos ha obligado a usar este procedimiento para volver a "unir" los dos caminos de control posibles, en este caso en la línea 1250.

Normalmente la utilización de diagramas de flujo favorece la introducción de sentencias GOTO como recurso para seguir punto por punto la representación gráfica del programa. En general, esta forma de utilizar los saltos no condicionados resulta bastante peligrosa. Si la versión del BASIC que se está empleando obliga a esta solución, el diagrama de flujo constituye un método excelente para que estemos en condiciones de juzgar en qué forma el control sigue la sucesión normal del programa.

Tomemos un último ejemplo para estudiar cómo la utilización de un diagrama de flujo nos permite representar exactamente los pasos necesarios para realizar una tarea sencilla: imprimir todos los números comprendidos entre uno y cien.




```
10 LET N = 1
20 PRINT N
30 LET N = N+1
40 IF N > 100 THEN END
50 GOTO 20
```

Esta forma de utilizar los diagramas de flujo tiende a favorecer un enfoque paso a paso de la escritura del programa, lo cual con frecuencia da como resultado programas poco elegantes, particularmente en proyectos más largos. Para quienes poseen un conocimiento aunque sea superficial del lenguaje BASIC, obviamente es más conveniente emplear un bucle FOR-NEXT. Por ejemplo:

```
10 FOR N = 1 TO 100
20 PRINT N
30 NEXT N
40 END
```

En un diagrama de flujo es imposible representar este BASIC "taquigráfico" y, de seguirlo exactamente, nos llevaría a una forma menos eficaz de resolver el problema. No obstante, nos proporciona cierta información acerca de la estructura del bucle FOR-NEXT y, por lo tanto, será valioso, cuando examinemos ésta y otras funciones BASIC, para determinar cómo están construidas.

Los diagramas de flujo son particularmente útiles durante la etapa de planificación o conceptualización de la programación, especialmente en las partes "difíciles". Los programadores muy experimentados tienden a utilizarlos menos que los principiantes, y a menudo recurren a un diagrama de flujo para ilustrar y documentar un programa escrito sin su ayuda. Sin embargo, tanto si el diagrama de flujo se traza sobre papel como si sólo existe en el subconsciente del programador, la idea de diagramar el flujo de información y de control es esencial para el usuario de un ordenador como herramienta para resolver los problemas.

Micros en marcha

Basta de consultar mapas de ruta y de preocuparse del indicador de la gasolina: el coche del futuro lo llevará a su punto de destino de forma económica y segura

Cortesía de Toyota

Un dinámico tablero de instrumentos
Puede que el tablero de instrumentos de su próximo coche se parezca al que muestra la fotografía. Todos los diales y los indicadores móviles utilizados en los tableros de instrumentos convencionales han sido reemplazados por un terminal especialmente diseñado, controlado por ordenador. Los diodos luminosos pueden controlar la velocidad, el nivel de gasolina y la temperatura con mayor velocidad y exactitud que los instrumentos electromecánicos

La fabricación y el diseño de coches será uno de los campos donde veremos funcionando claramente la tecnología de los ordenadores. En la actualidad ya es posible adquirir un automóvil que detecte él mismo el momento en que necesite un servicio de mantenimiento y que avise cuándo es preciso visitar el taller.

El coche puede poseer esta facultad porque unos sensores acoplados en diversos puntos del motor alimentan a un microprocesador con información acerca del kilometraje y las temperaturas, permitiendo que evalúe las condiciones en que está siendo utilizado el vehículo. El conductor sabrá que hace falta una reparación porque el diminuto ordenador hace funcionar una serie de luces verdes, ámbar y rojas, situadas en el tablero de instrumentos. Cuando se apagan las luces verdes y se encienden las rojas, el conductor es advertido de que es necesario llevar el coche a un taller.

En Europa y en Japón se fabrican coches que le hablan al conductor, indicándole que se coloque el cinturón de seguridad o avisándole de la inminencia de un problema en el motor, como un recalentamiento o la disminución de los niveles de agua y aceite. El coche puede hablar porque posee un sintetizador de voz; se trata de un ordenador que ha sido programado digitalmente con las características de la voz humana. Esto se conoce como *sonido digital*, porque las ondas sonoras se transforman en números binarios, para que resulten comprensibles al ordenador.

Cuando los sensores del motor detectan una situación de alarma, el ordenador activa el sintetizador de voz que, a su vez, convierte la salida digital en el sonido de una voz humana y la emite por un altavoz.

Un viaje tranquilo

Otra de las formas en que los ordenadores pueden contribuir a mejorar los coches es a través del control de la suspensión. La Lotus está trabajando en un sistema denominado "suspensión activa". Esta técnica utiliza un ordenador para regular la rigidez y la flexibilidad de los amortiguadores de coche en fracciones de segundo; de esta manera el vehículo conserva su estabilidad en la carretera tanto si el conductor viaja solo como si lo hace con pasajeros y equipaje.

Los coches deportivos tienen, por lo general, una suspensión rígida que les permite afirmarse bien sobre la carretera. El inconveniente de este tipo de suspensión es que la marcha es muy dura y los pasajeros sienten todos los baches. Pero si un coche tiene una suspensión blanda que le proporcione una marcha "entre algodones", no tomará bien las curvas, porque la carrocería se balanceará demasiado. Con el control de la suspensión por ordenador se logrará aunar las ventajas de ambos sistemas.

Las firmas japonesas Honda y Toyota también están desarrollando ordenadores para navegación. Estos ordenadores le señalan al piloto el rumbo a tomar. Y lo hacen midiendo la velocidad, la dirección y la distancia recorrida, al tiempo que comparan estos datos con un mapa de la ruta correcta que llevan en su memoria. El conductor puede entonces decidir si gira hacia la izquierda o hacia la derecha o si sigue adelante, de acuerdo a una serie de indicadores en el tablero de instrumentos.

Uno de los ordenadores más comunes en los coches mide el consumo de gasolina y puede calcular la hora de llegada. El conductor sabe, en un momento dado, cuánta gasolina está consumiendo, mientras el ordenador le indica cuál ha sido la velocidad promedio del viaje. Uno de los ordenadores para automóviles más avanzados permite que el conductor programe una velocidad de crucero. El coche, entonces, mantendrá esta velocidad sin que para ello el conductor necesite tocar el acelerador.

Los ordenadores que se instalan en *juggernauts* de larga distancia tienen una finalidad muy diferente. Sirven como cuaderno de bitácora electrónico y permiten que las autoridades de tráfico puedan determinar durante cuánto tiempo ha permanecido al volante el conductor y a qué velocidad, así como la distancia recorrida.

Una de las ventajas más importantes que ofrecen los ordenadores para coches es el obtener un mejor aprovechamiento del combustible. La firma alemana BMW ya posee una gama de automóviles con un sistema que determina la mezcla óptima de gasolina y aire requerida en un momento dado de la conducción.

En efecto, se trata de "sintonizar" al coche muchísimas veces por segundo para racionalizar todo lo posible el consumo de combustible. El sistema funciona midiendo continuamente la mezcla de aire y gasolina y realizando ajustes para tener presente la velocidad del coche, en qué marcha está trabajando y la temperatura del motor.

El futuro
¿Qué nos depara el futuro respecto a los ordenadores para automóvil? Teóricamente, sería posible que el ordenador se hiciera cargo por completo del acto de la conducción en sí mismo. Todo cuanto el conductor debería hacer sería programar el ordenador del coche con el punto de destino. El ordenador guiaría automáticamente el coche valiéndose de la información proveniente de sensores instalados en la carretera o comunicándose con los ordenadores centrales de tráfico. Otro desarrollo que podremos ver es el radar por ordenador, que reajustaría automáticamente la velocidad del vehículo si éste se acercara demasiado al coche que lo precediera.
Pronto los instrumentos de dial serán reemplazados por un tablero de instrumentos con una pantalla similar a la de un monitor en la cual se visualizarán los gráficos trazados por el ordenador. El conductor podrá solicitar la visualización electrónica que desee, como la temperatura del motor o el nivel de gasolina. La información relativa a la conducción podrá proyectarse sobre el parabrisas, de modo que el usuario no necesitará apartar sus ojos de la carretera. También es posible que los coches vengan equipados con ordenadores que informen de inmediato al mecánico acerca de cualquier tipo de problema. Se podría conectar el ordenador del coche al ordenador preprogramado del taller, que realizaría una revisión general. La fotografía muestra el prototipo de la Honda de un sistema de navegación por ordenador, montado en el tablero de instrumentos

Ian McKinnell

Carreras de informática

El profesional de la informática adquiere su experiencia trabajando primero como técnico y ascendiendo poco a poco de categoría

Cortesía de ICL

Gigantes dóciles
Los grandes ordenadores comerciales como el que muestra la fotografía (conocidos como "ordenadores de unidad principal", para diferenciarlos de los mini y microordenadores) requieren de un equipo de operadores altamente experimentados para mantenerse en un nivel de funcionamiento al máximo de sus posibilidades. Las máquinas de estas dimensiones pueden ejecutar cientos de programas simultáneamente y servir a miles de usuarios en cualquier lugar del mundo mediante las líneas telefónicas, los enlaces por microondas y los satélites de comunicaciones

La creciente utilización de los ordenadores en el hogar y en las escuelas está dando lugar a la aparición de muchos programadores geniales, personas que, en otras circunstancias, jamás habrían pensado en la posibilidad de seguir una carrera en el campo de la informática. Pero la cruda realidad es que, como siempre, un aprendizaje superficial es algo peligroso, especialmente si ese conocimiento superficial se limita al lenguaje BASIC. Es importante llegar a comprender que las exigencias de un programador profesional son fundamentalmente diferentes de las de un usuario que programe su ordenador personal, y que muchas de las cualidades no son transferibles.

La elección lógica para una persona que haya completado sus estudios secundarios y tenga un gran interés por los ordenadores sería seguir un curso de informática a nivel universitario o de educación superior, o bien matricularse en una carrera universitaria relacionada con esta nueva disciplina. Muchas universidades y facultades autónomas ofrecen cursos de informática con rango académico, y los estudiantes que hayan obtenido un buen rendimiento al acabar los mismos tienen posibilidades de optar entre diversas ofertas de trabajo. El nivel de desempleo de la industria de la informática se ha limitado al personal informático de nivel inferior, básicamente programadores y operadores, mientras que la demanda de ingenieros, analistas de sistemas y diseñadores no ha disminuido.

Una de las opciones que tiene un campo laboral cada vez más amplio es el de la enseñanza de informática en las escuelas. Hasta ahora, la informática como materia de estudio sólo se había impartido en las universidades y escuelas superiores. En el campo de la educación, el personal capacitado escasea, y una carrera así será muy bien remunerada en el futuro.

En la industria de la informática pueden delimitarse seis niveles principales. El inferior puede describirse como el nivel del "usuario experimentado". Esta categoría incluye a los trabajadores que han aprendido a manejar ordenadores para realizar tareas determinadas, como tratamiento de textos o teneduría de libros. A menudo esta capacitación se interpreta como una experiencia complementaria dentro de otras ocupaciones (por ejemplo, de una secretaria o de un administrativo), pero también comprende funciones de la industria de los ordenadores como son operador de terminales, operador de perforadora de fichas, etc. Para estos empleos se requiere una serie de cualificaciones básicas, bachillerato o escuela superior, y la capacidad de pensar lúcidamente. Experiencias como el dominio del funcionamiento del teclado y otras similares, por lo general se adquieren en el propio trabajo.

El siguiente nivel es el de operador de ordenadores. Aunque los ordenadores que se utilizan en la industria difieren bastante de los ordenadores personales tanto en su aspecto como en su uso, se basan en los mismos principios; por lo tanto, siempre es útil estar ya familiarizados con ellos. Los operadores comprenden muy pronto cómo trabajan los ordenadores y, por ello, ser operador es un buen trampolín para llegar a programador. Considere, no obstante, que esta actividad puede exigirle un intenso esfuerzo físico. Por ejemplo, la mayoría de las grandes instalaciones funcionan 168 horas por semana y durante todo ese tiempo han de contar con un personal que las atienda.

Las principales aptitudes que se requieren para convertirse en programador son poseer una mente lúcida y metódica y capacidad de concentración. Para llegar a ser un programador cualificado se requiere un tipo de aptitud muy especial y, si bien los requisitos normales para la admisión son un título académico o el bachillerato, por lo general es más importante la capacidad para trabajar lógicamente. Los programadores ingresan en la industria sin cualificaciones formales y esta amplitud de miras atrae a muchos padres, esperanzados en las aptitudes de sus hijos para la programación.

Una elección para toda la vida

Como cualquier otra dependencia de una empresa moderna, el departamento de informática se organiza según unas líneas jerárquicas. A la cabeza del departamento está el **director de proceso de datos,** responsable de las muchas y diversas tareas que han de desarrollarse bajo la denominación general de procesamiento de la información.

Todos los profesionales de la informática se emplean primero como técnicos y obtienen su capacitación a medida que van ascendiendo de categoría. Las tres áreas principales de especialización son: funcionamiento del ordenador, programación y análisis de sistemas; a medida que se avanza en el camino de la promoción, se va adquiriendo un conocimiento básico de las tres áreas de especialización.

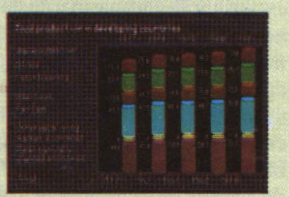
Tal como ocurre con otras profesiones, conviene entrar en el campo de la informática con la mayor cualificación posible. Si bien al comienzo parecerá no haber mucha diferencia entre los diferentes grados de cualificación, la persona menos preparada se encontrará pronto con escollos insalvables. ¡Es mucho más difícil concluir una carrera universitaria si uno trabaja durante toda la jornada!

Analistas

Antes de comenzar cualquier trabajo, se han de estudiar concienzudamente los objetivos y los recursos disponibles. El **analista de sistemas** tiene la función de entrevistar a los usuarios para determinar sus necesidades, conjuga estas necesidades con los recursos disponibles y sugerir un procedimiento para resolver el problema. Para que el analista sea capaz de desarrollar un sistema que funcione para otras personas, debe ser un pensador lógico, con buenas aptitudes para la comunicación y una chispa de creatividad. Con frecuencia es el agente de ventas del departamento de proceso de datos, por lo que debe causar siempre una impresión favorable entre sus "clientes": los usuarios de ordenadores de la empresa.

Programadores

El **programador** toma la amplia estrategia elaborada por el analista y la transforma, primero en un plan táctico, dividiendo el trabajo en segmentos más manejables, y luego en un código que el ordenador pueda reconocer e interpretar. Los **programadores de aplicaciones** se encargan de escribir programas para realizar tareas específicas, mientras que los **programadores de sistemas** están más relacionados con el rendimiento global del sistema de proceso de datos. Los programadores de aplicaciones tienden a trabajar de forma individual, incluso aunque estén integrados dentro de un equipo de proyectos. Para ellos es realmente importante su capacidad para concentrar la atención en la labor que se esté llevando a cabo. Los programadores de sistemas también necesitan de esta capacidad de concentración, pero, además, una actitud reposada. "Si usted puede mantener la cabeza fría cuando a su alrededor todos los demás la están perdiendo...", quizá entonces posea las condiciones requeridas para ser un programador de sistemas.

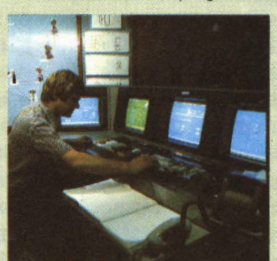


A los **operadores** de instalaciones más reducidas a veces se les solicita que ayuden a los programadores y a los ingenieros para diagnosticar fallos, además de ejecutar uno a uno los programas. Sin embargo, lo más importante es poseer un conocimiento profundo del método operativo del programa. Los programas de "amabilidad hacia el usuario" hacen que el trabajo del operador resulte más sencillo, y un programa bien "preparado" puede ser ejecutado por operadores con relativamente poca experiencia sin pérdida de eficacia.

Kai Choi

Operadores

Una de las especializaciones de esta industria que más esfuerzo físico exigen es la de operador de un gran ordenador de multiprogramación-multiusuario. Además de tener que caminar kilómetros en cada turno transportando discos, cintas o cajas de papel, el operador debe estar interiorizado con el sistema operativo del ordenador y con la importancia inherente a cada uno de los trabajos que se ejecuten en la máquina en cada momento.



A un **operador senior** se le puede solicitar que tome decisiones que afecten el trabajo de muchas otras partes de la gestión de la empresa, permitiendo o negando el acceso al sistema de ordenador.

Ingenieros de desarrollo de sistemas

Aunque tal vez en el futuro sean los propios ordenadores los que desarrollen la nueva generación de máquinas, en la actualidad este proceso de innovación se produce en el cerebro del **ingeniero de desarrollo de sistemas.** Éste desarrolla una labor en parte científica y en parte técnica: consiste en aprovechar los nuevos descubrimientos y desarrollos teóricos para mejorar y ampliar el rendimiento de un componente determinado del equipo. En este campo hasta el profesional menos cualificado ha estudiado cinco años o más en la universidad.

Tony Sleep

Especialistas técnicos

A menudo la única oportunidad de descansar que se le presenta a un operador ocurre cuando el ordenador sufre una avería: entonces se llama a un **especialista técnico** para repararlo. Gracias a la capacidad de los ordenadores modernos para detectar sus propias averías, y a la adopción casi universal de la construcción modular, el trabajo del técnico se ha simplificado considerablemente, si bien éste aún debe ser competente en cuanto a electrónica digital. También ha de ser un mecánico experimentado, capaz de trabajar con tolerancias aún más reducidas que las de un relojero medio. Para acceder a este campo se suele exigir una cualificación de nivel académico.

El trabajo que requiere menos esfuerzo intelectual es el de **operador de entrada de datos** (perforista-grabador). En este caso, las aptitudes que se requieren son básicamente las mismas que para un mecanógrafo: velocidad y precisión. Lo negativo de este trabajo es que puede llegar a ser repetitivo y tedioso, pero en muchas instalaciones pequeñas esta desventaja se compensa con la oportunidad que ofrece de llegar a familiarizarse con otros aspectos de la actividad del departamento de informática.

Como hemos mencionado anteriormente, comprender el BASIC no implica necesariamente tener abiertas las puertas de la industria informática. A pesar de tratarse de un lenguaje muy popular, la mayoría de los profesionales lo consideran mal estructurado y piensan que favorece hábitos de programación negativos y una forma de pensar muy desordenada. Este problema existe realmente, puesto que la mayoría de los niños con experiencia en ordenadores personales han aprendido BASIC en lugar de otros lenguajes mejor estructurados, como el LOGO o el COMAL.

Las universidades y escuelas universitarias que imparten cursos de informática manifiestan preferencia por la admisión de alumnos *que no hayan aprendido* BASIC, ya que consideran que este lenguaje favorece la adquisición de hábitos muy difíciles de eliminar.

A pesar de es: problema, bastantes jóvenes han hallado la manera de ser retribuidos económicamente por sus aptitudes para la programación en BASIC. Muchos de ellos escriben juegos en este lenguaje que les agradan a los otros jóvenes, y las empresas de software están ansiosas por hacerse con juegos dirigidos de forma tan directa a la mentalidad de los adolescentes. Algunos de los "niños precoces" que suelen aparecer en los periódicos porque, a pesar de su corta edad, ganan enormes sumas de dinero, sólo escriben en BASIC y no tienen un conocimiento profundo de la informática. Otros son auténticos fenómenos que escriben en lenguaje Assembly (el lenguaje de bajo nivel que controla con suma eficacia el código de lenguaje máquina de un microprocesador) y tienen por delante un brillantísimo porvenir. Raramente los periodistas están capacitados para diferenciar estas dos clases de jóvenes, y este tipo de reportajes puede inducir a los padres a creer que su pequeño hijo, obsesionado por el ordenador, está preparado para comenzar a ganar dinero a raudales. Esto, aunque posible, es poco probable.

En el negocio

En la industria informática propiamente dicha, los programadores se dividen en dos grupos: los programadores de aplicaciones y los programadores de sistemas. Los primeros escriben programas para realizar una tarea específica. Los programadores de sistemas son "amas de llaves": escriben programas para mantener a punto el sistema de ordenadores, por ejemplo, para detectar averías. El programador de aplicaciones suele encontrarse con otras personas fuera de la sala de ordenadores (clientes) y por lo general trabaja integrado en un equipo desarrollando programas para una función específica. Los programadores de sistemas están más especializados y suelen trabajar en solitario. Ellos hablan directamente con la "inteligencia" de la máquina.

Pero, en este punto, la industria de la informática traza una línea divisoria imaginaria, a partir de la cual sólo permite el acceso a los programadores más brillantes y a los graduados universitarios más cualificados. Éste es el reino reservado a los analistas de sistemas y a los diseñadores.

Los analistas de sistemas consideran un problema y luego deciden cómo puede un ordenador ayudar a resolverlo. Por ejemplo: una compañía petrolera descubre un nuevo yacimiento debajo del lecho marino. Ellos han medido las dimensiones del yacimiento y han comprobado que la calidad del petróleo difiere ampliamente. La compañía petrolera debe decidir si le conviene o no invertir los miles de millones de dólares necesarios para explotar el yacimiento. Esta decisión se tomará de acuerdo con las perspectivas del mercado internacional del petróleo durante el período de vida del yacimiento (supongamos, 20 años), y la compañía debe determinar qué parte del yacimiento ha de perforar primero. Dado que la inversión es tan grande, la compañía petrolera confía el problema a su personal de informática para que lo analice. El analista considera el problema, consulta con economistas, con expertos en marketing de crudos, con geólogos y otros especialistas, y construye un "modelo" por ordenador del yacimiento petrolífero durante los próximos años.

Entonces los ejecutivos de la compañía petrolera juegan con este modelo a "¿qué sucedería si...?", descubriendo cómo incidirían en el rendimiento global diversas decisiones acerca de precio, técnicas de refino y aproximaciones de mercado, recibiendo toda la información que necesitan para tomar las decisiones finales.

En la industria informática existen, además, diversos roles importantes, si bien pocos de ellos se tienen en tan alta estima como el del analista de sistemas. Una excepción quizá sea la del diseñador de hardware. La demanda de ingenieros electrónicos existe a todos los niveles, desde los centros de reparaciones y mantenimiento hasta los departamentos de investigación; pero las áreas de desarrollo del producto de investigación son coto de quienes poseen las más altas cualificaciones en ingeniería electrónica.

En líneas generales existe una notable escasez de personal cualificado en informática. Pero, al mismo tiempo, las cotas de desempleo son igualmente altas, afectando incluso a graduados universitarios y politécnicos. Este evidente desajuste es fuente de preocupación tanto para los responsables de la preparación de personal como para los industriales, y es de esperar que se adopten serias medidas para modificar la situación, incluyendo programas de reciclaje para personas cualificadas en otros campos, y una variedad de oportunidades mucho más amplia para aprender a nivel primario, secundario y terciario.

Diversos gobiernos, particularmente el de Gran Bretaña, consideran que la microelectrónica puede ser la respuesta para algunos de los problemas que plantea el desempleo a corto plazo. El Youth Training Scheme, que pretende proporcionar un entrenamiento y una experiencia laboral a los jóvenes que, habiendo finalizado sus estudios secundarios, todavía no han encontrado su primer trabajo, ofrece actualmente 4 500 plazas en los Information Technology Centres de Gran Bretaña. En estos centros, los jóvenes aprenden diversos aspectos de la microinformática mientras perciben una subvención de capacitación equivalente al subsidio de desempleo. Otros proyectos incluidos en el mismo plan ofrecen la familiarización con el ordenador a aquellos que no llegaron a conseguirla en la escuela (ya sea porque acabaron sus estudios antes de que se iniciara "la era del ordenador", o bien porque no fueron "seleccionados" para utilizarlo), aumentando también sus posibilidades de encontrar un empleo, porque para quien ha dejado el colegio sin haber conocido un ordenador, o para quien es analfabeto informáticamente hablando, las perspectivas de hallar un puesto de trabajo pueden presentarse dudosas.

Cortesía de "The Sunday Times"

David Simmonds
David Simmons, de 17 años, el año pasado ganó 10 000 libras esterlinas durante las vacaciones de verano.
Es un fenómeno de la programación y escribe programas para la Commodore (que fabrica los ordenadores PET y Vic). A diferencia de muchos adolescentes, David escribe software "serio" para aplicaciones comerciales y, para cuando acabe sus estudios, espera encontrar un puesto lucrativo dentro de la industria informática.
David empezó jugando con el ordenador que su padre trajo a casa desde el trabajo, pero muy pronto abandonó los juegos y comenzó a descubrir cómo programar. Inicialmente David publicó algunos de sus programas en la revista que la Commodore edita para sus usuarios, y poco a poco comenzó a vender copias de sus programas.
Finalmente la Commodore se enteró de ello y David logró que le dieran la oportunidad de mostrar lo que era capaz de hacer. El resultado fue su primera realización seria en programación

Eugene Evans
Eugene Evans tiene 17 años y percibe unas 40 000 libras al año. Es uno de los pequeños genios que están empezando a aparecer en el campo de la programación de ordenadores y está ayudando a mantener a quien le da empleo, la Imagine Software de Liverpool, entre los primeros fabricantes de juegos para ordenador de Gran Bretaña. Los elevados ingresos de estos fenómenos de la programación generalmente se traducen en concepto de royalties sobre la venta de los juegos (similares a los que perciben los escritores por sus libros), y los adolescentes son los más indicados para desarrollar juegos que agraden a otros adolescentes, principal mercado de los juegos por ordenador

Términos clave

Diálogo digital

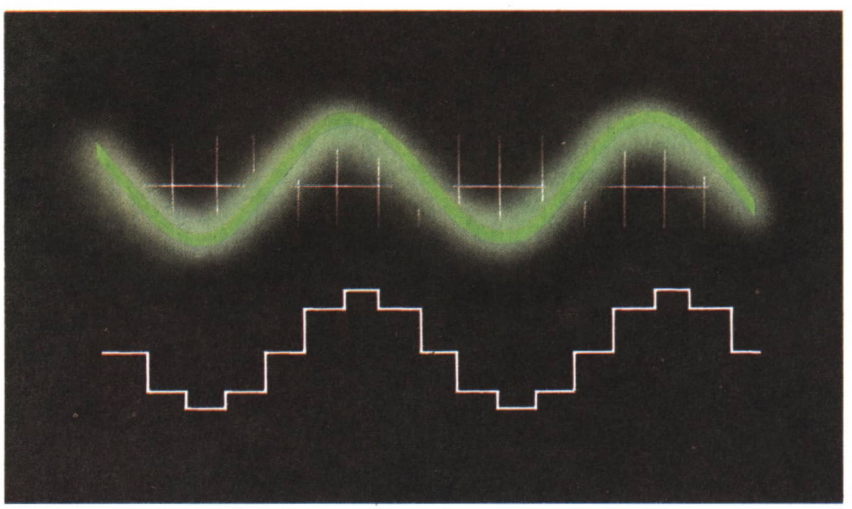
Input/Output, o Entrada/Salida: esenciales para el funcionamiento de cualquier sistema de ordenadores

De analógico a digital
En el mundo real, es muy poca la información que llega con pasos digitales, discretos. Por el contrario, la mayoría de ella es tan variable como los niveles de ruido o las mareas.
Con el fin de que esta información le resulte comprensible al ordenador, la señal ha de ser primeramente digitalizada. El convertidor Analogue-to-Digital, A/D (de analógico a digital), toma muestras de la fuente de señales a una velocidad constante y conocida, tal vez un centenar por segundo. Cada una de estas muestras se almacena en una dirección separada de memoria en forma de valor digital, permitiendo, de este modo, que se realicen los cálculos de variación y se reconozcan las condiciones fuera del límite.
Los convertidores Digital-to-Analogue, D/A (de digital a analógico), realizan una función similar pero a la inversa y para suavizar los picos y obtener una curva regular se utilizan técnicas estadísticas

Input/Output (Entrada/Salida) o, como se abrevia comúnmente, I/O, es el término que se emplea para describir la transferencia de información entre la CPU (Unidad Central de Proceso, que constituye el corazón del ordenador) y el "mundo exterior". "Mundo exterior" equivale, en este contexto, a todos los dispositivos que pueden conectarse al ordenador. No incluye ni la memoria RAM ni la memoria ROM, que se consideran como integradas en el ordenador. La distinción entre lo que sucede en el "interior" del ordenador y lo que ocurre en el "exterior" es un tanto arbitraria. Pero todos los circuitos lógicos diseñados para trabajar en íntima unión con la CPU y la memoria principal, se consideran como pertenecientes al "interior" del ordenador.

Los dispositivos externos, que utilizan I/O para comunicarse con el ordenador, incluyen una gran variedad de periféricos, desde el teclado a las unidades de disco flexible, pasando por las palancas de mando, las impresoras y las unidades de visualización de video.

Cuando la CPU desea recuperar información de su memoria, primero debe "localizar" la dirección en donde se halla almacenado el byte de información. Del mismo modo, si la CPU desea almacenar un byte de información para utilizarlo posteriormente, debe primero localizar la dirección en donde se ha de almacenar el byte. Este proceso se denomina *direccionamiento de memoria*. Implica la colocación, por parte de la CPU, de los dígitos binarios correspondientes a la dirección de memoria deseada en una serie de 16 cables conectados a las "patillas de dirección" de la CPU. Estos cables se denominan *bus de direcciones*. Un sistema de circuitos especial en la sección de memoria decodifica estos 16 dígitos binarios para seleccionar la dirección de memoria correcta. (Dieciséis dígitos binarios pueden dar 65 536 combinaciones exclusivas de unos y ceros y, por tanto, localizar la misma cantidad de direcciones de memoria diferentes.)



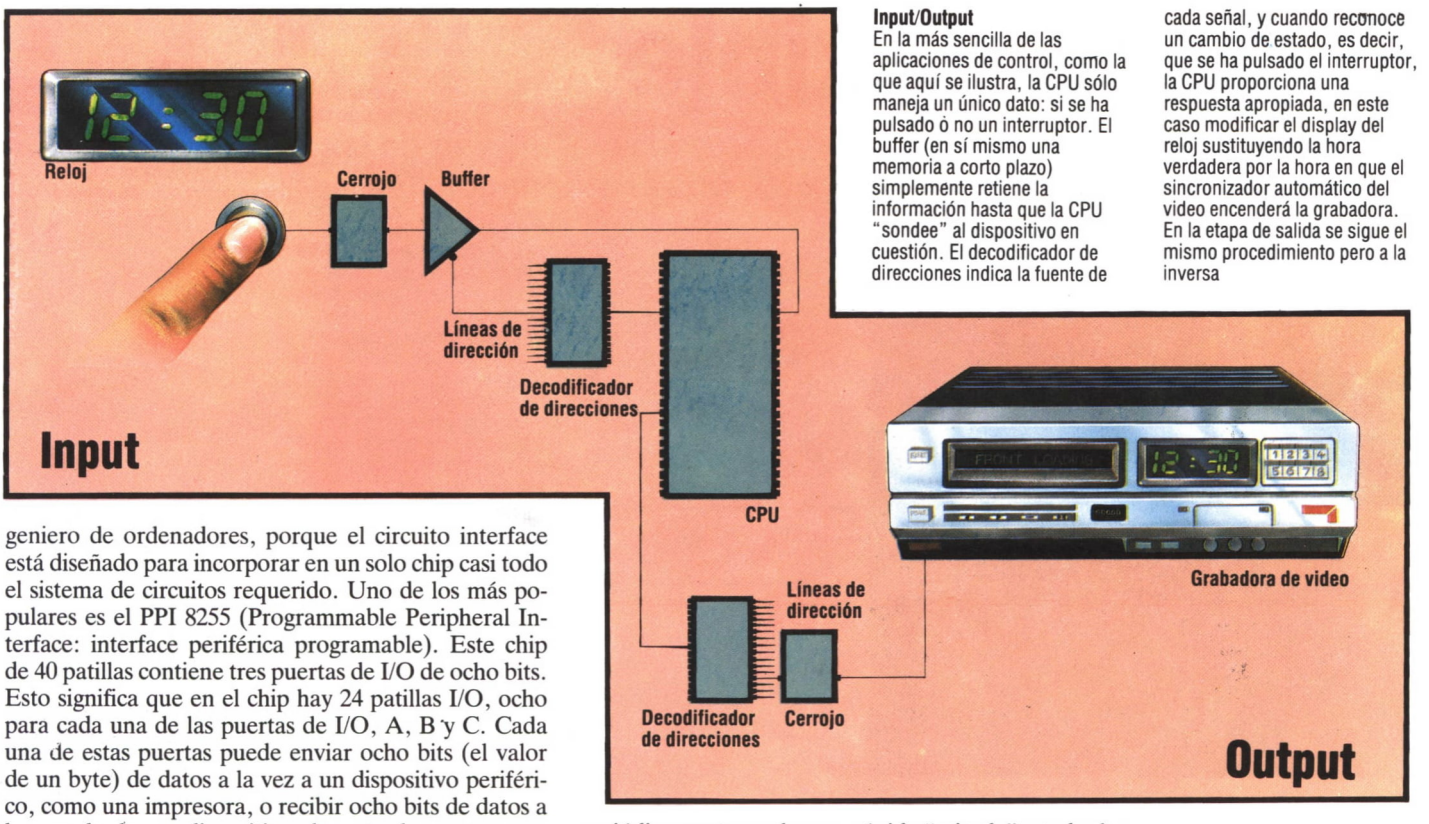
Mark Watkinson

Si el ordenador desea comunicarse con un dispositivo externo, debe localizar ese dispositivo de modo similar. Sólo dispone de ocho líneas de dirección. Esto limita a 256 el número total de direcciones de I/O separadas que pueden seleccionarse. Es una cantidad pequeña comparada con el potencial de direccionamiento de 16 líneas de dirección, pero en la práctica 256 resulta más que suficiente. Normalmente no se plantea la necesidad de conectar a un ordenador una cantidad mayor de unidades externas.

Seleccionando los dispositivos

Para averiguar cómo actúa realmente el ordenador para seleccionar una unidad externa y enviarle información, consideremos uno de los dispositivos de salida más sencillos: un LED (Light Emitting Diode: diodo emisor de luz) montado en el teclado del ordenador para indicar cuándo se ha pulsado la tecla CAPS LOCK (el microordenador BBC posee una tecla y un LED de este tipo). Para el ordenador, el LED es tan sólo otro dispositivo externo al cual puede enviarle información. En el caso de un único LED, la información será un único 1 (para encenderlo) o bien un único 0 (para apagarlo). Aun tratándose de un LED modesto que requiera una única información, necesita una dirección o localización. La CPU no puede ocupar todo su tiempo direccionando un LED. Necesita poder seleccionarlo una sola vez para decirle cuándo ha de encenderse, y otra vez para comunicarle cuándo ha de apagarse. Supongamos, sólo para seguir con el mismo razonamiento, que la dirección I/O del LED es 32. Para seleccionarla, las líneas de dirección serán, para la CPU, el equivalente binario de 32. Esto es, 00100000 en binario. El LED tendrá un circuito "decodificador" especial que ignorará todas las otras combinaciones de bits en las líneas de dirección. Cuando la línea de dirección sea 00100000, este circuito decodificador la reconocerá y producirá un voltaje alto y, por lo tanto, una salida "verdadera". Lo siguiente que se requiere en el circuito para hacer que el LED se encienda, es un pequeño chip llamado *cerrojo de datos*. Este chip cierra o retiene los datos que se le envían, de modo que el LED permanece encendido o apagado hasta la próxima vez que el chip es direccionado y se le envía una nueva información. Este proceso recibe el nombre de *toggling*.

La mayoría de los dispositivos externos con los que se comunica el ordenador son bastante más complicados que un LED. La impresora es un periférico típico y cada vez que el ordenador se comunica con ella los datos transmitidos representan el código para que se imprima un carácter. Normalmente, cuando han de transferirse grandes cantidades de información, como en el caso de una impresora, se utiliza un chip interface I/O especial. Estos chips simplifican la labor del in-



Input/Output
En la más sencilla de las aplicaciones de control, como la que aquí se ilustra, la CPU sólo maneja un único dato: si se ha pulsado o no un interruptor. El buffer (en sí mismo una memoria a corto plazo) simplemente retiene la información hasta que la CPU "sondee" al dispositivo en cuestión. El decodificador de direcciones indica la fuente de cada señal, y cuando reconoce un cambio de estado, es decir, que se ha pulsado el interruptor, la CPU proporciona una respuesta apropiada, en este caso modificar el display del reloj sustituyendo la hora verdadera por la hora en que el sincronizador automático del video encenderá la grabadora. En la etapa de salida se sigue el mismo procedimiento pero a la inversa

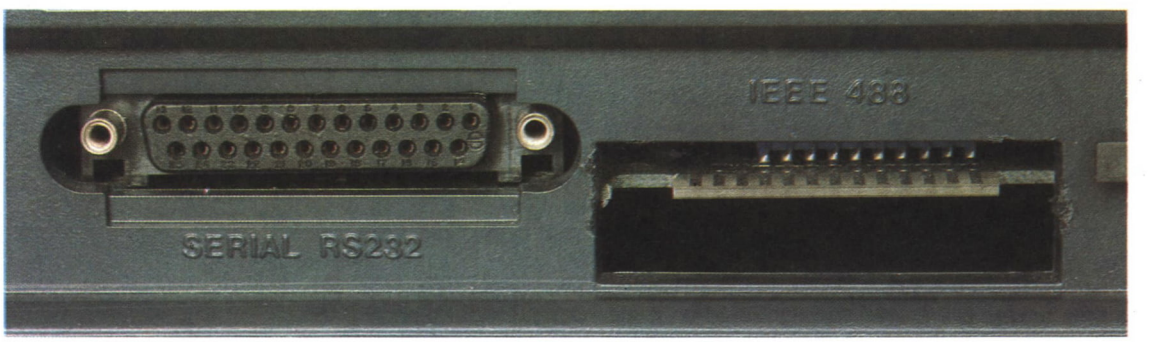
geniero de ordenadores, porque el circuito interface está diseñado para incorporar en un solo chip casi todo el sistema de circuitos requerido. Uno de los más populares es el PPI 8255 (Programmable Peripheral Interface: interface periférica programable). Este chip de 40 patillas contiene tres puertas de I/O de ocho bits. Esto significa que en el chip hay 24 patillas I/O, ocho para cada una de las puertas de I/O, A, B y C. Cada una de estas puertas puede enviar ocho bits (el valor de un byte) de datos a la vez a un dispositivo periférico, como una impresora, o recibir ocho bits de datos a la vez desde un dispositivo de entrada, como, por ejemplo, un teclado.

Para enviar ocho bits de datos a una impresora, la CPU se dirige primeramente al PPI y luego le envía ocho bits de información en el bus de datos. Esta información se almacena provisionalmente en la celda de memoria de un byte situada en el chip, que se conoce como registro. El PPI, entonces, hará que esta información quede disponible en el juego apropiado de patillas I/O. Un principio similar pero que funciona a la inversa permite almacenar los datos provenientes de dispositivos de entrada externos en un registro del chip, y luego colocarlos en el bus de datos cuando la CPU le envía la señal adecuada. Como hemos mencionado previamente, no se puede permitir que los dispositivos externos coloquen su información continuamente en el bus de datos del ordenador; se necesita transferir información desde y hacia la memoria y por otros dispositivos de I/O. El chip de I/O almacena temporalmente la información y sólo la coloca en el bus de datos (para que sea recogida por la CPU) cuando la CPU le indica que lo haga.

¿Cómo sabe la CPU que un dispositivo externo está intentando enviarle información al ordenador? Brevemente, existen dos procedimientos principales. La CPU deja de ejecutar el programa que está llevando periódicamente y echa una rápida "mirada" a todas las puertas de entrada. Si allí descubre que hay una información que está esperando entrada, instruye a la puerta en el sentido de que coloque la información en el bus de datos. El proceso de investigación de los dispositivos de entrada se denomina *sondeo*.

El otro procedimiento se basa en "interrupciones". El dispositivo a la espera de atención envía una señal de interrupción directamente a la CPU, que obliga a detener el programa que se está ejecutando mientras se atiende a la puerta de entrada. Más adelante nos ocuparemos con mayor detalle de las ventajas y los inconvenientes que ambos procedimientos ofrecen al usuario.

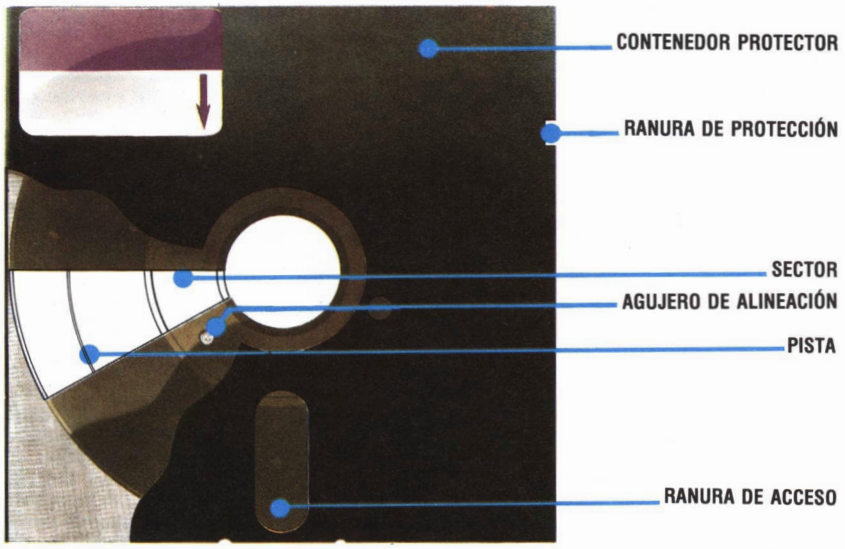
La I/O que hemos descrito hasta ahora se denomina *I/O en paralelo*, porque la información sale o entra a razón de un byte a la vez utilizando ocho cables o líneas de I/O (ocho bits en paralelo). Existe otra técnica que recibe el nombre de *I/O en serie*. En este caso la información de cada byte se alimenta a razón de un bit a la vez, uno detrás de otro. Algunas impresoras emplean interfaces en serie, y la salida de los modems también se realiza en serie. La principal ventaja radica en que, básicamente, la comunicación en serie permite usar un solo par de cables en lugar de ocho o más que requiere la *I/O en paralelo*.



Puertas en serie y puertas en paralelo
La mayoría de los microordenadores modernos poseen puertas tanto en serie como en paralelo; las primeras pasan los datos de a un bit a la vez, mientras que las segundas lo hacen en forma de bytes. El tipo de convención en serie más común, conocido como RS232C, puede utilizar una conexión subminiatura de "tipo D", como el ejemplo de 25 patillas que muestra la fotografía (izquierda), o bien, aunque más raramente, un enchufe DIN como el empleado en los sistemas de alta fidelidad. La puerta en paralelo (derecha) responde a la convención IEEE488, desarrollada por la Hewlett Packard y que el Institute of Electrical and Electronics Engineers de Estados Unidos ha adoptado como estándar para toda la industria

El disco flexible

Los discos magnéticos giran a gran velocidad en el interior de las unidades de disco, llevando información que el ordenador "lee"



David Weeks

El floppy o diskette
La superficie de un disco flexible está dividida en una cantidad de bandas separadas denominadas pistas. Estas pistas, a su vez, están subdivididas en sectores. En el Apple II, por ejemplo, cada pista se divide en 16 sectores. Cada sector posee un campo de dirección y un campo de datos. El sistema operativo en disco da acceso a los sectores individuales de una pista valiéndose del campo de dirección, que contiene los números de sector y de pista, y de un identificador (para verificar si el usuario está leyendo el disco correcto). Por lo tanto, puede recuperar información de modo bastante similar al de su recuperación de una dirección de memoria (utilizando su dirección)

El ordenador personal "olvidará" toda la información con la cual usted lo había programado tan pronto como se interrumpa la alimentación eléctrica. Este hecho puede implicar, en el mejor de los casos, una pequeña molestia, y, en el peor, un gran disgusto: ver cómo ha desaparecido inadvertidamente una programación cuya realización le ocupó toda una tarde. Por esta causa los fabricantes de ordenadores personales incorporan un procedimiento que permite almacenar con carácter permanente el contenido de la memoria del ordenador. Éste consiste, por lo general, en una cinta de cassette en la cual el programa se almacena digitalmente como una serie de tonos.

Sin embargo, cuando se trata de programas extensos o de una serie de programas cortos que han de utilizarse con frecuencia, el tiempo que se requiere para localizar y cargar el programa desde una cassette puede significar un serio contratiempo. Y ello se debe a dos razones. La primera es que para poder localizar un programa grabado en cinta, ésta se debe reproducir desde el principio (para lo que son de gran ayuda las grabadoras con contador de vueltas).

La segunda razón a la que aludíamos está relacionada con la forma en que se almacena el programa. Los patrones de bits retenidos en la memoria han de convertirse en una secuencia de tonos correspondiente: un tono alto representa a un bit que está encendido (o fijado en uno) y un tono más bajo representa a un bit que está apagado (o fijado en cero). Estos tonos deben luego grabarse en la cinta de la cassette. En la práctica, la mayor velocidad a la cual puede realizarse esta transferencia es de 150 bytes por segundo. Si se supera esta velocidad, el margen de error se incrementa hasta tal punto que el sistema deja de ser fiable.

Un sistema de cassette convencional que utilice una cinta C-10 puede invertir hasta cinco minutos por cada cara para hallar y direccionar un programa, siempre que se emplee un sistema de carga rápido. Algunos sistemas funcionan a una velocidad de tan sólo 30 bytes por segundo. Para estos programas extensos se necesitaría un sistema que hallara el comienzo del programa y lo cargara en cuestión de segundos.

Un sistema de almacenamiento de estas características es el disco flexible (conocido también como *floppy* o *diskette*), cuya utilización es viable en la mayoría de los ordenadores personales actuales. Si imagina los metros de cinta almacenados en una cinta de cassette que adopta la forma de un disco giratorio de aproximadamente cinco pulgadas de ancho (12,70 cm), comprobará con qué rapidez se puede localizar cualquier información almacenada en el disco. Éste viene alojado dentro de un contenedor protector y se introduce en una unidad de disco. Ésta tiene la función de hacer girar el disco (en el interior de su contenedor) a una velocidad constante, así como de proveer los medios para transferir los programas del disco al ordenador y viceversa. Esto lo realiza a través de una cabeza de grabación y reproducción, similar a la de las grabadoras de cassette aunque mucho menor. A diferencia de la de la cassette, que sólo pasa la cinta hacia adelante, esta cabeza puede moverse adelante y atrás a través de la superficie del disco mientras éste gira.

Placa impresa analógica
Este sistema de circuitos convierte las señales que salen de la cabeza o llegan a ella. Traduce la forma digital utilizada por la máquina en la forma analógica que requiere el disco

Indicador
Este LED (diodo emisor de luz) indica si se le está dando acceso a la unidad de disco

Eje de transmisión
Engancha el disco plástico y lo hace girar en el interior de su contenedor

El cuidado de los diskettes
Los discos flexibles son delicados y deben manejarse con gran cuidado. Atienda cuidadosamente las recomendaciones del fabricante

¡NO LO DOBLE! ¡NO LO APILE! APÁRTELO DE CAMPOS MAGNÉTICOS ALMACÉNELO CUIDADOSAMENTE MANTÉNGALO A TEMPERATURA AMBIENTE

Cable plano
La información se transmite a, y desde, la unidad de disco a través de este cable plano. Contiene ocho bits y otras señales de control

Conexión del cable plano
Proporciona una conexión segura, aunque desmontable, del cable plano

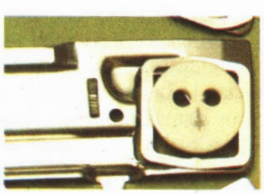
Engranaje motor
Hace girar el eje de transmisión

Cabeza de lectura-escritura

Motor de transmisión
Motor de gran precisión que hace mover la cabeza a través de la superficie del disco

Mecanismo de sujeción del disco
Conectado con el alerón de la puerta, este mecanismo de palanca proporciona al disco una situación exacta en el eje de transmisión

Cortesía de Newbury Data



Cabeza de lectura-escritura
Esta fotografía es una gran ampliación de la cabeza que lee y escribe la información en la superficie del disco. Es parecida a la cabeza de una grabadora de cassette, pero para el ojo humano es casi invisible

A diferencia de una cinta, que no es más que una larga cadena de bytes, un disco se compone de una serie de círculos concéntricos, cada uno de los cuales es tratado como trozos pequeños, por lo general de 256 bytes cada uno. Cada "sector" tiene una dirección.

Cuando se ha de escribir un programa en el disco, lo primero que sucede es que la cabeza se dirige hacia el directorio, un archivo especial que actúa a modo de índice de todo el disco. La cabeza examina el directorio para decidir dónde coloca el archivo. Si el programa se está reescribiendo, encuentra el primer sector de la copia antigua y la nueva información se almacena a partir de allí. Si se trata de un archivo nuevo, éste no tendrá entrada propia en el directorio, por lo cual habrá de otorgársele una: en este caso, la información se localiza en el primer sector libre, llenándose, de ser necesario, más sectores.

Las ventajas que ofrece el disco gracias a su expedito rendimiento y a su gran capacidad de almacenamiento explican la sustancial diferencia de precio entre una unidad de disco y una grabadora. La primera podría costar hasta seis veces más que una unidad de cassette. Esta diferencia de precio refleja el alto nivel tecnológico que requiere la producción de unidades de disco. La cabeza de grabación y reproducción de ésta es casi invisible y ha de colocarse en un espacio que mide apenas unas centésimas de pulgada.

El mecanismo que hace mover a esta minúscula cabeza se basa en un motor eléctrico que puede girar en fracciones de un grado. Está acoplado a un eje que transporta a la cabeza y la desliza a través de la superficie del disco en pasos calculados con suma precisión. Para asegurar que el disco gira a una velocidad constante se utilizan complejos sistemas electrónicos, y todos los componentes están montados sobre una estructura troquelada que se caracteriza por su especial resistencia, al objeto de reducir los efectos del calor y las vibraciones.

Desafiando a los elementos

A diferencia de sus congéneres simples, las variables con subíndice pueden contener cualquier número de elementos

En nuestro programa anterior para calcular cuántos días faltaban para la Navidad, nos encontramos con un nuevo tipo de variable denominada *variable subíndice*. Ésta se diferencia de las variables corrientes o "simples" en que en el interior de la caja puede albergar un número cualquiera de elementos. Las variables simples reconocen dos letras o letras seguidas de un dígito entre 0 y 9 (algunas versiones de BASIC permiten utilizar palabras enteras para los nombres de las variables). A, B, B1, C3 y R2 son todas variables simples. Las variables subíndice son, por ejemplo, de esta manera: A(6), B(12) o X(20). El número entre paréntesis es el subíndice. Los ejemplos que hemos dado se leen así: "A sub seis", "B sub doce" y "X sub veinte".

Si imaginamos a una variable simple como una caja que posee un nombre o una etiqueta, podemos asimismo pensar que una variable subíndice es una caja que contiene una cantidad específica de elementos internos. Si deseamos una variable con doce elementos, la creamos utilizando en primer lugar la sentencia DIM, de esta manera: DIM A(12). Se puede emplear cualquier letra del alfabeto.

A las variables simples se les asigna un valor de forma directa, utilizando tanto la sentencia LET como la sentencia INPUT, como: LET A=35, LET B1=365 o INPUT C3. A los elementos de las variables subíndice se les asigna valores de la misma manera. Veamos ahora cómo podemos asignarle valores a una matriz subíndice. (*Matriz* es otra palabra para designar un conjunto de variables subíndice.) Por ejemplo:

```
10 DIM A(5)
```

crea una variable subíndice con cinco elementos. Ahora podemos asignarle un valor a cada uno de estos elementos:

```
20 LET A(1)=5
30 LET A(2)=10
40 LET A(3)=15
50 LET A(4)=20
60 LET A(5)=100
```

Para comprender la diferencia que existe entre estas variables y las variables simples, asignemos valores a algunas variables simples:

```
70 LET X=5
80 LET Y=6
90 LET Z=7
```

Dé entrada a estas variables en su ordenador y luego verifique el contenido de cada una utilizando la orden PRINT. En BASIC muchas de las sentencias funcionan también como órdenes. Después de que haya dado entrada a las sentencias anteriores, verifíquelas pulsando LIST, pero después no digite RUN. En vez de RUN, digite PRINT X<CR>. En la pantalla debería visualizarse 5 instantáneamente. A continuación digite PRINT Y. El ordenador responderá a esta orden PRINT visualizando 6 en la pantalla. Si desea verificar los elementos de la variable subíndice, digite PRINT A(1) para averiguar el valor del primer elemento de la matriz. El ordenador debe responder imprimiendo 5 en la pantalla. Intente imprimir (PRINT) los valores de A(3) y A(5).

La importante diferencia entre las variables subíndice y las variables corrientes radica en el hecho de que el subíndice puede ser una variable en sí mismo. Para comprender lo que esto significa, digite PRINT A(X). La pantalla responderá con el número 100. ¿Por qué?

Observe la lista que ha digitado y luego verifique el valor de la variable X. Es 5. A(X) equivale a A (el valor de la variable X) y ésta equivale a A(5). Digitar PRINT A(X) es, por lo tanto, exactamente lo mismo que digitar PRINT A(5). ¿Qué valor esperaría si digitara PRINT A(Y−X)? Antes de intentarlo, trate de calcular usted mismo el resultado.

Asignando valores

Si sólo hay unas pocas variables simples, el modo más sencillo de asignarles valores es mediante la sentencia LET. Pero las variables subíndice bien pueden poseer en la matriz una gran cantidad de elementos; veamos, entonces, cuáles son los procedimientos alternativos para dar entrada a sus valores:

```
10 DIM A(5)
20 PRINT "ENTRE LAS VARIABLES"
30 INPUT A(1)
40 INPUT A(2)
50 INPUT A(3)
60 INPUT A(4)
70 INPUT A(5)
```

La digitación de este procedimiento es tan tediosa como si se emplearan sentencias LET, aunque funcionará con seguridad. Si sabemos exactamente cuántas variables hay (en este caso hay cinco), es más fácil utilizar un bucle FOR-NEXT como éste:

```
10 DIM A(5)
20 FOR X=1 TO 5
30 INPUT A(X)
40 NEXT X
```

Según este programa, para ejecutarlo se habrían de digitar cinco valores en el teclado del ordenador. Después de dar entrada a cada número habría que pulsar la tecla RETURN. Si sabemos de antemano cuáles son los valores de la variable, es mucho más sencillo darles entrada utilizando una sentencia READ junto con una sentencia DATA, de este modo:

```
10 DIM A(5)
20 FOR X=1 TO 5
30 READ A(X)
40 NEXT X
50 DATA 5, 10, 15, 20, 100
```

Pruebe con este corto programa y luego verifique los contenidos de la matriz utilizando la orden PRINT; es decir, utilice PRINT después de haber ejecutado (RUN) el programa. Por ejemplo, PRINT A(1) <CR> y PRINT A(5). Ahora podemos agregarle algunas líneas al programa para que se nos impriman automáticamente los elementos de la matriz:

```
60 FOR L=1 TO 5
70 PRINT A(L)
80 NEXT L
90 END
```

Ejecute (RUN) este programa y compruebe que en la pantalla se impriman los valores correctos. Después vuelva a digitar la línea 50 utilizando cinco ítems de DATA diferentes. Recuerde que en una sentencia DATA los números han de ir separados entre sí mediante comas, pero que no debe haber una coma antes del primer número ni después del último.

La forma más sencilla de asignar valores es utilizando las sentencias READ y DATA. Si los valores van a ser diferentes cada vez que se ejecute el programa, probablemente lo mejor sea emplear la sentencia INPUT dentro de un bucle FOR-NEXT. En el caso de que el número total de elementos en la matriz sea fijo, este número puede ser utilizado como límite máximo en la sentencia FOR.

Apliquemos todo cuanto llevamos aprendido hasta ahora para construir un programa corto pero eficaz. Supongamos que deseamos clasificar algunos números por orden ascendente. Antes de ponernos a escribir el programa, lo primero que debemos hacer es hallar una forma lógica de resolver el problema. Cuando la solución a éste parezca clara, habremos de escribir uno por uno todos los pasos utilizando oraciones cortas y concisas.

Supongamos que empezamos con cinco números: 4, 9, 2, 8, 3. Clasificarlos por orden ascendente es una tarea irrelevante. Simplemente, miramos la línea para ver cuál es el más pequeño y lo colocamos a la izquierda; luego repetimos el mismo proceso para los dígitos restantes.

El ordenador, sin embargo, necesita de una serie de instrucciones muy precisas, de modo que nosotros hemos de pensar con mucha claridad los pasos que se requieren. Un enfoque es el siguiente: compare el primer dígito con el segundo dígito. Si el primer dígito es mayor que el segundo deje el primero y coja el segundo. Si el primer dígito es menor que el segundo, no los cambie de lugar.

Compare el segundo dígito con el tercer dígito. Si el segundo dígito es menor que el tercero, déjelos en la misma posición.

Repita el proceso de comparar los dígitos de a pares hasta haber comparado el último par.

Si no hubo que cambiar ninguna posición, todos los números han de estar en orden. Si, por el contrario, hubo que invertir la posición de algún par, vuelva al comienzo y repita la operación.

Si analiza este proceso comprobará que, efectivamente, sirve para clasificar cualquier serie de números según un orden ascendente. Observe lo que sucedería con nuestra serie original de números a medida que los dígitos se comparan de a pares:

```
4 9 2 8 3
4 2 9 8 3
4 2 8 9 3
4 2 8 3 9
```

Ahora todos los pares han sido comparados y, cuando ha sido necesario, se han invertido las posiciones. Puesto que se ha producido por lo menos una inversión de posiciones, volvamos a comenzar y repitamos el proceso:

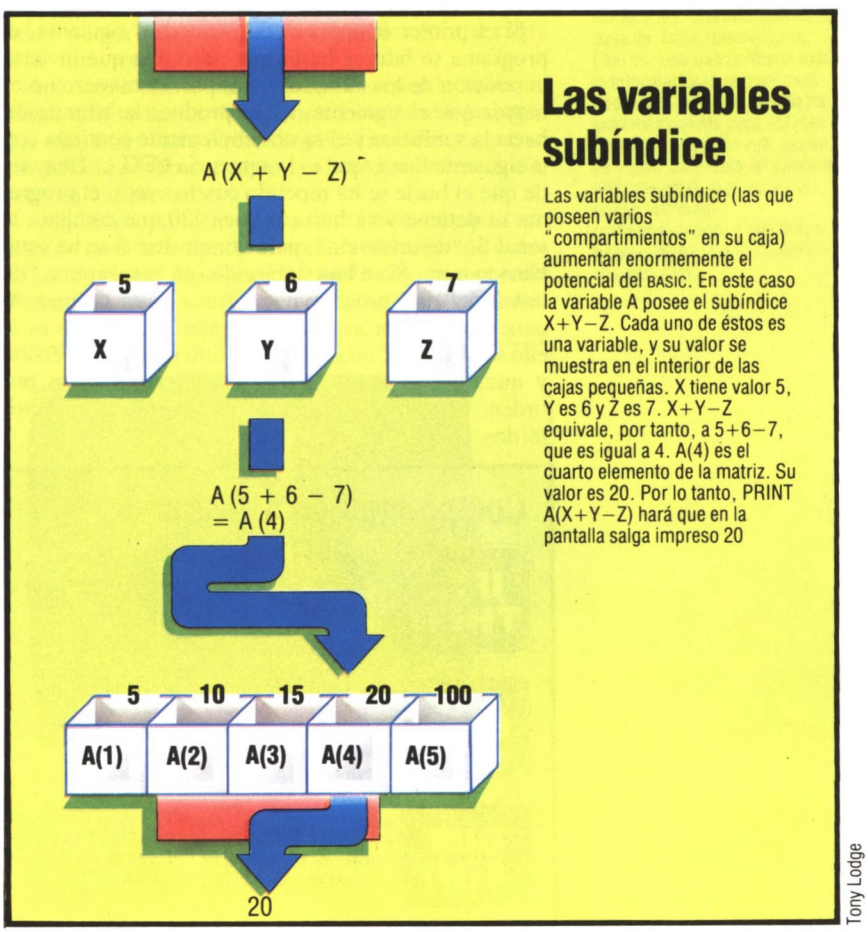
```
4 2 8 3 9
2 4 8 3 9
2 4 3 8 9
2 4 3 8 9
```

Aún se han producido dos inversiones, por lo tanto volvamos al principio y repitamos:

```
2 4 3 8 9
2 3 4 8 9
2 3 4 8 9
```

En esta última no ha habido inversiones, de modo que todos los números han de ser menores que el número situado a su derecha. Los números deben estar por orden ascendente y, en consecuencia, puede terminarse la operación.

La utilización de las variables subíndice permite que en BASIC una rutina de clasificación como ésta se realice fácilmente, porque el subíndice puede ser, en sí mismo, una variable. Si nuestros cinco números originales fueran los valores de una matriz, como: A(1)=4, A(2)=9, A(3)=2, A(4)=8 y A(5)=3, y si el valor de X fuese 1, luego A(X) sería el contenido de A(1), que es 4. A(X+1) sería el contenido de A(2), que es 9, y así sucesivamente.



Las variables subíndice

Las variables subíndice (las que poseen varios "compartimientos" en su caja) aumentan enormemente el potencial del BASIC. En este caso la variable A posee el subíndice X+Y−Z. Cada uno de éstos es una variable, y su valor se muestra en el interior de las cajas pequeñas. X tiene valor 5, Y es 6 y Z es 7. X+Y−Z equivale, por tanto, a 5+6−7, que es igual a 4. A(4) es el cuarto elemento de la matriz. Su valor es 20. Por lo tanto, PRINT A(X+Y−Z) hará que en la pantalla salga impreso 20

Tony Lodge

Observe el programa y vea si comprende exactamente lo que está sucediendo. La línea 20 establece que el valor de la variable N es la cantidad de números que deseamos clasificar. Supongamos que deseamos clasificar cinco números: al ejecutar el programa digitaremos 5 y luego pulsaremos RETURN.

La línea 30 es la sentencia DIM (DIMensión). Si N es 5, establece en 5 el tamaño de la matriz. Esta línea equivale a DIM A(5).

Entre las líneas 40 y 60 hay un bucle FOR-NEXT que nos permite digitar los cinco números. La mayoría de las versiones de BASIC alertan al usuario con un signo de interrogación en la pantalla. Después de dar entrada a cada uno de los números se debe pulsar RETURN. Los números pueden ser de más de un dígito y pueden incluir fracciones decimales.

La línea 90 establece que la variable S es 0. Esta variable se está utilizando como "señal". Más adelante en el programa, A se compara para ver si es o no es 1. Sólo se establece en 1 en el supuesto de que se haya invertido la posición de dos números, como veremos más adelante, en la línea 240. Más avanzada nuestra obra estudiaremos con mayor detalle de qué manera se utilizan las "señales".

La línea 100 establece los límites de un bucle; en este caso, entre 1 y 4 (porque, como N es 5, N − 1 es 4). La primera vez que se realiza el bucle L es 1, de modo que A(L) en la línea 110 será A(1) o el primer elemento de la matriz, y A(L+1) será A(2), el segundo elemento de la matriz. La próxima vez que se realice el bucle, L se aumentará a 2, de modo que A(L) equivaldrá a A(2) y A (L+1) equivaldrá a A(3). La línea 110 establece una comparación para ver si A(L) es mayor que el número de la matriz que se encuentra situado inmediatamente a su derecha. El signo "mayor que" es >.

Si el primer número es mayor que el siguiente, el programa se bifurca hasta una subrutina que invierte la posición de los números. Si el primer número no es mayor que el siguiente, no se produce la bifurcación hacia la subrutina y el BASIC simplemente continúa con la siguiente línea, que es la sentencia NEXT L. Después de que el bucle se ha repetido cuatro veces, el programa se detiene y va hasta la línea 130 que compara la señal S, "de inversión", para comprobar si se ha establecido o no. Si se ha establecido (en la subrutina "de inversión"), el programa se bifurca hasta la línea 90 para repetir el proceso de comparación. Si S no es 1, ello significa que no se ha producido ninguna inversión y que, por lo tanto, todos los números están por orden. El resto del programa es tan sólo para imprimirlos.

Complementos al BASIC

IF... THEN

Si este programa es para ejecutarse en el Spectrum, la línea 130 ha de modificarse para que diga: 130 IF S=1 THEN GOTO 90

END

El Spectrum no dispone de esta sentencia y, por lo tanto, se debe cambiar la línea 170 para que quede: 170 GOTO 260, y se debe agregar la línea 260 REM FIN DEL PROGRAMA

DIM A(N)

Este programa no funcionará en el TI 99/4A, debido a que los subíndices, como X en la línea 50 y L en la línea 110, han de ser números específicos y no variables

Para poder almacenar uno de los números que han de invertirse, la subrutina "de inversión" ha de ser una variable temporal. Después de que en las líneas 210, 220 y 230 los dos números se han invertido, la señal "de inversión" S se establece en 1 y entonces el programa retorna (RETURN) al programa principal.

```
10 PRINT "¿CUANTOS NUMEROS DESEA
   CLASIFICAR?"
20 INPUT N
30 DIM A(N)
40 FOR X=1 TO N
45 PRINT "NUMERO SIGUIENTE"
50 INPUT A(X)
60 NEXT X
70 REM
80 REM RUTINA DE CLASIFICACION
90 LET S=0
100 FOR L=1 TO N-1
110 IF A(L) > A(L+1) THEN GOSUB 200
120 NEXT L
130 IF S=1 THEN 90
140 FOR X=1 TO N
150 PRINT "A(";X;")=";A(X)
160 NEXT X
170 END
180 REM
190 REM
200 REM SUBRUTINA DE INVERSION
210 LET T=A(L)
220 LET A(L)=A(L+1)
230 LET A(L+1)=T
240 LET S=1
250 RETURN
```

Ejercicios

■ Extienda el programa para hallar el valor promedio de la entrada de números. El promedio es igual a la suma de los ítems dividido por el número total de ítems. La forma más sencilla de hacerlo consiste en insertar un GOSUB justo antes de la sentencia END de la línea 170. La subrutina habrá de leer cada uno de los elementos de la matriz y agregar los valores para dar una variable "suma". Después de que se hayan sumado todos los elementos, la suma habría de dividirse por el número total de los elementos. La suma se deriva con bastante mayor facilidad utilizando el número de elementos como límite máximo de un bucle FOR-NEXT.

■ Modifique una línea del programa de manera tal que los números queden clasificados según un orden descendente.

■ Este ejercicio está dirigido especialmente a los usuarios del TI99/4A, ordenador que no admite que se utilicen variables como subíndices de variables subíndice. No obstante, el BASIC de TI acepta sentencias similares a DIM A(12). Reescriba el programa de modo que la sentencia INPUT espere la entrada de una cantidad exacta de números, por ejemplo, 12. Esto evitará el problema de tener que utilizar como subíndice un nombre de variable. También habrán de modificarse las líneas 100 y 110. Por la misma razón, la subrutina de inversión no funcionará en el BASIC del TI. También esto tendrá que modificarse.

■ He aquí un ejercicio difícil. El modo de clasificar los números que hemos elaborado no es, de ninguna manera, el único que existe. Intente pensar un procedimiento alternativo.

Selector

Un ordenador puede seleccionar hechos y recopilar listas a partir de la información almacenada en una base de datos

Tony Lodge

¿Alguien juega al tenis?
Hemos utilizado el cubo de Rubik como una analogía de una base de datos rudimentaria. Esto es, una que contiene toda la información que podemos necesitar, pero que aún no ha sido manipulada para obtener el orden correcto. En este ejemplo, se busca un tenista (el símbolo de la raqueta), que posea un coche (símbolo correspondiente) y que tenga tiempo libre el día en cuestión (cuadro rojo).

Una acumulación de datos almacenada en un ordenador y que sea accesible a éste, se conoce como "base de datos". Todos usamos diversas bases de datos no mecanizadas en nuestra vida cotidiana.

El listín de teléfonos es un ejemplo de base de datos no mecanizada. Sin embargo, la información no necesita estar clasificada o almacenada en un orden determinado para ser una base de datos. En un ordenador, ese orden determinado crea realmente limitaciones importantes.

Un programa de base de datos es una serie de rutinas que permite hacer una selección de datos. La gama de programas varía desde sistemas de fichas a lenguajes completos.

Normalmente, una base de datos mecanizada será amplia y contendrá información de muy diferente tipo. Pero ello no significa que sea necesario poseer una máquina de enormes dimensiones. Cualquier ordenador puede manejar una base de datos práctica. La única limitación real es el tamaño y la velocidad de la memoria.

Por ejemplo, podríamos confeccionar una lista denominada "gente" que contenga datos sobre varias personas. Si colocamos estos datos en fichas corrientes, obtendremos una lista similar a la denominada "índice personal". Resulta evidente que en ella hay diferentes clases de información. Dentro de cada categoría, muchos de los apartados son palabras simples, y otros son números. En dos de ellas las posibilidades son limitadas: "sexo" puede ser sólo hombre o mujer, y en "estado civil" deberá ser "soltero, casado, divorciado o viudo".

Podría ser útil colocar ciertos apartados en listas de palabras o números. Por ejemplo: profesión, nombre de la empresa, dirección profesional, teléfono profesional y el nombre del director podrían agruparse bajo el encabezamiento de "empresa", mientras que el modelo y número de años del coche pueden formar parte de la lista "coche"

Aplicando este mismo concepto, se puede hacer un listado con todas las direcciones. Esto es más práctico que mantener cada dirección por separado, ya que se puede desear saber en qué ciudad vive una persona, pero no en qué calle.

Asimismo también puede ampliarse el apartado "estado civil" añadiendo el nombre del cónyuge cuando sea aplicable. Éste puede consistir sólo en una palabra, pero como hace referencia a una persona determinado para ser una base de datos. En un ordenador, ese orden determinado crea realmente limitaciones importantes.

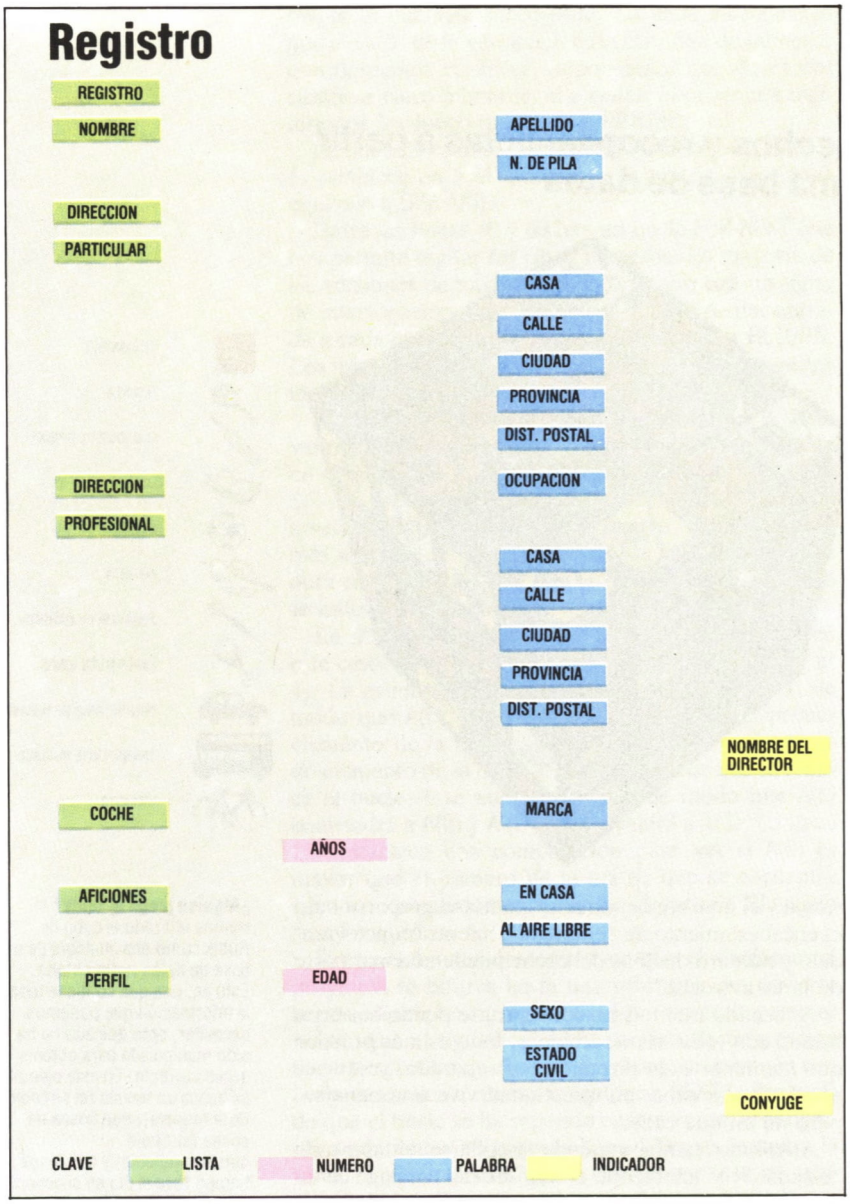
```
DATABASE  HCCSOFT
ENTER DATE AS DD/MM/YY: 14/10/93

*ACCESS

* use AMIGOS
* while AMIGOS, do COCHES
* while AFICIONES, do TENIS;
* end

.CLOSE
```

```
JAMIGOS, COCHES subconjunto aficiones: TENIS

MARAVILLAS MARTINEZ          340 75 56
JOAQUIM CALVET               347 80 46
PILAR SIERRA                 313 17 35
ALBERTO GRAELLS              307 77 19
JOSE ANTONIO BERMEJO         231 69 38

JEND
```

Con preguntas adecuadas

La mayoría de programas de base de datos comerciales emplean códigos que por sí mismos son casi lenguajes de programación. El código que hemos usado aquí es característico. "ACCESS" indica al programa que deseamos "interrogar" o formular preguntas al archivo que hemos creado previamente. "Use" indica que vamos a utilizar un subconjunto del archivo "AMIGOS", y "while" (mientras) estas condiciones sean verdaderas, extraeremos todos los ítems etiquetados con "COCHE" o "TENIS". El resultado es una lista de amigos, con sus números de teléfono, que tienen coche y juegan al tenis

Ian McKinnell



Información organizada
Una base de datos jerarquizada conduce al usuario de una información a otra, ofreciéndole cada vez una elección. Se da por supuesto que no se conoce el contenido

minada, y éste es el tema del archivo, sería útil si esta palabra se pudiera relacionar de algún modo con otro registro.

Puesto que cada registro está en un lugar determinado del archivo, se le asigna un número. Así se puede utilizar el número de registro que describe a esa persona, en lugar de su nombre, para establecer una entrada.

A este tipo de dato se le llama registro "índice". Si se utiliza esta técnica para referirse al director de la empresa en la que trabaja una persona determinada, el resultado es una estructura parecida a la denominada "registro".

La diferencia entre una ficha y una base de datos mecanizada es que la primera sólo se puede clasificar de una forma, generalmente la alfabética.

La ficha es adecuada si se quiere saber, por ejemplo, en qué empresa está empleada una persona determinada. Pero ¿qué podemos hacer si lo que se pretende es saber el nombre de todos los empleados de determinada empresa?

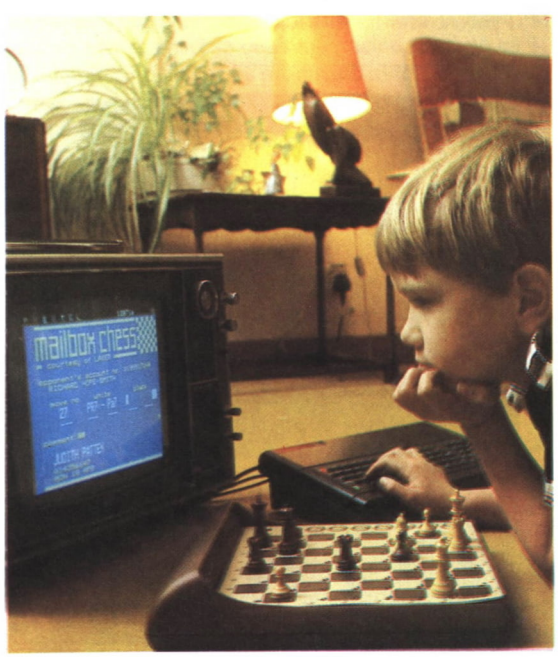
Si se utilizan fichas, tendríamos que mirarlas todas y extraer las adecuadas. Hacer esto no sólo requiere tiempo, sino que probablemente será causa de errores.

Con un sistema mecanizado, sin embargo, se le puede pedir a la máquina que busque sucesivamente cada registro y que imprima el nombre de cada persona que trabaja para la empresa en la que estamos interesados.

Por otra parte, se puede hacer que la máquina clasifique el fichero y que los datos referentes a esta empresa sean los más importantes. Se tendría así la misma base de datos, con los mismos datos, pero con una "forma" totalmente diferente. La clasificación situaría todo lo referente a una empresa dada en un grupo, y esta sección daría los nombres de todos los empleados. En fichas, sólo habría un tema prioritario, el apellido. En cambio, en un sistema mecanizado, cualquier apartado puede ser el principal.

De este modo se puede "reformar" la base de datos. Por ejemplo, el tema principal puede ser los coches que poseen estas personas, o el nombre de las ciudades en que viven. Ésta es la gran ventaja que ofrece el sistema de base de datos mecanizado.

La enciclopedia de nuestros días
Los servicios del tipo Prestel son un intento de hacer accesibles al gran público bases de datos de gran tamaño mediante servicios que los usuarios ya poseen en sus casas. Los sistemas de videodatos usan el televisor doméstico como monitor, y mediante un teclado similar al de un microordenador, pueden conectar con el ordenador central a través de las líneas telefónicas ordinarias. Los distintos servicios que ofrece la base de datos se muestran en la pantalla, y el usuario llega a la "página" deseada de información siguiendo el organigrama. Introduciendo el código de la tarjeta de crédito, incluso se pueden hacer compras desde casa



Cortesía de Prestel Ltd.

Términos clave

Las leyes del pensamiento

Un siglo antes de que se inventara el ordenador, George Boole publicó sus ideas sobre la lógica matemática

Cortesía de la Royal Society

George Boole (1815-1864) George Boole, que nació antes de que los ordenadores electrónicos fueran incluso imaginados, es uno de los fundadores de la lógica matemática usada por los ordenadores actuales. Era hijo de un zapatero remendón y aprendió matemáticas por sí mismo en sus ratos libres. Estaba convencido de que las decisiones que las personas toman cada día se basan en la razón, y de que ésta puede expresarse en términos de lógica matemática. Boole publicó sus ideas en 1847 y, casi inmediatamente, se convirtió en un personaje famoso. Llegó a ser el primer profesor de matemáticas en la entonces recién creada Universidad de Cork

En 1815, el año de la derrota de Napoleón en la batalla de Waterloo, ocurrió otro acontecimiento significativo para la historia de la humanidad. En efecto, en dicho año nacía en Lincoln, Inglaterra, George Boole, hijo de un zapatero remendón que habría de convertirse en uno de los genios que harían posible la invención del ordenador. Aunque murió en 1864, un siglo antes de que empezara la revolución del microordenador, sin sus ideas no hubiera sido posible el desarrollo del ordenador moderno.

Boole sabía que los procesos de razonamiento que las personas efectúan normalmente pueden ser descritos en términos de la lógica formal, de la cual fueron precursores los griegos. Creía que si se intentaba firmemente, se podía llegar a expresar el razonamiento humano en términos matemáticos. Boole se puso a hacer exactamente esto; aprendió matemáticas por sí mismo y empezó sus investigaciones sobre la lógica de las decisiones humanas.

Conjuntos de información

Imagínese que una noche acude a una fiesta. Le apetece bailar y por ello va a la sala y busca una pareja.

La gente de esta sala o bien está bailando o no lo está: no puede hacer ambas cosas a la vez. La pareja a la que se acerca o es un chico o una chica. Obviamente, una persona puede ser hombre o mujer, pero no las dos cosas.

Boole habría enfocado el problema de otra forma. Hubiera visto una pista de baile que incluía "conjuntos" de gente: el conjunto de hombres y el conjunto de mujeres, o H y M. Boole hubiera visto también B y E, o el conjunto de personas que bailan y el de las que esperan para bailar.

Lógicamente, su pareja tendría que satisfacer dos condiciones: ser del sexo opuesto y estar también esperando para bailar. Boole se dio cuenta de la importancia del "y" que relaciona las dos condiciones y le atribuyó un símbolo: una ∪ invertida. Entonces pudo catalogar el conjunto de posibles parejas de baile como M∩E (o H∩E).

Sin embargo, si esta persona no quisiera bailar, sino sólo charlar con alguien, podría elegir a cualquiera de H o M, porque estos dos conjuntos comprenden a todas las personas de la sala. Otra vez, Boole vio la importancia de la aparentemente inocente "o" de la premisa y le dio el símbolo ∪. Así, en su álgebra lógica H∪M incluye a todos los hombres y mujeres presentes en la sala.

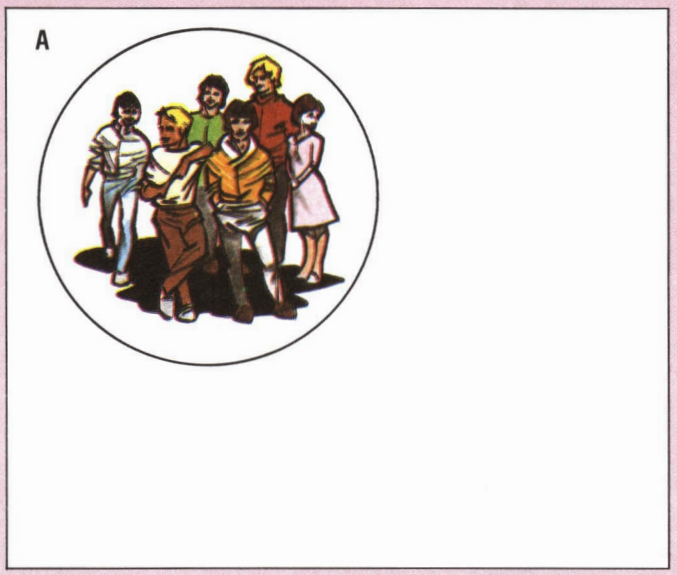
Las puertas lógicas de los ordenadores se designaron como los símbolos de Boole: AND (y) y OR (o). En programación BASIC, descubriremos pronto dos órdenes muy útiles llamadas AND y OR. Pero, volviendo sobre el tema, hay una interpretación muy pintoresca de la lógica booliana, inventada por los matemáticos ingleses John Venn (1834-1923) y Charles Dodgson (1832-1892), este último más conocido como Lewis Carroll, seudónimo con el que publicó numerosas obras para niños.

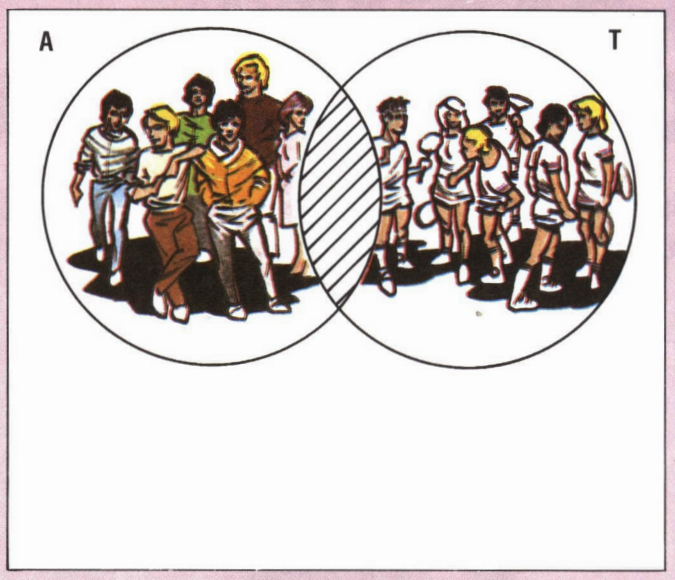
Veamos un ejemplo práctico. Imaginemos que en la

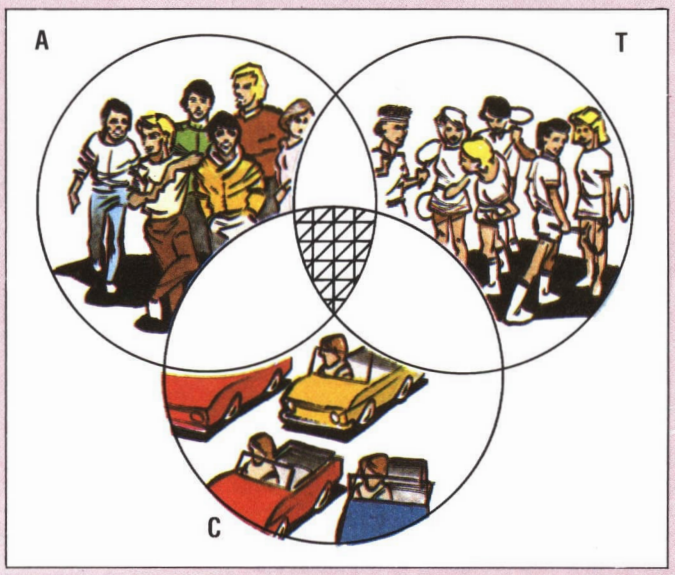
El programa "Amigos"

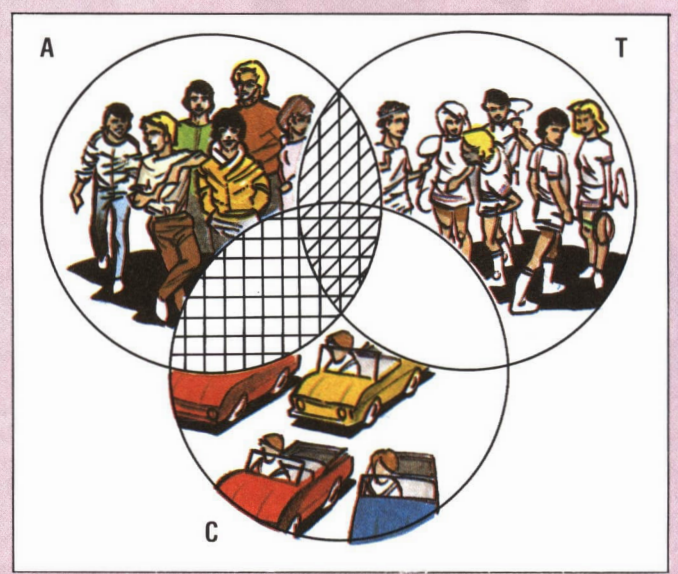
```
 10 DIM N$(10), D$(10), F$(10), T$(10), C$(10)
 15 REM NOMBRE, NO. TEL., ¿AMIGO?, ¿TENIS?, ¿COCHE?
 17 PRINT "INTRODUCIR DETALLES EN LA FORMA":
 18 PRINT "NOMBRE, TELEFONO, SI/NO, SI/NO, SI/NO"
 20 FOR K=1 TO 10
 30 INPUT N$(K), D$(K), F$(K), T$(K), C$(K)
 40 NEXT K
 45 REM RUTINA BUSCAR
 50 FOR J=1 TO 10
 60 IF F$(J)="SI" AND T$(J)="SI" AND C$(J)="SI" THEN GOSUB 100
 70 NEXT J
 80 END
100 PRINT N$(J), D$(J)
110 RETURN
```

Diagramas de Venn









El recuadro representa la colección, o conjunto, de todas las personas listadas en el ordenador. Generalmente, en los diagramas de Venn, éste recibe el nombre de conjunto universal. Los círculos representan los conjuntos individuales. Los amigos se identifican como las personas que figuran en el conjunto A. No todas las personas juegan al tenis. Las que sí lo hacen se incluyen en el conjunto T. Como algunas cumplen ambas condiciones, los dos círculos se superponen (A∩T). El conjunto C identifica a aquellos que poseen coche. Este conjunto interseccíona con los otros dos, y las personas que cumplen las tres condiciones están situadas en ese área (A∩T∩C). El último diagrama representa a los mismos conjuntos (A,T,C), pero las condiciones que se deben cumplir han cambiado. Ahora las premisas son jugar al tenis o ir a dar una vuelta en coche. El conjunto de conocidos que son amigos y juegan al tenis es el representado por diagonales y verticales. El de conocidos que son amigos y poseen coche es el cuadriculado. La intersección de ambas superficies representa a los amigos con los que se podría jugar al tenis o ir a dar una vuelta en coche

Kevin Jones

memoria de su ordenador ha almacenado una lista de las personas que conoce. Con cada nombre se incluye otra información; por ejemplo: número de teléfono, aficiones, etc. Una tarde decide que quiere jugar un partido de tenis en una pista situada en el otro extremo de la ciudad. Para ello necesita un amigo (en contraposición al concepto conocido) que juegue al tenis y que tenga coche. Su ordenador recibe las instrucciones para que escriba el nombre y el número de teléfono de todas las personas que cumplan las tres condiciones: juega al tenis Y tiene coche Y es un amigo.

El programa de la página anterior analiza, primero, la información sobre cada conocido: ¿es amigo?, ¿tiene coche?, ¿juega al tenis? Se supone que usted tiene 10 conocidos, pero puede variar su número según desee (recordando cambiar el 10 entre paréntesis de la sentencia DIM en la línea 10). La lista es examinada por la orden IF...THEN, en la cual se ha insertado una condición múltiple. La mayoría de programas BASIC permiten a la orden IF...THEN trabajar sobre una premisa formada por subcondiciones unidas por AND y OR. Por último, aparecen en la pantalla el nombre y el número de teléfono de todos los conocidos que reúnen las condiciones requeridas: ser amigo, jugar al tenis y tener coche.

En algunos programas se hallan combinaciones muy complejas de funciones lógicas. El álgebra de Boole, que en su tiempo representó poco más que una curiosidad, ha cobrado su verdadera importancia en la era del ordenador.

Organice su programa

Ordenemos un programa usando funciones preestablecidas para reorganizar la información

Nuestra habitual sección de "Programación Basic" ilustra sobre cómo un programa relativamente complejo puede dividirse en subprogramas sencillos o subrutinas que es posible escribir y verificar por separado.

Además de la ventaja de poder probarlas por separado, el empleo de subrutinas permite desarrollar el programa según una progresión lógica. Existen muchas formas de escribir un programa en BASIC. Una de las más comunes es la denominada "prueba y error", que consiste en empezar a introducir líneas en BASIC en el ordenador sin haber pensado cuidadosamente cómo funcionará el programa. Este procedimiento conduce a programas mal estructurados que no funcionarán la primera vez. Si la estructura del programa no es clara, no resulta fácil encontrar los errores o *bugs*.

Una forma mucho más correcta de abordar el problema sería empezar a trabajar a partir de unas notas previas y elaborar primero la estructura del programa, perfilándolo cada vez con mayor precisión, hasta poder escribir un programa correcto y que funcione. Realizar un diagrama de flujo también ayudará. Veamos cómo se hace.

PROBLEMA: Escribir un programa que introduzca un número de nombres de personas, el nombre de pila seguido por el apellido. Invertir después el orden, de forma que aparezca el apellido en primer lugar, seguido por una coma y un espacio y por el nombre. El programa debe clasificar los nombres por orden alfabético e imprimirlos.

Por ejemplo, si se introducen los nombres BERNARDO TORRES y FRANCISCO ALVAREZ (en este orden), el programa imprimirá:

ALVAREZ, FRANCISCO
TORRES, BERNARDO

Antes de intentar escribir un programa que realice esto, es necesario escribir el input y output deseado en términos muy generales:

Paso 1
Introducir nombres sin orden, primero el nombre de pila
Extraer nombres en orden alfabético, primero apellidos

Esto clarifica lo que queremos que haga el ordenador. Es un primer paso esencial para ordenar adecuadamente el programa. El paso siguiente consiste en afinar las etapas del primero y asegurarse de que el programa funciona. A este nivel, no es necesario detallar en exceso. Simplemente, escribir con algo más de precisión las partes de que consta:

Paso 2
Detallar número de nombres a introducir
Introducir nombres
Invertir nombres
Clasificar nombres
Imprimir nombres

Revisemos la lista anterior y comprobemos si puede funcionar. ¿Hay algo que está mal? ¿Hay algún defecto en la lógica? Si todo está correcto, se puede pasar al siguiente nivel de precisión.

Los apartados del paso 2 son lo suficientemente reducidos y sencillos como para poder escribirlos por separado en subprogramas pequeños. En BASIC, los subprogramas reciben el nombre de *subrutinas*. Asignemos nombres a las subrutinas para identificarlas con mayor facilidad. Subrutina 1, detallar número de nombres a introducir: puede denominarse NUMERO. Subrutina 2, introducir nombres: INTRODUCIR. Subrutina 3, invertir nombres: INVERTIR. Subrutina 4, clasificar nombres: CLASIFICAR. Por último, subrutina 5, imprimir nombres: IMPRIMIR NOMBRES.

Paso 3.1 NUMERO
El operador debe introducir el número requerido
Obtener el número N
Usar N para establecer la longitud del vector del string

Paso 3.2 INTRODUCIR
Si el número de nombres es menor que N, hacer que el operador introduzca otro nombre
Añadir nombre al vector

Paso 3.3 INVERTIR
Hallar longitud del string (nombre)
Encontrar "espacio" en el vector
Colocar caracteres en serie hasta "espacio" en una variable del vector provisional
Colocar caracteres en serie desde "espacio" hasta el final en otra variable provisional
Añadir coma y espacio al final de variable
Asignar segunda seguida por primera variables provisionales al vector original

Paso 3.4 CLASIFICAR
Comparar primer ítem del vector con siguiente
Si primer ítem es mayor que el siguiente (posterior en el alfabeto), intercambiar
Comparar segundo ítem con tercero
Intercambiar, si es necesario
Repetir hasta comparar todos los pares
Retroceder y volver a empezar vector y repetir comparación de pares hasta que no se produzcan intercambios

NOTA: Esta rutina de clasificación es exactamente la misma que la empleada en la parte anterior del curso de programación. La sección "intercambio" *(swap)* será tratada como una subrutina extraída de la subrutina CLASIFICAR.

Paso 3.5 IMPRIMIR NOMBRES
Imprimir cada ítem del vector hasta que hayan sido impresos todos

Cada uno de los pasos necesarios para construir este programa ha sido ya desarrollado con el detalle necesario. La rutina de CLASIFICACION sólo ha sido mencionada superficialmente, puesto que ya se habló de ella en el capítulo anterior de "Programación Basic". Por último, INTERCAMBIAR *(swap)*, que se extrae de la subrutina anterior, ha sido omitida por completo. Veamos ahora qué sencillo resulta convertir programas tratados en lenguaje corriente a un programa en BASIC.

Paso 4
1. NUMERO

Las tres líneas del paso 3.1 traducen directamente en términos BASIC. El operador recibe instrucciones mediante la sentencia PRINT, el número se halla con INPUT y la relación de nombres se dimensiona utilizando la sentencia DIM:

```
PRINT "¿CUANTOS NOMBRES DESEA INTRODUCIR?"
INPUT N
DIM A$ (N)
RETURN
```

Ahora la variable N contiene el número máximo de nombres que se deben introducir. DIM dimensiona un vector. Las variables contienen series de caracteres alfanuméricos, en vez de números. Un nombre de variable siempre termina con un signo "dólar". A$ sola únicamente puede contener un string. DIM A$ (N) crea una relación que puede contener N strings. Las variables subíndices ya han sido estudiadas con anterioridad en este curso.

La sentencia RETURN devuelve el control al programa principal, en la línea siguiente a la que se ha introducido la subrutina. Los valores asignados a las variables serán retenidos por el programa principal y pueden emplearse en cualquier punto de éste, incluso en otras subrutinas.

2. INTRODUCIR

Mientras el número de nombres introducido sea menor que N, hay que recordar al operador que introduzca un nombre, el cual debe ser añadido al vector. Esto hace necesario crear un bucle FOR-NEXT; sabemos que el primer nombre del vector será su primer elemento, y que el último será el enésimo, por consiguiente:

```
FOR X = 1 TO N
PRINT "INTRODUCIR NOMBRE"
INPUT A$ (X)
NEXT X
RETURN
```

Esto debería ser suficiente para introducir todos los nombres, pero el lector atento habrá intuido qué sucede cuando se invierte el orden del nombre y apellido en la subrutina INVERTIR. Cada elemento (nombre) del vector debe ser extraído otra vez, invertido y vuelto a colocar en el vector. En vez de complicar y alargar el programa al hacer esto, sería más sencillo extraer la subrutina INVERTIR del interior de la subrutina INTRODUCIR después de que se haya impreso cada nombre. El nombre puede invertirse así antes de ser asignado al vector. Para realizar esto, sólo se tiene que añadir una línea:

```
FOR X = 1 TO N
PRINT "INTRODUCIR NOMBRE"
INPUT A$(X)
GOSUB [INVERTIR]
NEXT X
RETURN
```

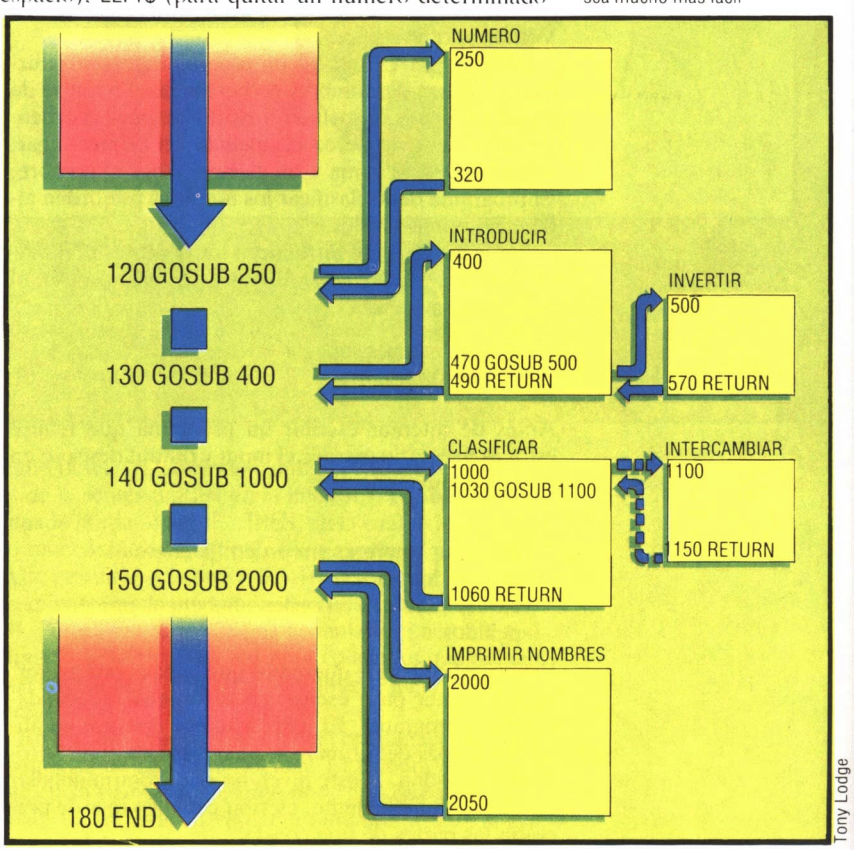
Todos los nombres del vector aparecerán ahora en orden inverso (primero el apellido, seguido del nombre de pila) y, en consecuencia, estarán listos para ser clasificados.

3. INVERTIR

Para invertir el orden de los nombres, necesitamos saber dónde va el "espacio" que separa el nombre de pila del apellido. Cuando sepamos dónde está este espacio, podremos utilizar varias funciones para extraer secciones del vector y asignarlas a otros vectores. Las funciones en lenguaje BASIC consisten en órdenes que realizan una operación predefinida sobre el valor que sigue al nombre de la función. Esta parte está siempre entre paréntesis. Muchas funciones están ya incorporadas, pero también cabe la posibilidad de definirlas uno mismo. Una típica función "preestablecida" es la SQR ( ). Esta función halla la raíz cuadrada del valor entre paréntesis. Así, LET A = SQR(9): PRINT A imprimirá un 3.

INVERTIR utiliza las funciones LEN (para hallar la longitud de la serie), INSTR (para hallar la posición del espacio), LEFT$ (para quitar un número determinado

Programas dentro de un programa
Esta vez el programa principal es muy corto. El auténtico trabajo es realizado por los subprogramas (llamados, en BASIC, subrutinas). Cada uno de los pasos necesarios para que funcione el programa es separado y escrito como un "miniprograma". Luego son unidos por el programa principal.
Al pasar el programa, cada vez que se encuentra una sentencia GOSUB, aquél se desvía hacia el número de línea de la subrutina especificada y se efectúa esa sección del programa. El final de la subrutina se indica con la sentencia RETURN.
Al llegar aquí, el programa vuelve al punto inmediatamente posterior al GOSUB que ha introducido la subrutina.
Las subrutinas, a su vez, pueden "anidar" otras y desviar el programa hacia ellas. Así, INTRODUCIR conduce a la denominada INVERTIR, y CLASIFICAR, a veces, hace aparecer otra subrutina llamada INTERCAMBIAR.
Al dividir un problema en subrutinas independientes, relacionadas por un programa principal sencillo, se logra que el desarrollo y verificación de éste sea mucho más fácil



Tony Lodge

de caracteres de la izquierda del string) y RIGHT$ (para quitar un número determinado de caracteres de la derecha del string). No entraremos en detalles, por el momento, de cómo actúan exactamente estas funciones. En el próximo apartado del curso veremos con más detalle las funciones en BASIC.

4. CLASIFICAR

CLASIFICAR y la subrutina INTERCAMBIAR extraída de ella son muy parecidas a las rutinas utilizadas más arriba.

5. IMPRIMIR NOMBRES

Es muy sencilla:

```
FOR Q=1 TO N
PRINT A$(Q)
NEXT Q
RETURN
```

Ahora lo único que falta es escribir el programa principal. Es tan simple como:

```
REM PROGRAMA PRINCIPAL
GOSUB [NUMERO]
GOSUB [INTRODUCIR]
GOSUB [CLASIFICAR]
GOSUB [IMPRIMIR]
END
```

Los "nombres" de las subrutinas se han puesto entre corchetes. Algunos lenguajes BASIC pueden llamar a las subrutinas por el nombre, pero la mayoría tiene que usar números de línea. Cuando el programa está realmente escrito, los números de línea correspondientes se insertan en lugar de los nombres de las subrutinas. También se añaden los REM adecuados y los mensajes PRINT.

Ejercicios

Ahora que ya se han explicado casi todas las características más importantes del lenguaje BASIC, es el momento de comprobar su progreso realizando estos ejercicios. La dificultad para resolverlos es diferente en cada caso: fluctúa desde lo muy fácil hasta lo moderadamente difícil.

■ **Variables.** Algunos de los nombres de variables que se dan a continuación pueden almacenar valores numéricos y otros no son nombres válidos para emplear como variables. Marcar con un círculo las variables numéricas válidas y tachar las que no lo sean.

A B6 2Z D$ 15 X$ A12 D9 Q81 Q5 6F H$

■ **Aritmética 1.** Escribir un programa corto para asignar el valor 6 a la variable B y PRINT (imprimir) el valor de B.

■ **Aritmética 2.** Escribir un programa corto para asignar el valor 5 a la variable A, 7 a la variable B y 9 a la C. Sumar los valores de estas tres variables y asignar el valor obtenido a la variable D. PRINT el valor de la variable D.

■ **Aritmética 3.** Mire estas líneas de BASIC y calcule cuál ha de ser el valor de C.

```
LET C = 5 + 4 * 3
PRINT C
```

```
10 REM PROGRAMA CLASIFICA NOMBRES
20 REM POR ORDEN ALFABETICO
30 PRINT "PRIMERO DECIDIR CUANTOS"
40 PRINT "NOMBRES DESEA INTRODUCIR"
50 PRINT "LUEGO INTRODUCIR LOS NOMBRES"
60 PRINT "NOMBRE (ESPACIO) APELLIDO"
70 PRINT "ORDEN."
80 REM
90 REM ESTE ES EL PROGRAMA PRINCIPAL
100 PRINT
110 PRINT
120 GOSUB 250
130 GOSUB 400
140 GOSUB 1000
150 GOSUB 2000
160 REM
170 REM FIN DEL PROGRAMA PRINCIPAL
180 END
250 REM SUBRUTINA PARA HALLAR NO. DE
260 REM NOMBRES A INTRODUCIR
270 PRINT "¿CUANTOS NOMBRES"
280 PRINT "DESEA INTRODUCIR?"
290 PRINT
300 INPUT N
310 DIM A$(N)
320 RETURN
400 REM SUBRUTINA PARA INTRODUCIR NOMBRES
410 PRINT "INTRODUCIR NOMBRE EN ESTA FORMA:"
420 PRINT "NOMBRE(ESPACIO)APELLIDO(CR)"
430 PRINT "P. EJ. ANA TORRES"
440 FOR X = 1 TO N
450 PRINT "INTRODUCIR NOMBRE"
460 INPUT A$(X)
470 GOSUB 500
480 NEXT X
490 RETURN
500 REM SUBRUTINA PARA INVERTIR ORDEN
    DE NOMBRES
510 LET L = LEN(A$(X))
520 LET S = INSTR(A$(X)," ")
530 LET C$ = LEFT$(A$(X),S - 1)
540 LET F$ = RIGHT$(A$(X),L - S)
550 LET F$ = F$ + ","
560 LET A$(X) = F$ + C$
570 RETURN
1000 REM RUTINA CLASIFICAR
1010 LET S = 0
1020 FOR P = 1 TO N - 1
1030 IF A$(P)> A$(P + 1) THEN GOSUB 1100
1040 NEXT P
1050 IF S = 1 THEN GOTO 1000
1060 RETURN
1100 REM SUBRUTINA INTERCAMBIAR
1110 LET T$ = A$(P)
1120 LET A$(P) = A$(P + 1)
1130 LET A$(P + 1) = T$
1140 LET S = 1
1150 RETURN
2000 REM IMPRIMIR SUBRUTINA
2010 PRINT
2020 FOR Q = 1 TO N
2030 PRINT A$(Q)
2040 NEXT Q
2050 RETURN
```

■ **Aritmética 4.** ¿Qué resultado se imprimirá en este programa?

```
LET A = 3
LET B = 2
LET C = 9
LET D = 4
LET E = (A + B) * (C - D)
PRINT E

LET E = 5
LET E = E * E
PRINT E
```

■ **Comparaciones 1.** ¿Cuál será el valor de X correcto para que se imprima el mensaje PRINT?

```
 70 LET A = 5
 80 LET B = X
 90 LET R = B - A
100 IF R = 0 THEN GOTO 120
110 GOTO 10
120 PRINT "¡FELICITACIONES! HA GANADO"
999 END
```

■ **Comparaciones 2.** ¿Cuál es el valor menor de X que hace saltar el programa a la línea 300?

```
250 IF X > 6 * 100 THEN GOTO 300
```

■ **Comparaciones 3.** ¿Cuál es el menor valor de Z que hace saltar el programa al mensaje de "felicitaciones"?

```
340 IF Z < 10000 THEN GOTO 500
350 IF Z > = 10000 THEN GOTO 520
:
:
500 PRINT "SU PUNTUACION ES DEMASIADO BAJA.
    INTENTELO DE NUEVO"
510 GOTO 600
520 PRINT "FELICITACIONES. AHORA ES UN
    MAESTRO"
530 GOTO 700
```

■ **Print 1.** Suponga que el valor de T es 50. Escriba una sentencia PRINT que diga: EL VALOR DE T ES 50. Se aconseja colocar el "mensaje" entre comillas, usar un punto y coma y el nombre de la variable.

■ **Print 2.** Mire el siguiente programa y complete la sentencia PRINT de forma que el programa imprima un mensaje semejante a éste:

```
PERDON, PERO SU PUNTUACION DE 175 ES
DEMASIADO BAJA
```

Complete la línea de forma que el valor real de la puntuación pueda variar cada vez.

```
620 REM LA VARIABLE S ES LA PUNTUACION
    HASTA AHORA
620 IF S < = 500 THEN GOTO 640
630 GOTO 700
640 PRINT "LO SIENTO"
```

■ **Print 3.** ¿Qué mensaje se imprimirá una vez se haya pasado el programa?

```
200 LET A$ = "¿MI COMPUTER?"
210 LET B$ = "¿LE GUSTA?"
220 PRINT B$
230 PRINT A$
```

■ **Input 1.** INPUT es una forma de asignar un valor a una variable. Si se pasa el siguiente programa, ¿qué tecla deberá digitarse para que el programa imprima 12 como respuesta?

```
60 INPUT N
70 LET N = N * 2
80 PRINT N
```

■ **Input 2.** ¿Qué se imprimirá aquí?

```
100 PRINT "POR FAVOR DIGITE SU NOMBRE"
110 INPUT N$
120 PRINT "HOLA"; N$; "SOY SU ORDENADOR"
```

Complementos al BASIC

Este programa no funcionará en el Atari 400/800, puesto que su tratamiento de vectores es muy diferente del de las otras máquinas.

DIM

Existe en el Spectrum, pero su uso no es estándar; por ello es necesario suprimir la línea 310 y reemplazarla por:

```
310 DIM A$ (N,30)
```

GOTO

En la línea 1050, la orden GOTO 1000 viene inmediatamente después de la palabra THEN. En este caso, la mayoría de ordenadores permiten omitir la palabra GOTO; por tanto, la línea 1050 podría escribirse:

```
1050 IF S=1 THEN 1000
```

INSTR

Spectrum, VIC 20, C64 y Oric-1 no disponen de esta función.
En la serie Commodore y Oric-1, se debe suprimir la línea 520 y reemplazarla por estas cinco:

```
515 FOR P=1 TO L
520 CH$=MID$ (A$(X),P,1)
522 LET S=0
523 IF CH$="" THEN LET S=P: LET P=L
525 NEXT P
```

LEFT$

RIGHT$

El Spectrum no dispone de ninguno de estos mandos, pero pueden crearse versiones propias de ellos con DEF FN; por tanto, suprimir la línea 320 y reemplazarla por las cuatro siguientes:

```
320 DEF FN M$ (X$,N)=X$(N)
330 DEF FN L$ (X$,N)=X$ (TO N)
340 DEF FN R$ (X$,N)=X$ (N TO)
350 RETURN
```

Luego suprimir las líneas 510 a 560 y reemplazarlas por las siguientes:

```
510 LET D$=A$ (X)
520 LET L=LEN (D$)
530 LET S=0
540 FOR P=1 TO L
550 IF FN M$ (D$,P)="" THEN LET S=P
560 IF S<>0 THEN LET P=L
570 NEXT P
580 LET C$=FN L$ (D$,S-1)
590 LET F$=FN R$ (D$,L-S)
600 LET G$=F$+"," + C$
610 LET A$ (X)=G$
620 RETURN
```

END

No existe en el Spectrum, Oric-1 y Dragon 32; reemplazarlo por STOP.

El eslabón perdido

Por medio del modem, la información puede pasar de un ordenador a otro a través de miles de kilómetros

Ian Dobbie

El acoplador acústico
La mayoría de los modems que existen en la actualidad son dispositivos "totalmente electrónicos" que conectan directamente con la línea telefónica y se enchufan en el conector del teléfono. Las compañías telefónicas obligan a respetar unas pautas muy rígidas para los dispositivos del tipo de los modems, y ésa es la razón por la cual tienden a ser tan caros. Una solución más económica, puesto que pasa por alto la reglamentación en este sentido, consiste en utilizar un "acoplador acústico". Se trata de una clase de modem que convierte las señales de audiofrecuencia por ondas sinusoidales en sonidos verdaderos que alimentan un pequeño altavoz

La palabra *modem* es la contracción de "modulador /demodulador". A pesar de que ya hace aproximadamente cinco años que los modems se venden en el mercado, sólo ahora está comenzando a aumentar el número de propietarios de ordenadores personales que los utilizan. Si todos podemos ser propietarios de un ordenador, ¿qué sentido tiene gastar dinero en comprarse un modem para conectarlo con el sistema telefónico?

Con un modem, su ordenador podrá «hablar» con otros ordenadores situados en cualquier lugar del mundo. Para ello lo único que se requiere es que el ordenador que se encuentre en el otro extremo de la línea telefónica también posea su propio modem. Este otro ordenador puede ser tan sólo un corriente micro personal perteneciente a otro entusiasta de la informática, o bien un gran ordenador de unidad principal propiedad de una universidad o de una institución financiera. Conectar su ordenador con un gran ordenador de unidad principal puede permitirle el acceso a grandes bancos de datos, servicios de información e incluso a las últimas cotizaciones de la bolsa. Al conectar su micro al de un amigo podrían ambos intercambiar software o enviarse un "correo electrónico" gratuito, e incluso jugar en dos direcciones.

Los modems funcionan de forma similar a la interface para cassette que acompaña a la mayoría de los ordenadores personales. Tanto las interfaces para cassette como los modems convierten los unos y los ceros del ordenador en frecuencias audibles. En el caso de las interfaces para cassette, estas frecuencias se pueden grabar fácilmente, como si fueran señales de audio en la cinta de cassette. En el caso de los modems, las frecuencias audibles se envían a través de la línea telefónica y son convertidas en números binarios por el modem situado en el otro extremo.

Sin embargo, para grabar en cinta las interfaces para cassette sólo necesitan convertir las señales binarias en señales audibles (este proceso se denomina *modulación*). O bien realizan lo contrario y convierten las señales audibles reproducidas por la cassette en señales binarias *(demodulación)*. Por otra parte, la mayoría de los modems están diseñados para comunicarse bidireccionalmente a través de una única línea telefónica y, por tanto, requieren dos bandas de frecuencias y cuatro frecuencias individuales. Un modelo popular usa una frecuencia de 1 070 Hz para un 0 y de 1 270 Hz para un 1 para transmitir, y 2 025 Hz para un 0 y 2 225 Hz para un 1 para recibir. Observará que las dos frecuencias en cada una de las bandas (de baja frecuencia y de alta frecuencia) están muy próximas. Sólo hay una diferencia de 200 Hz de frecuencia para un 1 y un 0 en ambas bandas. Esto contrasta con las interfaces para cassette, en las cuales la frecuencia para un 1 es normalmente el doble de alta que la frecuencia para un 0. Para decodificar frecuencias tan próximas se requiere un sistema de circuitos electrónicos muy complicado y por ello los modems pueden considerarse un lujo: un modem puede costar tanto como varios micros personales pequeños.

La máquina FAX
Las máquinas FAX (abreviación de máquinas facsímil) se están popularizando rápidamente en las oficinas de Europa y de Estados Unidos. En Japón hasta las empresas más pequeñas las poseen, y también se utilizan en muchos hogares. Las máquinas FAX pueden transmitir grandes documentos, incluyendo dibujos e imágenes, a otras máquinas FAX en cuestión de segundos, valiéndose sólo de un modem incorporado y de un teléfono corriente

Alta resolución

Ahora es posible comprar una pantalla para el ordenador, con una alta calidad de resolución para gráficos y juegos

Debido a que cada vez es mayor el número de personas que empiezan a usar ordenadores, videos y otros equipos que necesitan pantalla, los precios de los monitores especializados tienden a bajar. En los primeros años, un buen monitor en color no bajaba de las 150 000 pesetas, pero en la actualidad es posible encontrarlo por la mitad de dicho precio. Obviamente, los monitores monocromáticos son mucho más baratos.

Dado el desarrollo constante de las posibilidades gráficas de los microordenadores, muchos de los cuales son en color, adquirir un monitor es una idea muy acertada.

Existen dos tipos principales de monitores en color: el conocido como RGB (*Red, Green, Blue:* rojo, verde, azul), y el video compuesto. El monitor RGB se controla directamente con los tres cañones de electrones, que forman los colores según las indicaciones del ordenador. Los pulsos que se utilizan para sincronizar el ordenador con el monitor también son producidos directamente por el ordenador.

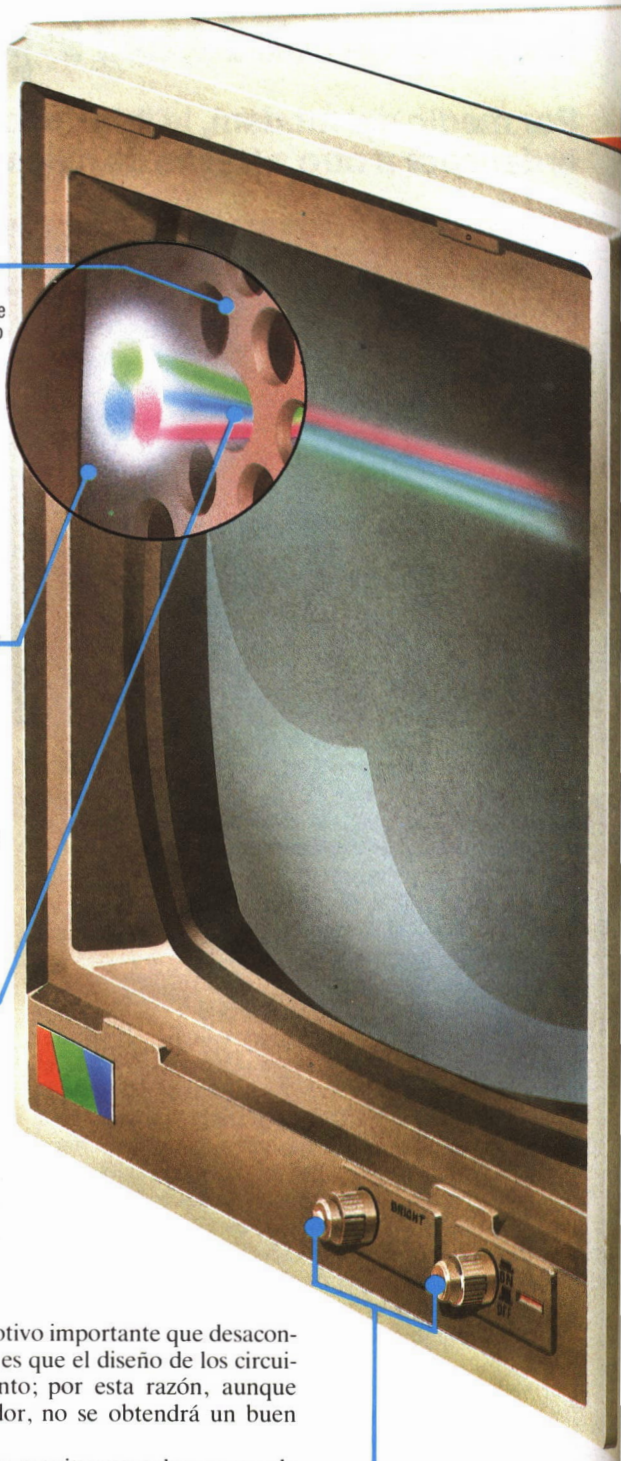
Hay dos tipos de pulsos de sincronización: uno para cada línea de la imagen, y el otro para cada imagen completa. Al final de cada campo, el monitor recibe un pulso, que le dice que se ha llegado al final de la pantalla, y por ello el haz de electrones (y en consecuencia el punto que origina) debe volver a la esquina superior izquierda de ésta.

Al final de cada línea se produce un proceso similar, que indica que esa línea determinada ha sido completada y que el haz de electrones debe regresar al lado izquierdo de la pantalla, para empezar la línea siguiente. En un monitor RGB, cada una de estas señales (roja, verde, azul, sincronismos de línea y de campo) es enviada al monitor mediante cables independientes.

En cambio, un monitor compuesto se asemeja más a un televisor, puesto que todas las señales se combinan en una sola, siendo enviadas al monitor a través de un cable coaxial. Una vez en el monitor, el sincronismo de línea, el de campo y las tres señales del color son de nuevo separados y utilizados para controlar la imagen.

Un monitor es un televisor sin sintonizador. De hecho, se puede transformar un monitor en un televisor añadiéndole un sintonizador, o modificar un televisor normal suprimiendo el mecanismo de selección de canales.

Sin embargo, es totalmente desaconsejable hacer esta adaptación, puesto que en todo componente de un equipo que contenga un tubo de rayos catódicos se generan altos voltajes muy peligrosos. Incluso los técnicos profesionales tratarían este tema con suma precaución.



Rejilla de pantalla
Para asegurar que los cañones de electrones apuntan exactamente al lugar correcto de la pantalla, la superficie del tubo incorpora una rejilla

Fósforo de la pantalla
La imagen en color se compone (tal como se muestra en el diagrama) de tres colores. En el cristal están depositadas varias sustancias que, alcanzadas por el haz de electrones, emiten radiaciones de los tres colores. Al mezclarse, éstos dan las diferentes tonalidades de la imagen, según sea la intensidad del haz en cada punto

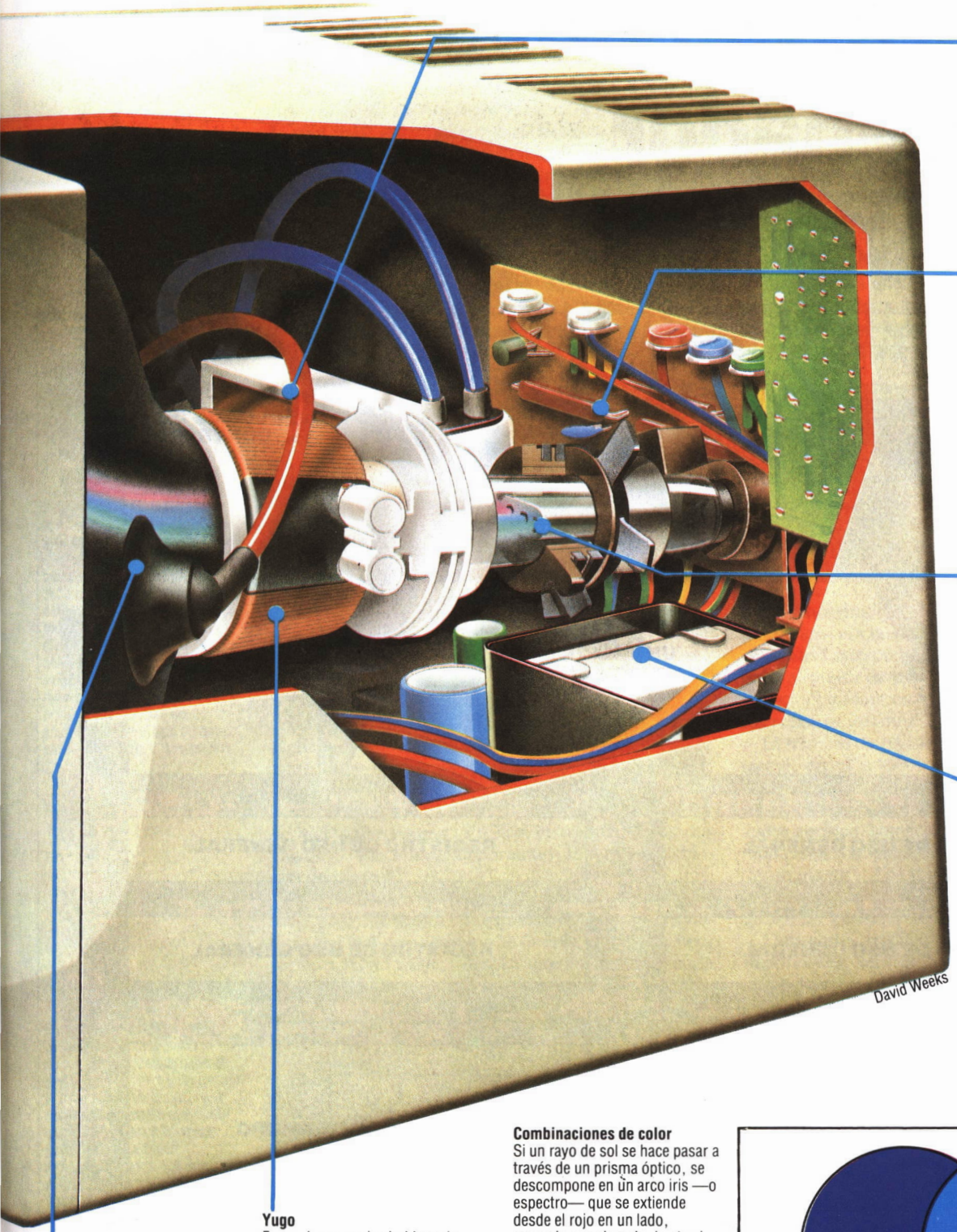
Haces de electrones
En el tubo hay tres haces de electrones, cada uno de los cuales "excita" un elemento de fósforo diferente para producir un punto de color

Controles
En un monitor, al igual que en un televisor, hay varios controles. Normalmente, los de sincronismo vertical y horizontal son accesibles al usuario. La intensidad de color y otras variables, por lo general, no necesitan ser ajustadas y se sitúan en el interior del aparato

Existe además otro motivo importante que desaconseja dicha adaptación, y es que el diseño de los circuitos es ligeramente distinto; por esta razón, aunque se suprima el sintonizador, no se obtendrá un buen monitor.

La razón de utilizar un monitor en color en vez de un televisor como terminal se funda en que éste sólo funcionará con una señal transmitida mediante una onda portadora de UHF (Ultra-High Frequency). Esto significa que la señal limpia generada por el ordenador tiene que ser codificada, enviada a través del cable y decodificada otra vez. Mediante este proceso se recibe una señal "sucia" y, por tanto, se obtiene una imagen borrosa.

Por otra parte, un monitor no necesita esta modulación y demodulación de la señal y la imagen producida será más limpia y precisa. Esto constituirá un descanso para la vista y los programas tendrán un aspecto mucho más profesional.



Circuitos de alta tensión
Debido a que los tubos de rayos catódicos necesitan voltajes muy altos, deben tener un circuito rectificador para elevar el voltaje de entrada (240 v) al nivel requerido

Tablero circuito principal
Los circuitos necesarios para producir las corrientes de control que mueven el haz y hacen girar a los cañones de electrones se encuentran aquí. Parte de la sección de sincronismo de línea, que trabaja a frecuencias muy elevadas, puede utilizarse como fuente de energía para el tubo

Cañones
Un monitor en color, al igual que un televisor, tiene tres cañones: rojo, verde y azul, que están alineados al final del tubo

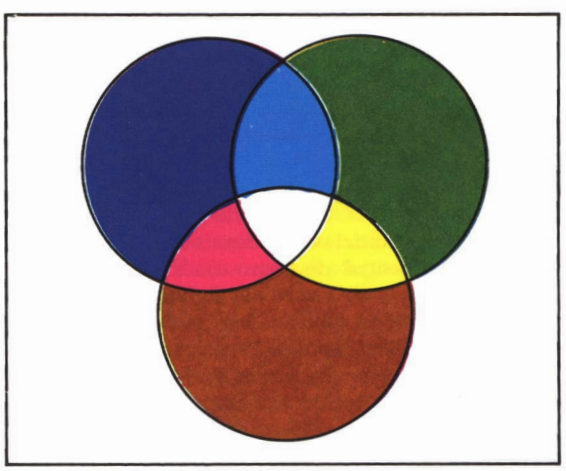
Suministro de energía
Un tubo de rayos catódicos debe funcionar con voltajes de corriente continua muy estables y requiere altas intensidades, y por ello es necesario un gran transformador

David Weeks

Yugo
Formado por varias bobinas de tamaño considerable que producen potentes campos magnéticos. Éstos varían rápidamente, y así se logra el movimiento del punto fosfórico que produce la imagen

Acoplamiento anódico
Una vez que el haz es proyectado desde los cañones, es acelerado por un campo de alto voltaje. Éste debe estar en el otro extremo del tubo y se aplica por medio de una placa fuertemente aislada, situada al final del cable

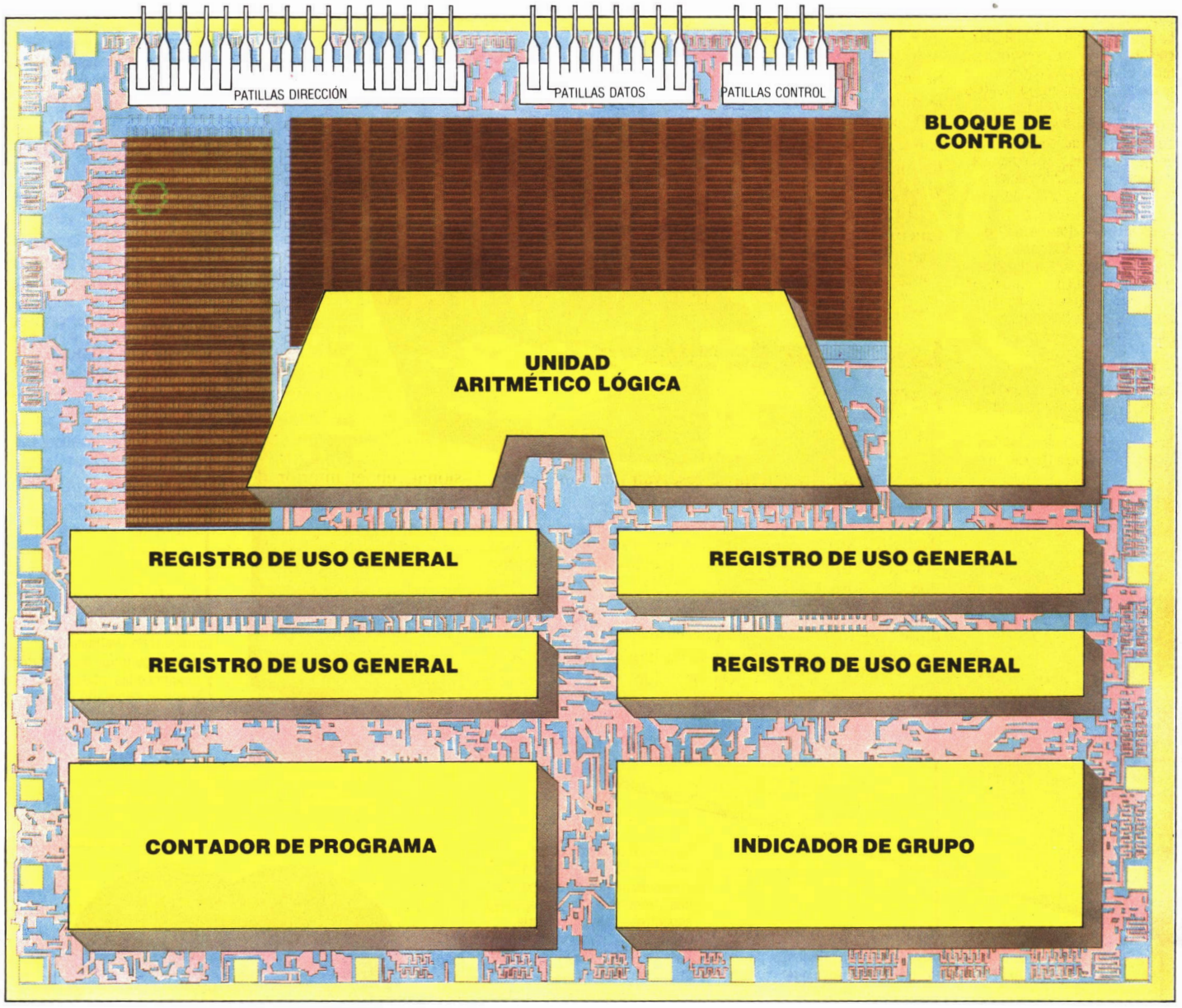
Combinaciones de color
Si un rayo de sol se hace pasar a través de un prisma óptico, se descompone en un arco iris —o espectro— que se extiende desde el rojo en un lado, pasando por el verde, hasta el azul-violeta en el otro. Si este espectro se hace pasar a través de otro prisma similar, los colores se vuelven a combinar para formar la primitiva luz solar (a menudo llamada "luz blanca"). Este proceso de recombinación o adición se utiliza en un monitor en color. Añadiendo intensidades diferentes de los tres colores básicos —rojo, verde y azul— se pueden crear todos los colores



Mark Watkinson

El centro nervioso del ordenador

Todos los canales de la actividad de un ordenador se originan y confluyen en su "unidad central de proceso"



Kevin Jones

El funcionamiento del ordenador es controlado por la CPU (unidad central de proceso). El contenido semántico de estas palabras se puede interpretar casi literalmente: unidad (forma un conjunto independiente); central (está situada en el corazón del ordenador); de proceso (realiza el trabajo). Un ordenador muy simple (véase la ilustración) puede estar formado tan sólo por unos circuitos de CPU, memoria e input/output(I/O).

El ordenador necesita el I/O para comunicarse con el mundo exterior. En una aplicación muy sencilla, un ordenador incorporado a una lavadora automática necesitaría los circuitos de I/O para conectar el motor y los calentadores. La memoria se necesita para almacenar las instrucciones y datos que debe procesar la CPU. Estos datos procesados por la CPU pueden incluir números y códigos binarios que representan caracteres (letras, dígitos y signos tales como @ y !).

Si unas zonas de la memoria contienen instrucciones para la CPU, y otras datos que deben ser procesados por ésta, ¿cómo diferencia ambos conjuntos? Para contestar a esta pregunta, es necesario conocer el interior de un microordenador.

La CPU de los microordenadores de 8 bits (la mayoría de los ordenadores personales pequeños son de este tipo) normalmente está formada por un solo chip de 40 patillas, 20 en cada uno de sus lados. Cada una de estas patillas (excepto las conectadas a la fuente de 0 y +5 v) transporta señales desde o hacia la CPU y otros componentes, por ejemplo los circuitos I/O o los de memoria.

Por lo general, una CPU de 8 bits tiene 16 patillas de dirección, que se conectan al "bus de direcciones". Cada una de estas patillas lleva una señal de salida, que representa un uno o un cero. Se pueden formar 65 536 combinaciones distintas de unos y ceros. Se emplean para seleccionar puntos específicos de la memoria.

También hay ocho patillas de "datos", que están conectadas al "bus de datos". Éstas transportan datos desde la memoria o I/O al interior de la CPU o viceversa.

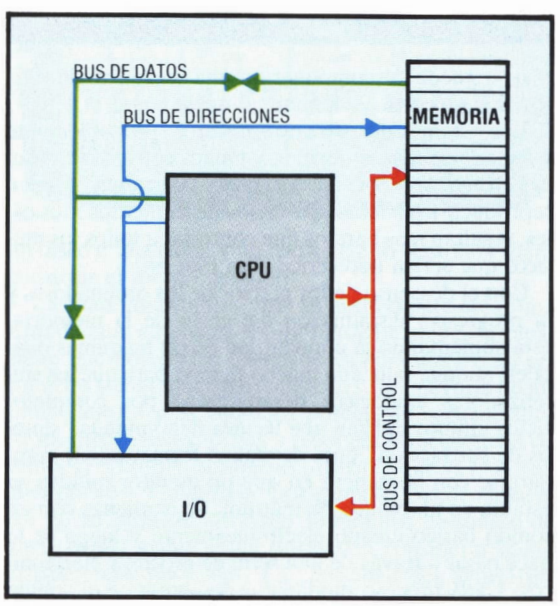
Otras patillas transportan señales de "control" que funcionan como entradas o salidas de la CPU. Más adelante veremos cómo se utilizan estas señales de control.

Bajo control
La ilustración muestra una CPU con "registros" de memoria, una unidad aritmético lógica, que consta de cientos de puertas lógicas (para realizar las operaciones de adición, AND y de complementación de números binarios), y un bloque de control. Este último admite las instrucciones codificadas (en binario), las interpreta y hace que las otras partes de la CPU procesen adecuadamente. Por ejemplo, si una orden significa que el contenido del acumulador debe almacenarse en un punto determinado de la memoria, el bloque de control pondrá la dirección en las patillas correspondientes, enviará señales de control para que la memoria almacene los datos, y pondrá el contenido del acumulador en el bus de datos para transmitirlo a la memoria

Las celdas de memoria o registros

En el interior de la CPU hay unas celdas de memoria de uno o dos bytes, llamadas *registros*. Algunas de estas celdas de memoria se reservan para fines especiales, y las otras se usan para el almacenaje temporal de información. Estas últimas se denominan *registros de utilización general*. En la CPU existen otros dos "bloques" funcionales importantes: la ALU y el "bloque de control".

La ALU (Arithmetic and Logic Unit: unidad aritmético lógica) realiza las operaciones lógicas y aritméticas que incluyen (pero no se limitan a) adición, operaciones AND y OR, y desplazamiento de bits hacia la derecha o la izquierda dentro de un byte.



Cumpliendo órdenes
Un ordenador muy sencillo puede constar únicamente de una CPU, memoria y un circuito I/O. La memoria almacenará órdenes especiales que harán que la CPU realice acciones específicas. También almacenará datos que serán procesados por la CPU, según las instrucciones. El circuito I/O es necesario para que la CPU se comunique con el exterior. Si el ordenador controla una lavadora automática, el circuito I/O introducirá las señales de los mandos y permitirá que salgan las órdenes para conectar y desconectar el motor y los calentadores.
Los códigos de órdenes para la CPU estarán en sistema binario. Cada modelo diferente de CPU tiene su propio conjunto de códigos

El bloque de control es un circuito especial diseñado para lograr que la CPU se comporte de acuerdo con las instrucciones recibidas desde la memoria. Podemos ver un ejemplo muy explícito utilizando los códigos de instrucciones para la popular CPU del Z80. Si desde la memoria se recibe la instrucción codificada 11000110, la CPU sumará el contenido del próximo byte de la memoria al de uno de los registros del interior de la CPU. Si se quiere almacenar el resultado de esta adición en un punto determinado de la memoria, la siguiente instrucción que reciba la CPU tendrá que ser 00110010, seguida de dos bytes que especifiquen la situación real en la memoria en la cual se puede almacenar el resultado.

Supongamos que el resultado de la adición sea 37 (en notación decimal), y que los dos bytes que determinan la dirección sean 33126 (también en notación decimal). El código de instrucción hará que el bloque de control coloque las patillas de dirección según el equivalente binario de 33126 (éste sería 1000000101100110). Asimismo, hará que las patillas de control envíen señales a la memoria diciéndole que va a recibir datos que deben ser almacenados (memorizados). También motivará que las patillas de datos adopten la disposición del equivalente binario de 37 (00100101). Esta información pasará a través del bus de datos a la memoria y será almacenada en el punto de la misma determinado por el bus de direcciones. Si más tarde la CPU necesitara procesar estos datos de otra forma (impresos en la pantalla, por ejemplo), se podría enviar a la CPU una orden diferente. El bloque de control interpretaría esta instrucción como: "Dirigirse al punto de memoria 33126, tomar el byte que hay en él y almacenarlo provisionalmente en uno de los registros internos".

El número de registros, o celdas de memoria provisional, en el interior de la CPU depende de ésta. Serán registros del tipo 8 bits (un byte) o del tipo 16 bits (dos bytes). Por lo general, los registros especializados reciben nombres determinados; por ejemplo, "indicador de grupo", "contador de programa" o "acumulador". Los registros generales reciben la denominación de "el registro X", "el registro Y", "el registro C", etcétera.

Uno de los registros de 16 bits, y uno de los más importantes, será el "contador de programa". Esta celda de memoria interna contiene siempre la dirección (en binario) de la siguiente instrucción que debe ejecutarse. Cuando llegue el momento de extraer la siguiente orden para la CPU, el contenido del contador de programa será colocado en el bus de direcciones, y el byte correspondiente será transmitido (vía bus de datos) a la CPU.

El registro más importante de 8 bits es el "acumulador". Éste es el que, por lo general, almacena (provisionalmente) el resultado de las operaciones realizadas por la ALU: bytes tomados de la memoria o de I/O, o del lugar donde están almacenados provisionalmente antes de ser enviados a éstos.

Esta introducción a la CPU ha sido de carácter muy general; los puntos específicos se desarrollarán en detalle más adelante. El propósito ha sido mostrar que las instrucciones especiales leídas en la memoria hacen que la CPU lleve a cabo determinadas operaciones y disponga las patillas de dirección para poder acceder a puntos concretos de la memoria. Los datos son traídos desde estos puntos, o enviados a ellos, mediante el bus de datos. Las órdenes hacen, asimismo, que las señales del bus de control indiquen a la memoria o al I/O si estos datos deben ser "leídos" o "grabados".

"La voz de su amo"

Los ordenadores ya ocupan un lugar propio en el campo de la música profesional. También se están incorporando pequeños sintetizadores de música en muchos ordenadores personales

Los ordenadores son a la vez divertidos y serios. Y no sirven tan sólo para procesar información o para jugar. También se pueden utilizar como instrumentos musicales para, por decirlo de alguna manera, divertirse con seriedad. El proceso de hacer música artificialmente se denomina síntesis musical.

Los ordenadores también se pueden utilizar para hacer más amena la enseñanza de la música y su concurso resulta bastante más económico que contratar a un profesor particular. En este capítulo no explicaremos cómo se realiza este tipo de música con un ordenador personal, sino que nos concentraremos en cómo lo hacen los profesionales del medio.

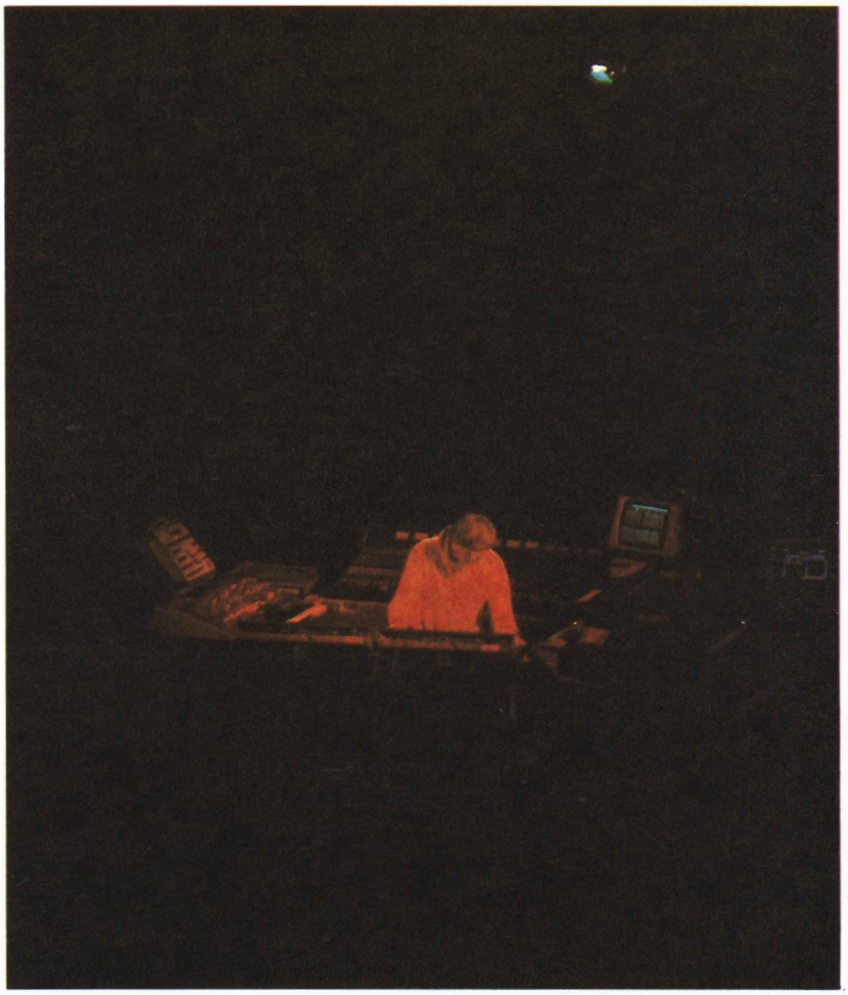
Los instrumentos musicales automáticos siempre han gozado de popularidad y tienen mucho en común con los ordenadores. La pianola, una especie de piano automático que durante el siglo pasado solía adornar las salas de estar de las familias acomodadas, funcionaba mediante un rollo de papel perforado, y las cajas de música llevaban un disco o un tambor metálico con "dientes" que ejecutaban una melodía sobre un peine de metal.

En cierto sentido, hasta los organillos callejeros que funcionaban a manivela eran programables, ya que las melodías que ejecutaban se podían sustituir por otras. No obstante, estos organillos no eran del agrado de Charles Babbage, uno de los padres fundadores de la informática, quien era partidario de que a los organilleros se les prohibiera hacer música en las calles. Éstos, como respuesta, acudían a tocarle melodías bajo su propia ventana.

En la actualidad, en Gran Bretaña, es el Sindicato de Músicos el que está tratando de impedir el uso de dispositivos musicales programables; en mayo de 1982, la sección sindical central de Londres decidió, mediante votación, prohibir su utilización en las sesiones de grabación y en las actuaciones en vivo. Como es evidente, esta preocupación de los sindicatos de músicos nace del hecho de que, dado que estos dispositivos pueden imitar el sonido de muchos instrumentos diferentes y de forma simultánea, con el tiempo los músicos no serían necesarios.

Los sintetizadores electrónicos salieron al mercado hace ya muchos años, pero la introducción de técnicas digitales les ha abierto un campo totalmente nuevo. En vez de tener que manipular clavijas y pulsar botones para producir cada uno de los sonidos, ahora se puede grabar cualquier sonido, analizarlo mediante ordenador en las partes que lo constituyen y reproducirlo en cualquier tono.

El sonido digitalizado es algo así como una fotografía publicada en un periódico; si mira la página desde muy cerca, verá que la imagen se compone de muchos puntos pequeños y separados unos de otros, mientras que la fotografía original (analógica) tiene tonalidades difuminadas de sombras que se mezclan continuamente entre sí. Del mismo modo, el sonido analógico normal se puede descomponer en una secuencia de dígitos. Esta técnica se denomina muestreo.



Cortesía de EMI

Un hombre, una orquesta
Los músicos sintetizadores como Klaus Schultz, a quien vemos en la fotografía, están utilizando cada vez más el enorme potencial de sus instrumentos basados en microprocesadores para producir, en vivo, la misma variedad de sonidos para la que hace veinte años se hubiera necesitado toda una orquesta

Los sistemas de este tipo son caros (probablemente los modelos más conocidos y menos sofisticados sean el Fairlight y el Synclavier), pero, puesto que pueden reproducir los sonidos de varios instrumentos musicales, resultan más baratos que contratar a todos los músicos que serían necesarios para este fin.

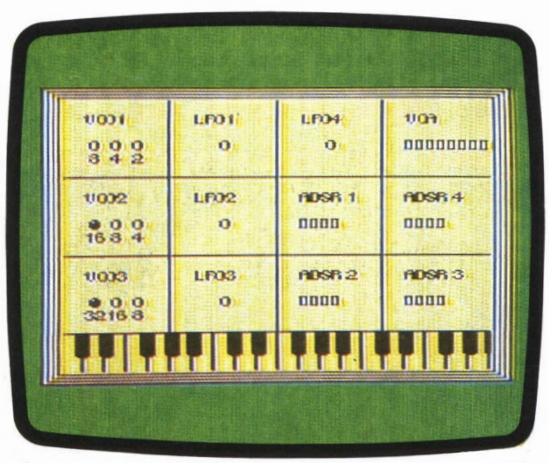
Con el descenso de los precios de los ordenadores y la progresiva disminución del costo de la memoria, está aumentando la popularidad de las máquinas digitales, aunque falta aún mucho tiempo para que los sintetizadores analógicos desaparezcan por completo. Estos últimos utilizan una técnica denominada "síntesis de sustracción" que, de alguna forma, puede compararse con la manera en que un escultor moldea su estatua en un bloque de mármol. Se comienza con un sonido básico creado electrónicamente y luego se lo hace pasar a través de una serie de procesos electrónicos. Cada proceso modifica o descompone el sonido

Ian McKinnell

Efectos especiales
Los sintetizadores de música controlados por ordenadores personales se están haciendo cada vez más populares. Los efectos musicales, que hace diez años sólo existían para los equipos profesionales más costosos, ahora tienen un precio muy asequible. El que vemos en la fotografía puede leer en su memoria música que previamente ha sido codificada en papel en forma de código de barras. Luego el ejecutante la puede reproducir o modificar. Muchos de estos sintetizadores personales se pueden acoplar directamente a un ordenador personal, para aprovechar la pantalla o la memoria adicional. Sin embargo, cada vez son más numerosos los ordenadores personales que llevan incorporada alguna forma de síntesis musical

en sus mínimos componentes, para darle la forma deseada. La síntesis de sustracción estimula a experimentar con diferentes combinaciones de procesos y su técnica resulta sencilla hasta para un principiante.

Por el contrario, todo cuanto se cree con un sintetizador digital ha de planificarse tan cuidadosamente como la construcción de un gran edificio. Ello se debe a que el dispositivo se vale de la *síntesis de adición*: el sonido final se produce mediante la adición de componentes, uno sobre el otro. Uno ha de estar próximo al final del proceso para poder ser capaz siquiera de reconocer mínimamente el sonido. Sin embargo, se puede tomar un sonido convencional, descomponerlo en sus componentes básicos, almacenar éstos en la memoria del ordenador, ya sea en una RAM interna o en un disco o una cinta, y utilizarlos como "ladrillos" para construir el sonido deseado.



Ian McKinnell

Cajas de música
Actualmente existen muchos paquetes para ordenadores personales que permiten aprovechar al máximo la capacidad que éstos poseen para realizar música. En estos paquetes, la visualización en pantalla se puede utilizar para hacer una intepretación visual de la música que se está ejecutando, o para ayudar al músico recién iniciado, que puede pulsar el teclado QWERTY como si fuera el de un piano

Casi tan importante como su aptitud para crear una variedad tan amplia de sonidos individuales es la capacidad que poseen los ordenadores para almacenar secuencias y composiciones musicales. La mayoría de los sintetizadores más vendidos poseen un secuenciador (dispositivo que almacena y recupera las secuencias de sonidos) opcional, aunque es probable que ya lo lleven incorporado. El Fairlight, por ejemplo, puede almacenar en su memoria de disco 30 minutos de sonido de hasta ocho "voces", que se pueden considerar como instrumentos musicales. La capacidad del Synclavier corresponde al doble.

De todo lo anterior se deduce que en los sintetizadores más caros los papeles tradicionalmente asignados al compositor, los músicos y el director se funden en el de una sola persona, que, en gran medida, es, además, un programador.

Si alguna vez ha intentado reproducir a una velocidad mayor una cinta grabada con su voz (o si con ese fin ha reproducido a 45 r.p.m. un disco de 33 r.p.m.), se habrá dado cuenta de que el tono se eleva considerablemente. Una de las características más sorprendentes de los sintetizadores controlados por ordenador es su capacidad para salvar este inconveniente y reproducir una pieza musical a mayor o menor velocidad que la original sin alterar su tono, o, por el contrario, trasponer la melodía a una clave distinta a la misma velocidad.

Incluso es posible tomar, por ejemplo, la parte de una trompeta, duplicarla y, simultáneamente, modificar el sonido para que adopte el de una trompa francesa. Es factible, entonces, tocar ambos instrumentos armónicamente, sea al unísono o de forma alternada Esto se conoce con el nombre de *track bouncing*.

A muchas de estas máquinas se les proporcionan las instrucciones por medio de un lenguaje de composición (el del Synclavier se denomina SCRIPT) que no difiere demasiado del BASIC (también posee números de líneas), aunque tal vez sea un poco más difícil de utilizar. La configuración más relevante es la denominada *compilación invertida*: se ejecuta una pieza musical en el teclado y el ordenador produce un listado en SCRIPT para su composición. Esto equivaldría, si es que uno se pudiera imaginar algo así, a que se pudiera jugar a un juego nuevo creado por uno mismo en la pantalla del ordenador y luego, apretando un botón, ¡se obtuviera un listado completo del programa para este juego!

Si considera que su sentido del ritmo no ha sido todo lo perfecto que deseaba, existe la posibilidad de editar y volver a reproducir el listado en SCRIPT utilizando un teclado convencional (QWERTY) y una pantalla, igual que en BASIC. Uno de los sintetizadores Yamaha tiene un sistema menos flexible pero más atrayente: la melodía que usted haya ejecutado la imprime utilizando el procedimiento musical convencional, es decir, por medio de notas en el pentagrama.

Los sintetizadores también se están introduciendo en el cine. La película *TRON*, producida por Walt

Disney, ha recibido entusiastas críticas, que elogian sus sorprendentes gráficos realizados por ordenador. Mucho menos conocido resulta el hecho de que se emplearon microordenadores para la música y los efectos sonoros. Además de la parte de música "verdadera" ejecutada por la London Philharmonic Orchestra, se grabó y se modificó una gran variedad de sonidos usando un sintetizador Fairlight, incluyendo los ruidos producidos por la aeronave Goodyear y por una nevera.

Fue tal la cantidad de sonidos que se emplearon, que se hubo de mantener actualizado un catálogo completo mediante el paquete de base de datos File Manager 800+, procesado con un ordenador personal Atari 800. Algunas de las voces artificiales de la película también se realizaron mediante un ordenador personal y un dispositivo para síntesis de voz que permitía mezclar voces y música. Atari también les facilitó un paquete que jamás anteriormente se había utilizado fuera de dicha empresa. Consistía en una clase de programa para síntesis de adición que Atari había venido empleando para crear sofisticados efectos de sonido en sus propios programas para juegos y para sus máquinas tragaperras. La impresión general es que la importancia de este "sistema electrónico ensamblador de sonidos" respecto a la creación de sonidos es equiparable a la que tuvo el tratamiento de textos en relación a la creación de textos.

En realidad existe un número bastante elevado de accesorios con los cuales se puede convertir un ordenador personal en un sofisticado sintetizador de música. El Apple es un modelo especialmente popular para este tipo de aplicación, debido a las ranuras para conectar dispositivos adicionales que posee el ordenador en su parte posterior. La finalidad de estos dispositivos es hacer que el ordenador realice todo el trabajo creativo y proporcionar un sintetizador analógico de gran calidad como salida. Otros incluyen un teclado exactamente igual al de un piano.

En el extremo más doméstico del mercado de sintetizadores informatizados, el Casio CT7000 incluye lo que se conoce como *secuenciador polifónico*, el cual, mediante una unidad de cassette y una gran cantidad de memoria RAM, puede funcionar como un sistema de grabación profesional de pistas múltiples. Una grabadora de cassette corriente sólo posee una pista por cada lado, mientras que una estereofónica posee dos. Pero una unidad profesional puede llegar a tener más de 24, de modo que la interpretación de cada instrumento se puede grabar por separado, para luego mezclar todos los sonidos para obtener la banda sonora definitiva. De esta manera, con el CT7000 uno puede crear sus propias sinfonías.

Su antecesor, el CT701, utilizaba un lector de código de barras para leer música a partir de los códigos de barras impresos en la memoria de la máquina. Lamentablemente, parecería que no existen medios viables para que el usuario pueda crear códigos de barras impresos de sus propias creaciones, por lo cual este procedimiento sólo se puede emplear con la música preimpresa del sintetizador Casio.

Para triunfar en el mundo de la música pop, ni siquiera se necesita contar con un equipo de este nivel, como lo ha demostrado el grupo alemán Trio, que con su tema *Da Da Da*, para cuya grabación utilizaron un Casiotone VL1 que cuesta 30 libras (unas 6 700 pesetas), obtuvieron más éxito que el que Peter Gabriel consiguiera con un Fairlight de miles de libras.

A pesar de que el VL1 es un dispositivo monofónico (en el cual sólo se puede ejecutar una nota a la vez), es capaz de emular diversos instrumentos musicales y de almacenar una secuencia de notas. También ofrece la posibilidad de cambiar determinadas cualidades de las notas para que el usuario tenga la posibilidad de crear su propio sonido. Esto se conoce como alterar la envoltura ADSR (Attack, Decay, Sustain, Release: atacar, decaer, sostener, soltar). Ahora bien, lo más interesante es que ordenadores personales como el Commodore 64 y el Oric-1 poseen exactamente las mismas características. Y, por supuesto, el solo hecho de que el aparato sea programable en BASIC le da la facultad de contar con un secuenciador incorporado.

Existen en el mercado paquetes de programas para composición musical para muchos de estos ordenadores personales, incluso para los que, comparativamente, poseen configuraciones musicales más sencillas. Algunos de ellos visualizan en pantalla un pentagrama, y la música se compone seleccionando los diferentes tipos de notas musicales con la ayuda de una palanca de mando o de un lápiz óptico, y colocándolas en el pentagrama. Al pulsar el botón de "disparo" de la palanca de mando, o dar alguna orden igualmente sencilla, la composición musical empieza a ser ejecutada. Otra alternativa consiste en que la visualización represente el teclado de un piano; en este caso las notas también se seleccionan con el lápiz óptico o con la palanca de mando, o bien con el teclado del ordenador.

Los efectos musicales pueden programarse en BASIC sin que sea necesario utilizar un paquete de este tipo. Al igual que las configuraciones para gráficos, el procedimiento exacto varía considerablemente de una máquina a otra, así como la sofisticación de las órdenes en BASIC creadas con este fin. El Dragon, por ejemplo, posee una sola voz, pero dispone de una orden PLAY que inicia una secuencia de notas, que se digitan como la antigua notación de la escala musical: de la A a la G (de *la* a *sol*, según la notación actual). El Commodore 64, por el contrario, tiene incorporadas en su hardware muchas y sofisticadas configuraciones musicales, pero no posee órdenes en BASIC listas para utilizar, como el Dragon. Próximamente le enseñaremos cómo realizar música con un ordenador.

Martin Rushent

Productor de grupos como Altered Images y Dexy's Midnight Runners, Martin Rushent es uno de los vanguardistas de los sintetizadores musicales controlados por ordenador. En su estudio del distrito rural de Berkshire hay no menos de nueve sistemas diferentes, que representan la situación actual de este arte. Cada uno de estos sistemas utiliza un procedimiento de programación diferente y distintos códigos para las notas de una escala musical. Puesto que no ha sido factible conectarlos a todos entre sí para formar un único sistema gigante, Rushent, para trasponer las piezas musicales de uno a otro sistema, utiliza un programa que escribió él mismo en un pequeño ordenador personal.

Gráficos con vida

Las memorias de gran capacidad permiten producir con ordenadores personales imágenes plenas de color y movimiento

Una de las características más sorprendentes de los ordenadores personales es su capacidad para producir gráficos y visualizaciones en movimiento, comúnmente denominadas imágenes animadas. En la mayoría de los microordenadores, el usuario puede trazar puntos individuales, dibujar líneas y círculos y cambiar los colores del fondo y de los primeros planos.

Para los juegos de simulacro y de acción rápida, debemos ser capaces de simular el movimiento. La manera más sencilla de conseguirlo consiste en producir una serie de imágenes fijas, una después de la otra. Esto se debe realizar con la rapidez suficiente como para crear la ilusión de movimiento. Las imágenes de televisión se producen mediante un procedimiento similar al descrito.

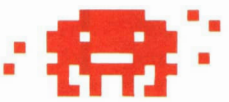
Rapidez de acción

Otra forma de crear la ilusión de movimiento consiste en imprimir un carácter, borrarlo y volverlo a imprimir en una posición ligeramente alejada de la original. Para que el movimiento fluya uniformemente es imprescindible que la distancia entre un paso y otro sea mínima. Asimismo, el tiempo invertido en producir la forma y en borrarla debe ser lo más corto posible.

La utilización del BASIC para producir animación da como resultado un movimiento lento. Una forma de superar este inconveniente es recurrir al lenguaje ensamblador, método que requiere mucha práctica, atención y cuidado si desea obtener visualizaciones libres de parpadeos. Sumado a ello, se plantea el problema adicional de controlar los caracteres individuales a medida que aumenta su número en la pantalla.

Diversos ordenadores, en particular el Commodore 64, el Sord M5, el Texas Instruments TI99/4A y la gama Atari, superan este inconveniente y proporcionan una animación que utiliza las mismas técnicas que emplean las máquinas de juegos recreativos accionados con monedas. Esta técnica se conoce como *gráficos sprite*.

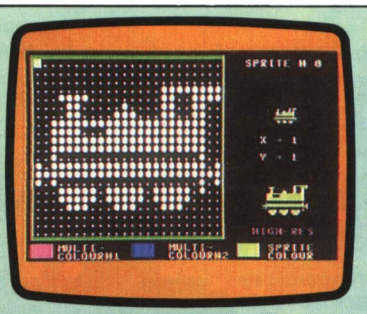
Los sprites son "objetos" o formas que pueden ser desplazados, independientemente unos de otros, a través de la pantalla. Esto se logra, sencillamente, cambiando el contenido de un par de direcciones de memoria, que especifican las coordenadas X e Y (las posiciones izquierda-derecha y arriba-abajo). Típicamente, X puede abarcar entre 0 y 255, e Y entre 0 y 191. Algunos sistemas permiten incluso especificar la velocidad y la dirección del movimiento de cada sprite, y el ordenador hace todo el resto.



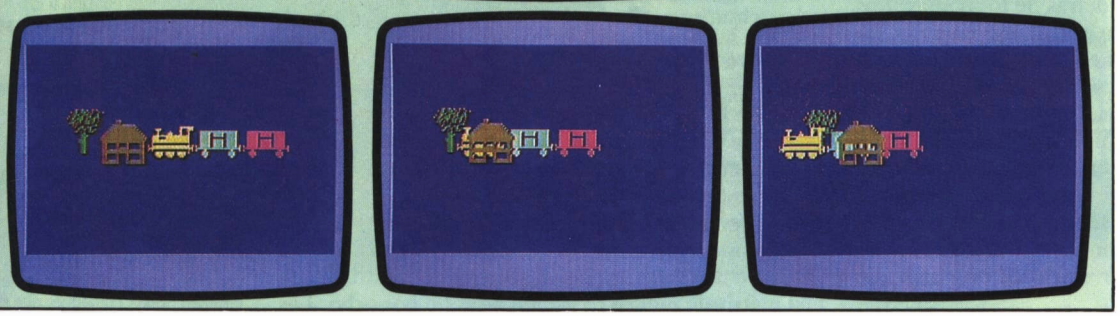
Los sprites normalmente se realizan utilizando chips o

El tren de pensamientos

El tren que vemos en la ilustración se construyó a partir de tres sprites (la locomotora más dos vagones) utilizando en el Commodore 64 un paquete denominado Spritemaker. La imagen se creó a gran escala usando



las facilidades de edición del paquete, y se guardó en una cassette, con las imágenes de la casa y del árbol. Una vez vueltos a cargar en la memoria del Commodore, los sprites se manipularon mediante órdenes POKE para determinar sus posiciones en la pantalla, su color y la velocidad del tren. También se especificó la "prioridad" de los sprites, de modo que el tren pasara por detrás de la casa pero por delante del árbol



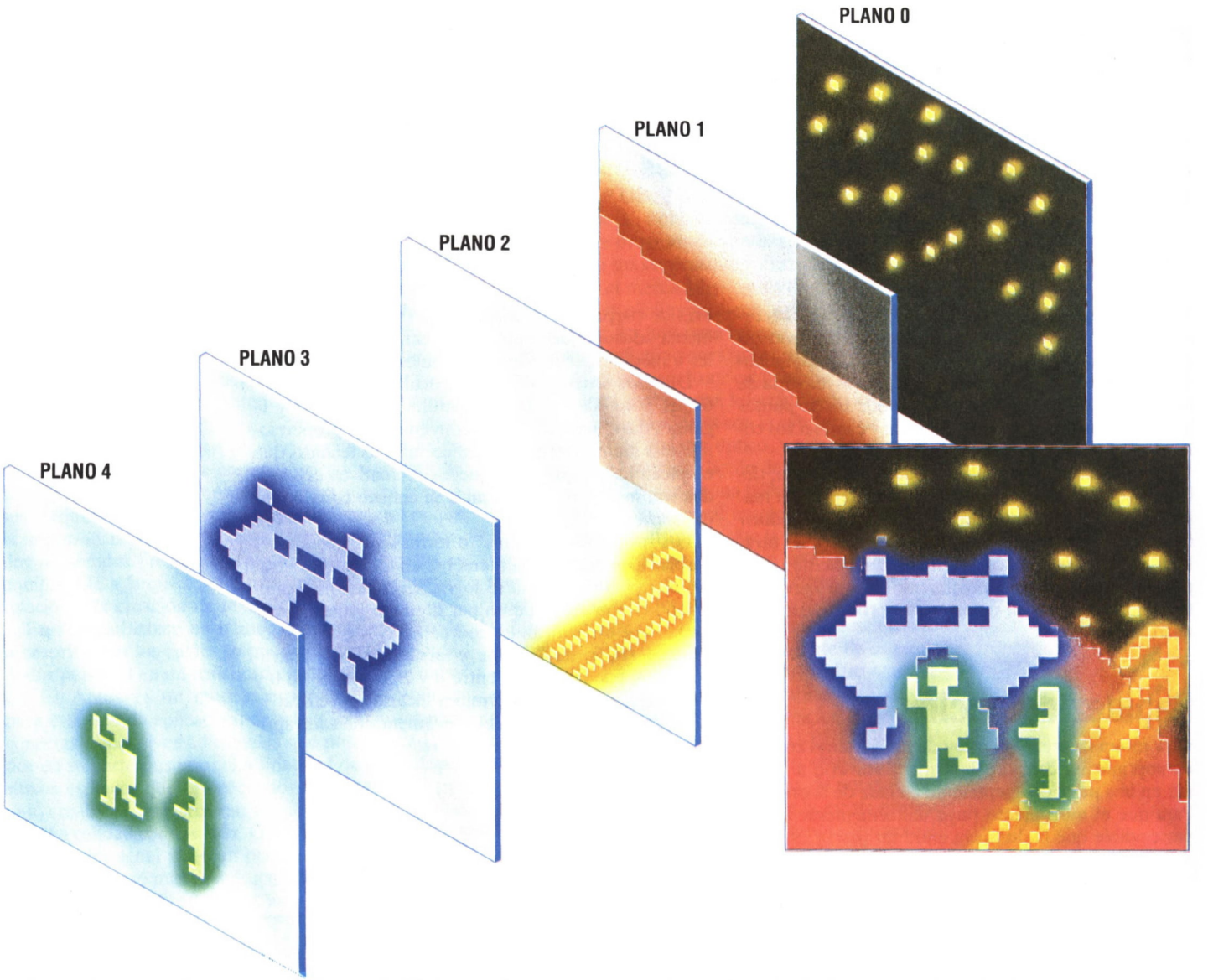
Ian McKinnell

La utilización del BASIC para producir animación da como resultado un movimiento lento. Una forma de superar este inconveniente es recurrir al lenguaje ensamblador, método que requiere mucha práctica, atención y cuidado si desea obtener visualizaciones libres de parpadeos. Sumado a ello, se plantea el problema adicional de controlar los caracteres individuales a medida que aumenta su número en la pantalla.

Diversos ordenadores, en particular el Commodore 64, el Sord M5, el Texas Instruments TI99/4A y la gama Atari, superan este inconveniente y proporcionan una animación que utiliza las mismas técnicas que emplean las máquinas de juegos recreativos accionados con monedas. Esta técnica se conoce como *gráficos sprite*.

Los sprites son "objetos" o formas que pueden ser desplazados, independientemente unos de otros, a través de la pantalla. Esto se logra, sencillamente, cambiando el contenido de un par de direcciones de memoria, que especifican las coordenadas X e Y (las posiciones izquierda-derecha y arriba-abajo). Típicamente, X puede abarcar entre 0 y 255, e Y entre 0 y 191. Algunos sistemas permiten incluso especificar la velocidad y la dirección del movimiento de cada sprite, y el ordenador hace todo el resto.

Los sprites normalmente se realizan utilizando chips o un sistema de circuitos de hardware especiales dentro del ordenador. Es posible comprar software para otros ordenadores con el fin de obtener resultados similares, pero por lo general esta solución resulta poco satisfactoria.



Los sprites se pueden mover a distintas velocidades, ya que cada uno de ellos está situado en lo que se conoce como un *plano de sprite*. Una escena de gráficos en pantalla se compone, por tanto, de una serie de planos apilados uno detrás del otro, si bien a simple vista uno los percibe en la pantalla como si se tratara de un solo plano.

El efecto tridimensional se obtiene haciendo pasar los sprites por delante o por detrás de los demás. Los sprites se numeran de 0 hasta el máximo número disponible que, por ejemplo, en el Sord M5 es 32. Cuando dos sprites se superponen se visualiza el que posee el número más bajo. De manera que si los sprites se ordenan cuidadosamente, se puede obtener con suma facilidad un efecto tridimensional: por ejemplo, si se hace pasar un tren por delante de un árbol, éste se oscurecerá a medida que el tren vaya avanzando. Por otra parte, el mismo tren se puede neutralizar parcialmente si se lo hace pasar por detrás de una casa que posea un número más bajo.

Utilizando la gama de colores que ofrezca el ordenador, cada sprite se puede colorear de forma individual. A veces se puede ampliar o reducir un sprite si se cambia el contenido de una dirección de memoria.

Con todo, es en los programas de juegos donde los sprites demuestran toda su valía; la *detección de colisiones* es una configuración particularmente útil. Cuando se superponen dos o más sprites (por ejemplo, cuando su misil entra en contacto con la nave espacial enemiga), el sistema puede ser programado para que salte hasta otra parte del programa con el fin de crear los gráficos para una explosión.

Antes de utilizar los sprites es preciso crearlos, y el modo de hacerlo se asemeja mucho a la forma en que se diseña un nuevo carácter. Las letras del alfabeto, los números y los símbolos especiales para gráficos están almacenados dentro del ordenador, en un chip que se conoce como *generador de caracteres*.

Los caracteres, como ya hemos mencionado en otro apartado de esta obra, por lo general se construyen en una matriz de puntos (o *pixels*) de ocho por ocho. Las dimensiones máximas de los sprites varían de una máquina a otra, pero generalmente su ancho y su alto corresponden a los de varios caracteres. En el Commodore 64, por ejemplo, la dimensión máxima es 24 pixels de ancho por 21 de alto.

La mejor forma de realizar un sprite consiste en tra-

En diferentes planos
Utilizando los gráficos sprite, se puede crear una compleja imagen al situar los elementos componentes en distintos planos, uno detrás de otro. La inmensa ventaja que ofrece este sistema es que los objetos de los diferentes planos se pueden mover de forma absolutamente independiente respecto a los demás, aunque en la pantalla se perciban como si se tratara de un solo plano. El programador les asigna a los planos un orden de prioridad, de modo que cuando en la pantalla se superponen dos sprites, el que posee la prioridad más alta es el que se ve por delante del otro, obteniéndose así un efecto tridimensional
La mayoría de los ordenadores que disponen de gráficos sprite también pueden disponer que el programa se interrumpa cada vez que entran en contacto dos sprites, para crear la consiguiente explosión o aumentar el puntaje del jugador

Dimensiones del sprite
Sólo cuatro de los microordenadores personales más populares disponen de gráficos sprite: el Commodore 64, el Sord M5, el TI99/4A y la gama Atari.
En el caso del Atari, los 16 colores que normalmente se ofrecen pasan a ser 256, ya que cada color posee 16 grados de "luminosidad".
Cuando se utilizan para juegos, a menudo es interesante saber cuándo dos sprites se tocan (por ejemplo, al chocar dos naves espaciales), y por ello se proporciona una configuración para "detección de colisiones".
El efecto tridimensional se crea colocando cada sprite en un plano diferente; lo mejor para un sistema es que disponga de la mayor cantidad de planos posible. Las dimensiones máximas de cada sprite se expresan en pixels. Los sprites se pueden ampliar o reducir, y desplazar por la pantalla.
Para simplificar el proceso de construcción de los sprites existen paquetes especiales de "programas de utilidad"

MICROORDENADOR	NÚMERO DE COLORES	DETECCIÓN DE COLISIONES	PLANOS	DIMENSIONES DE LOS SPRITES	SOFTWARE
Commodore 64	16	sí	8	24 × 21	sí
Atari 800	256	sí	16	16 × 16	sí
Texas TI99/4A	16	sí	28	32 × 32	sí
Sord M5	16	sí	32	8 × 8	no

zar una cuadrícula que represente las dimensiones máximas, y construir la forma deseada rellenando los cuadrados adecuados. Ya se sabe que los ordenadores trabajan utilizando solamente unos y ceros. Los cuadrados en color se deben representar como unos y los cuadrados en blanco como ceros.

El ordenador maneja la mayor parte de la información en forma de bytes (grupos de ocho bits). En el manual del ordenador se le explicará cómo se debe separar por grupos de ocho la cuadrícula entera de puntos que compone un sprite. Cada byte, a su vez, ha de convertirse en un número decimal, entre 0 y 255 como máximo, antes de que se lo pueda utilizar para un programa en BASIC. Esto se consigue multiplicando el dígito binario (bit) situado más a la izquierda por 128, el siguiente por 64, y así sucesivamente. Luego se suman los resultados.

La serie de números decimales resultante define por completo el diseño del sprite. Estos números se colocan en direcciones de memoria empleando un programa en BASIC; el procedimiento exacto a seguir varía de una máquina a otra. Luego el ordenador requiere que se le proporcionen instrucciones acerca de en qué lugar de la memoria puede hallar las especificaciones para cada uno de los sprites requeridos.

Entonces se puede realizar lo que se desee utilizando órdenes simples para especificar la posición normal de cada sprite en la pantalla, para cambiar su color, agrandarlo o reducirlo y detectar cuándo se superponen dos sprites o más.

Existen paquetes de software, denominados *programas de utilidad*, para la mayoría de las máquinas con sprites. Estos programas logran que el proceso de creación de la imagen resulte menos tedioso. Visualizan la cuadrícula en pantalla a gran escala y permiten dibujar la imagen simplemente moviendo el curso intermitente a través de la cuadrícula. Toda la aritmética se efectúa de forma automática y luego los resultados se colocan en los bytes adecuados. Por último, la cuadrícula desaparece y aparece en pantalla el propio sprite, listo para ser manipulado.

Los sprites han revolucionado los gráficos de los ordenadores personales al proporcionar un método sencillo y eficaz de producir visualizaciones con movimiento rápido y a todo color.

Ha nacido un sprite

Para construir un sprite, lo mejor es comenzar con un trozo de papel cuadriculado. El objeto se dibuja en él rellenando algunos de los cuadrados. Algunos ordenadores pueden realizar sprites multicolores, si bien aquí, por razones de simplicidad, hemos ilustrado un sprite de un solo color. En una segunda cuadrícula, los cuadrados en color se representan como unos y los blancos como ceros. Y como la memoria del ordenador está dividida en bytes (de ocho bits cada uno), toda la cuadrícula se ha de dividir en grupos de ocho cuadrados. Cada grupo se ha de convertir en un número decimal. El bit situado más a la izquierda se multiplica por 128, el siguiente por 64, y así sucesivamente. Estos resultados se suman luego para obtener un resultado que abarca entre 0 y 255. Este se utiliza después en el programa en BASIC

Mark Watkinson

Términos clave

Bien direccionada

La CPU ha de localizar las instrucciones y los datos almacenados en miles de bytes de memoria del ordenador. Le revelamos lo que sucede en la CPU al ejecutar las instrucciones de un programa

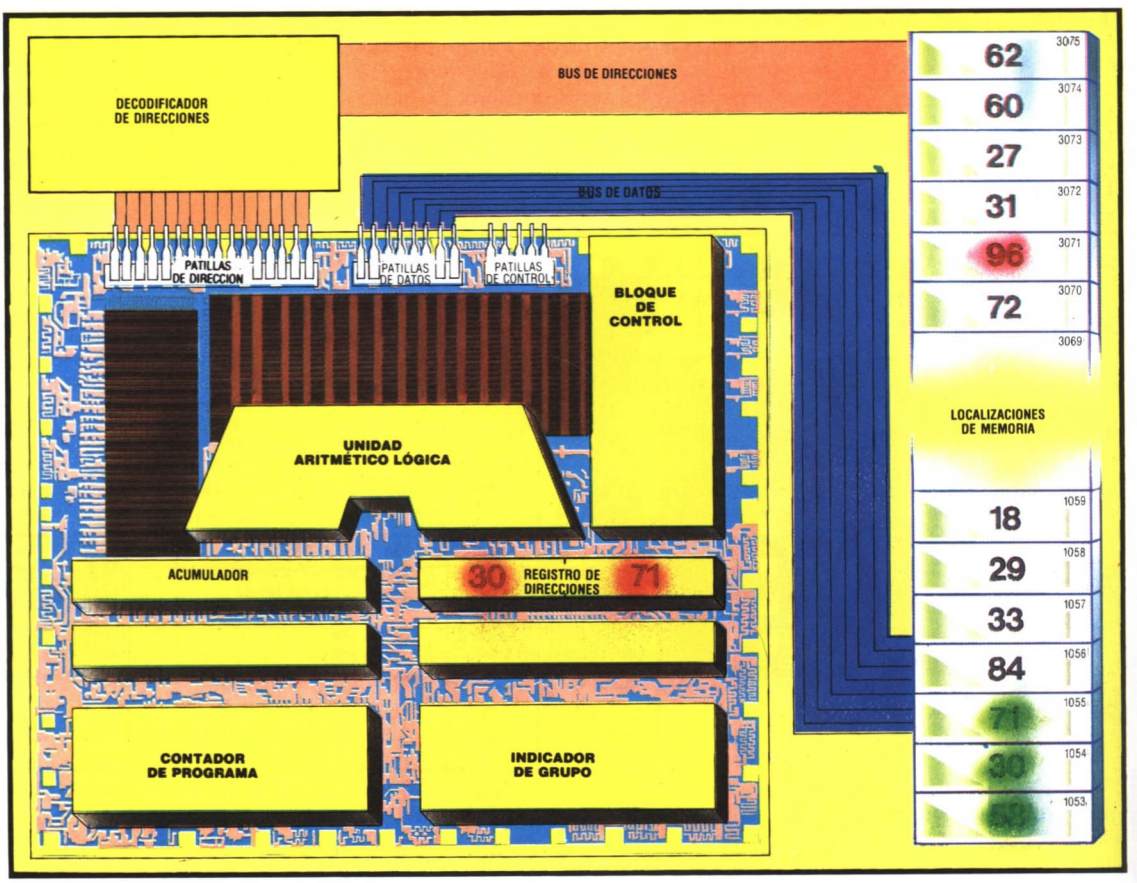
La CPU recibe sus instrucciones y sus datos desde las direcciones situadas en la memoria del ordenador, colocando las patillas de dirección en el código binario requerido por la localización de memoria y leyendo luego el contenido de la localización en la CPU a través del bus de datos. Expuesto de esta manera, el proceso no parece ofrecer mayores dificultades; sin embargo, nada más lejos de la realidad, por cuanto en la práctica esta operación es bastante más complicada.

El problema reside en que los bytes (códigos binarios de ocho bits) de cualquiera de las miles de celdas de memoria del ordenador pueden ser instrucciones que le digan a la CPU que realice algo, o bien información que la CPU ha de manipular de alguna forma. ¿Cómo sabe la CPU cuáles bytes son instrucciones y cuáles son datos?

Reconociendo los códigos

En primer lugar, veamos qué es una "instrucción". Una instrucción es un código, en binario, que hace que en el interior de la CPU se lleve a cabo una secuencia específica de operaciones. De modo que el código 00111010, en caso de que la CPU lo reconozca como una instrucción y no tan sólo como un dato, podría hacer que la CPU direccionara los dos próximos bytes de la memoria, leyera los datos que contienen, colocara esos datos en un "registro de direcciones" especial, situara las patillas de dirección en el mismo número, fuera hasta la localización de memoria recientemente direccionada, a continuación colocara el contenido de dicha localización en el bus de datos y, para finalizar, cargara aquellos contenidos en el acumulador de la CPU.

Todo esto, expresado en palabras, puede resultar confuso; pero lo que acabamos de describir es uno de los procedimientos de direccionamiento de memoria que utiliza la CPU del popular Z80. El gráfico ilustra el proceso completo de tomar un byte de información de la memoria y colocarlo en la CPU. Supongamos que la CPU ya sabe que el próximo byte de la memoria al cual accederá será una instrucción (y no información) y que este byte está en la localización de memoria 1053. (Todos los números utilizados en esta ilustración están en notación decimal.) Esta dirección, la 1053, se colocará en el bus de direcciones. En binario es 0000010000011101. Las 16 patillas de dirección están en posición "encendido" o "apagado" de modo tal que correspondan a este número. Cuando el "decodificador de direcciones" recibe esta dirección que



Cadenas de acontecimientos
Aun la operación más sencilla de la CPU implica muchas etapas. Las instrucciones, también llamadas "códigos de operación", se le leen a la CPU desde la memoria. Estas instrucciones son decodificadas por el bloque de control y hacen que se lleven a cabo operaciones específicas. En este ejemplo, se lee la instrucción 58 de la localización de memoria 1053. Esta instrucción determinada hace que se produzca la siguiente cadena de acontecimientos: el byte de la siguiente localización de memoria (1054) será leído y almacenado en una mitad del registro de direcciones de 16 bits de la CPU. El byte de la localización siguiente (1055) será leído y almacenado en la otra mitad. Ahora estos dos bytes representan la dirección (en algún otro lugar de la memoria) donde se han almacenado unos datos. El contenido del registro de direcciones se coloca en el bus de direcciones, de modo que la próxima localización de memoria a la que se acceda será la dirección 3071. El contenido de esta dirección se pone en el bus de datos y se le lee a la CPU. Este byte (el 96 en nuestro ejemplo) es colocado entonces en el acumulador de la CPU, donde permanecerá hasta que una instrucción posterior lo requiera. El bus de direcciones regresará entonces a su anterior dirección +1, de manera que ahora se dirigirá a la localización 1056. La CPU sabe que, sea lo que fuere lo que contenga esa localización, ha de ser una instrucción, y que se repetirá una secuencia de operaciones similares. En este ejemplo la siguiente instrucción es la 84, que es interpretada por el bloque de control para "complementar" o invertir los bits del acumulador. Dado que 84 es una instrucción "de un byte", la CPU sabe que el byte de la próxima localización de memoria, la 1057, también será una instrucción

viene en el bus de direcciones, la "decodifica" y enciende una, y sólo una, de sus líneas de salida. Ésta es la línea que se encarga de seleccionar la localización de memoria 1053.

La próxima etapa consiste en colocar el contenido de esta dirección, que es 58, o 00111010 en binario, en el bus de datos y "cargarlo" en la CPU. Aquí, como la CPU está esperando una instrucción, el byte es interpretado por el bloque de control y hace que se realice una secuencia muy precisa de operaciones. Esta instrucción en particular especifica que los dos próximos bytes de memoria contendrán 16 bits para ser utilizados como localización de memoria, y que el contenido de esta localización se ha de cargar en el acumulador de la CPU. No bien la CPU reconoce esta instrucción, sabe que los dos próximos bytes de memoria especificarán una dirección y que el contenido de esa dirección se habrá de cargar en el acumulador. En consecuencia, sabe que no recibirá otra instrucción desde la memoria hasta después que se hayan efectuado estas operaciones, y que la próxima instrucción estará en la localización 1056.

La instrucción que estamos utilizando en el ejemplo hace que el bus de direcciones se incremente en 1, de manera que la próxima localización de memoria que se direcciona es la 1054. El contenido de esta localización se coloca entonces en el bus de datos y se carga en la CPU. Esta vez, sin embargo, se coloca en la mitad de un registro de dirección. Después de hacerlo, la CPU vuelve a incrementar el bus de direcciones de modo que ahora direccione la ubicación 1055. El contenido de esta localización se coloca en el bus de datos e, igualmente, se carga en la CPU, si bien en esta ocasión pasa a almacenarse en la otra mitad del registro de direcciones.

Transfiriendo números

La próxima etapa (recuerde que todas estas acciones se producen automáticamente a consecuencia de la instrucción original) consiste en que los números del registro de direcciones se transfieren al bus de direcciones. Estos números, como podemos ver, son 3071. Por lo tanto, la localización de memoria que se está direccionando ahora es la 3071. Esta dirección (en binario 0000101111111111) es decodificada por el decodificador de direcciones y selecciona la celda de memoria 3071. El contenido de esta localización, 96 (en binario 01100000), se coloca en el bus de datos y se carga en la CPU. Esta vez, sin embargo, la información se colocará en el acumulador de la CPU. Después de efectuada esta operación, el bus de direcciones se establecerá en 1056 y allí la CPU esperará encontrar otra instrucción.

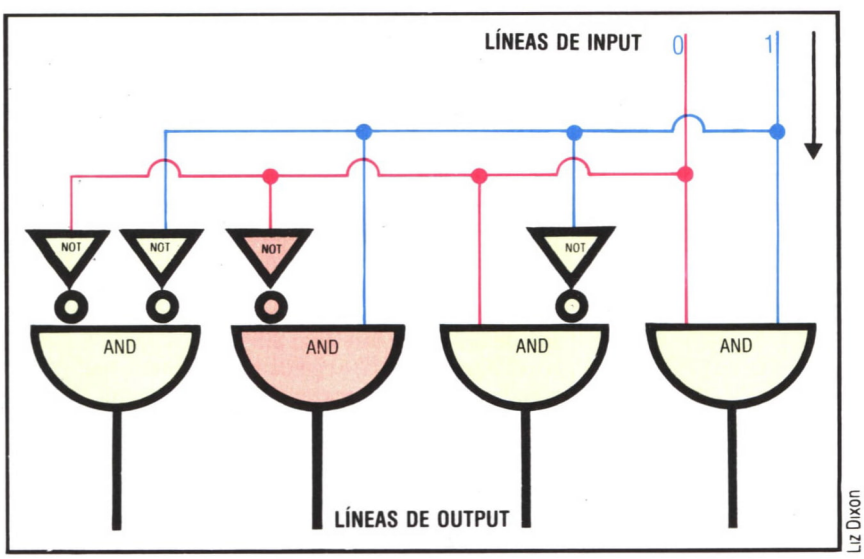
Ahora que la CPU posee en su acumulador una información determinada, ¿qué clase de instrucción se podría esperar a continuación? Podría ser prácticamente cualquiera; las CPU son capaces de reconocer desde docenas hasta centenares de instrucciones, según la CPU. Pero supongamos que lo que deseábamos era invertir los datos del acumulador. Invertir significa cambiar cada uno por un cero y cada cero por un uno. La instrucción para hacerlo estaría localizada en la dirección 1056. En nuestra CPU imaginaria, el código para esta instrucción sería 84. Cuando la CPU recibiera este número, se invertirían los datos del acumulador. El número que estaba en el acumulador era 96 (01100000 en binario). La instrucción para invertir haría que éste se sustituyera por el número binario 10011111. La instrucción para invertir un número del acumulador es una instrucción "de un byte", de modo, pues, que nuevamente la CPU sabría que el contenido de la siguiente localización de memoria, 1057, será otra vez una instrucción y no un dato.

Este procedimiento de direccionar una localización de memoria para recuperar un dato sólo es uno de los diversos métodos de que dispone el programador. Los bytes de instrucción específicos que hemos utilizado en el ejemplo (58 para cargar el acumulador y 84 para invertir los contenidos del acumulador) son instrucciones para nuestra CPU hipotética, pero el mismo principio se aplica para todos los otros chips de microprocesadores. La única diferencia estriba en que se emplean códigos distintos para las diversas instrucciones y que cada versión de CPU posee su propio "juego de instrucciones", que difiere ligeramente del de las demás.

Las localizaciones de I/O (Input/Output) también deben poseer direcciones exclusivas, pero los principios en virtud de los cuales la CPU las direcciona son los mismos. Por lo general, en los microprocesadores de ocho bits sólo ocho de las líneas de dirección están disponibles para el direccionamiento de I/O, de manera que el número máximo de direcciones de I/O es 256. No obstante, esto es más que suficiente para la mayoría de las aplicaciones de los ordenadores pequeños.

La decodificación de direcciones siempre es necesaria, de manera que el dispositivo seleccionado por la CPU (ya sea una localización de memoria o una localización de I/O) está especialmente activo cuando todas las demás localizaciones de memoria o de I/O están inactivas. Este proceso se denomina *activación*. Cuando hay que decodificar una pequeña cantidad de líneas de dirección, se pueden utilizar chips de puertas lógicas simples para que realicen la decodificación. El gráfico ilustra el principio del decodificador de línea de dos a cuatro. Este tipo de decodificación sencilla se emplea con frecuencia para seleccionar dispositivos de I/O. Sin embargo, a medida que va aumentando el número de líneas de dirección, la complejidad del circuito de decodificación se incrementa de manera considerable. Cuando es preciso proceder a seleccionar individual y exclusivamente 65 536 localizaciones de memoria separadas, lo normal es que la mayor parte de la decodificación de direcciones se lleve a cabo en el interior de los chips de memoria.

Decodificación de direcciones
Las 16 líneas que constituyen el bus de direcciones son capaces de identificar de modo exclusivo cualquiera de las 65 636 localizaciones de memoria separadas. La combinación de unos y ceros del bus de direcciones se decodifica en los decodificadores de direcciones. Parte de la decodificación la efectúan los decodificadores de direcciones, compuestos por puertas lógicas simples en chips montados en el circuito; gran parte de la decodificación la efectúan circuitos equivalentes situados dentro de los propios chips de memoria. La ilustración muestra cómo se pueden decodificar dos líneas de dirección para seleccionar uno, y sólo uno, de cuatro chips



Liz Dixon

Totalmente funcional

El BASIC posee funciones incorporadas; ello significa que gran parte de la programación ya ha sido hecha para usted. El saber utilizarlas le permitirá disponer de un mayor potencial informático

Supongamos que en uno de sus programas desea averiguar la raíz cuadrada de un número. La puede calcular de diversas maneras. La forma más elemental y menos satisfactoria sería crear una tabla de valores de raíz cuadrada y utilizarla para obtener el valor deseado para un número determinado. Es probable que haya aprendido en la escuela la forma de confeccionar una tabla de este tipo. Un método alternativo consiste en utilizar la "función" raíz cuadrada, que incorporan la mayoría de las versiones de BASIC. Aquí es este lenguaje el que se encarga de la aritmética de la operación, sin que el programador deba preocuparse por ella. Veamos cómo funciona:

```
10 REM ESTE PROGRAMA BUSCA LA RAIZ
   CUADRADA
20 REM DE UN NUMERO
30 PRINT "DE ENTRADA AL NUMERO CUYA RAIZ
   CUADRADA"
40 PRINT "DESEA HALLAR"
50 INPUT N
60 LET A = SQR(N)
70 PRINT "LA RAIZ CUADRADA DE"; N; "ES"; A
80 END
```

Digite este corto programa y verá que, en efecto, le proporcionará la raíz cuadrada de cualquier número que digite. Estudiemos las reglas relativas a la forma de utilizar esta función de "raíz cuadrada".

Una "función" en BASIC es generalmente una palabra de orden (en este caso SQR, abreviatura de *square root*: raíz cuadrada) seguida de paréntesis que encierran la expresión a operar. En este programa, N es la entrada del número desde el teclado. Es el número cuya raíz cuadrada deseamos hallar. La línea 60 dice "asignemos la raíz cuadrada de N a la variable A". La línea 70 imprime el valor de A.

La expresión entre paréntesis se denomina *argumento* de la función y no siempre ha de ser una variable: es igualmente posible utilizar números reales. Digite lo siguiente y vea qué sucede cuando lo ejecuta:

```
10 PRINT SQR(25)
20 END
```

Comprobará que esto funciona exactamente igual. De la misma manera, podemos incluir entre los paréntesis argumentos más complicados. Pruebe con este ejemplo:

```
10 LET A = 10
20 LET B = 90
30 LET C = SQR(A + B)
40 PRINT C
50 END
```

Este pequeño programa se puede acortar combinando las líneas 30 y 40 de la siguiente manera:

```
10 LET A = 10
20 LET B = 90
30 PRINT SQR(A + B)
40 END
```

Las funciones se deben considerar como programas cortos incorporados al BASIC, disponibles para que el programador los emplee en cualquier momento. La mayoría de las versiones de BASIC ofrecen una cantidad bastante importante de funciones, así como la posibilidad de que el programador defina algunas nuevas para utilizarlas dentro de un programa. Más adelante veremos la forma de hacerlo. Ahora estudiaremos algunas de las otras funciones disponibles comúnmente. Éstas presentan dos variedades: las funciones numéricas, en las cuales el argumento (la parte encerrada entre paréntesis) es un número, una variable numérica o una expresión numérica, y las funciones en serie, en las cuales el argumento es una serie de caracteres o una expresión alfanumérica. Veamos primero algunas de las funciones numéricas.

Con anterioridad ejecutamos un programa que calculaba el número de azulejos que se necesitaban para revestir las paredes de una habitación. Un pequeño *bug* de este programa consistía en que la respuesta no podía incluir fracciones decimales de un azulejo. La ejecución de este programa podría dar 988,24 como posible resultado. En ocasiones como ésta deseamos disponer de algún procedimiento para redondear la respuesta en el número completo más próximo. En matemáticas, los números completos se denominan *enteros* y una de las funciones del BASIC es "devolver" la parte entera de cualquier número. Funciona así:

```
10 PRINT "DE ENTRADA A UN NUMERO QUE
   CONTENGA FRACCION DECIMAL"
20 INPUT N
30 PRINT "LA PARTE ENTERA DEL NUMERO ES";
40 PRINT INT(N)
50 END
```

Si ejecuta este programa y da entrada a 3,14, el programa imprimirá en pantalla:

```
LA PARTE ENTERA DEL NUMERO ES 3
```

Por supuesto, si se trata de azulejos necesitaríamos luego sumarle 1 a la respuesta, para asegurarnos de comprar más cantidad de la requerida y no menos.

En otra ocasión podríamos desear averiguar el "signo" de un número para ver si es negativo, cero o positivo. Para ello, la mayoría de las versiones de BASIC incorporan una función SGN. Ensayemos con el siguiente ejemplo:

```
10 PRINT "DE ENTRADA A UN NUMERO"
20 INPUT N
```

```
30 LET S = SGN(N)
40 IF S = -1 THEN GOTO 100
50 IF S = 0 THEN GOTO 120
60 IF S = 1 THEN GOTO 140
100 PRINT "EL NUMERO ERA NEGATIVO"
110 GOTO 999
120 PRINT "EL NUMERO ERA CERO"
130 GOTO 999
140 PRINT "EL NUMERO ERA POSITIVO"
150 GOTO 999
999 END
```

Si observa los valores "devueltos" a S en la línea 30 por la función SGN (éstos se comparan en las líneas 40, 50 y 60), verá que los valores son tres. Se devuelve −1 si el argumento entre paréntesis era un número negativo, 0 si el argumento era cero y 1 si el argumento era un número positivo. La utilización de la función SGN en la línea 30 ahorra varias líneas de programación. Podríamos haber escrito:

```
IF N < 0 THEN LET S = -1
IF N = 0 THEN LET S = 0
IF N > 0 THEN LET S = 1
```

La acción que se consigue realizar mediante el empleo de una función de BASIC siempre se puede obtener mediante la programación normal, pero el empleo de una función supone un ahorro de tiempo, de espacio y de esfuerzo de programación.

A continuación reseñamos unas pocas funciones numéricas más. ABS devuelve el valor "absoluto" de un número. El valor absoluto de un número es el de su valor real pero sin su signo. Por lo tanto, el valor absoluto de −6 es 6. Probemos:

```
10 LET X = -9
20 LET Y = ABS(X)
30 PRINT Y
40 END
```

MAX busca el valor máximo de dos números. Por ejemplo:

```
10 LET X = 9
20 LET Y = 7
30 LET Z = X MAX Y
40 PRINT Z
50 END
```

MIN es similar a MAX, pero averigua el valor más pequeño de dos números. Veamos el proceso:

```
10 PRINT "DE ENTRADA A UN NUMERO"
20 INPUT X
30 PRINT "DE ENTRADA A OTRO NUMERO"
40 INPUT Y
50 LET Z = X MIN Y
60 PRINT Z
70 END
```

Observe que estas dos últimas funciones poseen dos argumentos en lugar de uno, y que no es necesario encerrarlos entre paréntesis. La mayoría de las versiones de BASIC poseen también algunas otras funciones numéricas, incluyendo LOG para encontrar el logaritmo de un número, TAN para descubrir la tangente, COS para encontrar el coseno y SIN para hallar el seno. Más adelante veremos algunas de las maneras en que se pueden utilizar estas funciones "trigonométricas".

El BASIC posee, asimismo, varias funciones incorporadas que operan con variables alfanuméricas. En nuestro programa para seleccionar un nombre empleamos algunas de ellas, pero en aquella ocasión no estudiamos con detención la forma en que funcionaban. Ahora veremos con mayor detalle aquellas funciones en serie y algunas más.

Una de las funciones en serie más útiles es LEN. Ésta cuenta el número de caracteres de una serie de ellos encerrada entre comillas dobles o el número de caracteres asignados a una variable alfanumérica. Probemos con el siguiente ejemplo:

```
10 LET A$ = "COMPUTER"
20 LET N = LEN(A$)
30 PRINT "EL NUMERO DE CARACTERES DE LA
   VARIABLE ES"; N
40 END
```

¿Por qué necesitaríamos saber, alguna vez, de cuántos caracteres se compone una variable alfanumérica? Para ver por qué, dé entrada y ejecute este breve programa creado para construir un "triángulo de nombre". En primer lugar imprimirá la primera letra de una palabra, luego la primera y la segunda, después la primera, la segunda y la tercera, y así sucesivamente hasta imprimir la palabra completa.

```
5 REM IMPRIME UN 'TRIANGULO DE NOMBRE'
10 LET A$ = "JONES"
20 FOR L = 1 TO 5
30 LET B$ = LEFT$(A$,L)
40 PRINT B$
50 NEXT L
60 END
```

Ahora ejecute este programa. ¿Se imagina qué es lo que imprimirá? Se leerá algo similar a esto:

```
J
JO
JON
JONE
JONES
```

Este corto programa utiliza la función LEFT$ para extraer los caracteres de una variable. LEFT$ lleva dos argumentos. El primero especifica la variable y el segundo (que viene precedido por una coma) especifica el número de caracteres a extraer de la variable, empezando desde el lado izquierdo de ésta. A A$ se le había asignado la serie "JONES", de modo que LEFT$(A$,1) "devolvería" la letra J. LEFT$(A$,2) devolvería las letras JO. El breve programa anterior emplea un índice, L, que abarca desde 1 hasta 5, de manera que el segundo argumento de la función LEFT$ asciende desde 1 hasta 5 cada vez que se efectúa el bucle. Sabemos exactamente cuántos caracteres había en la palabra que deseábamos imprimir (JONES), por lo cual fue sencillo decidir que 5 sería el límite máximo del bucle FOR-NEXT. Pero ¿qué haríamos si no supiéramos de antemano cuántos caracteres habrá de tener el bucle?

Aquí entra en juego la función LEN. Esta función toma como argumento una variable (entre comillas dobles) o una variable alfanumérica. A continuación le proporcionamos algunos ejemplos para que vea cómo funciona LEN:

```
10 REM PROGRAMA PARA COMPROBAR LA
   FUNCION 'LEN'
20 PRINT LEN ("COMPUTER")
30 END
```

Al ser ejecutado, este programa imprimirá 8. Ha contado el número de caracteres que componen la palabra COMPUTER y ha devuelto este valor. Hagamos lo mismo pero de un modo ligeramente diferente:

```
10 REM BUSCANDO LA LONGITUD DE UNA VARIABLE
20 LET A$ = "MI COMPUTER"
30 LET L = LEN(A$)
40 PRINT L
50 END
```

Este programa, al ser ejecutado, habría de imprimir en pantalla 11. En esta variable hay 11 caracteres y no 10. Recuerde que, en lo que atañe al ordenador, el espacio que separa una palabra de otra es también un carácter. Ahora apliquemos la función LEN en una modificación de nuestro programa anterior para imprimir un "nombre triangular":

```
10 REM ESTE PROGRAMA IMPRIME UN 'TRIANGULO
   DE NOMBRE'
20 PRINT "DIGITE UN NOMBRE"
30 INPUT A$
40 LET N = LEN(A$)
50 FOR L = 1 TO N
60 LET B$ = LEFT$(A$,L)
70 PRINT B$
80 NEXT L
90 END
```

Cada vez que se ejecute este bucle, el valor de L se incrementará desde 1 hasta N (que es la longitud del nombre en la variable). Si ahora se da entrada al apellido SIMPSON, la línea 40 equivaldrá a LET N = LEN ("SIMPSON"), de manera que N se establecerá en 7. La primera vez que se efectúe el bucle, la línea 50 establecerá L en 1 y la línea 60 equivaldrá a LET B$ = LEFT$ ("SIMPSON", 1), de modo que a B$ se le asignará un carácter de la variable, empezando por la izquierda. Este carácter es S.

La segunda vez que se efectúe el bucle, L se establecerá en 2, de modo que la línea 60 equivaldrá a LET B$ = LEFT$ ("SIMPSON", 2). Ésta tomará los dos primeros caracteres de la variable y los asignará a la variable alfanumérica B$. Por lo tanto, B$ contendrá SI.

La función LEN descubrió que los caracteres de la variable SIMPSON eran 7 y le asignó este valor a la variable N, de manera que la última vez que se efectúe el bucle a B$ se le asignarán los 7 caracteres de la variable y se imprimirá la variable completa.

LEFT$ posee una función compañera, RIGHT$, que toma los caracteres de la variable alfanumérica exactamente de la misma forma pero comenzando por la derecha.

Por último, estudiaremos otra función alfanumérica, que también empleábamos en nuestro programa de clasificación de nombres. Se trata de INSTR y se utiliza para encontrar la localización de la primera aparición de una variable específica (denominada *subvariable*) dentro de una variable. En el programa de clasificación de nombres se utilizaba INSTR para localizar la posición del espacio entre el nombre de pila y el apellido. Funciona de la siguiente manera:

```
10 LET A$ = "GUARDABOSQUE"
20 LET P = INSTR(A$, "BOSQUE")
30 PRINT P
40 END
```

Antes de dar entrada a este programa y de ejecutarlo, intente predecir el valor que se imprimirá para P. Recuerde que INSTR localiza la posición inicial de la primera vez que se produce la subvariable dentro de la variable. Si la variable es GUARDABOSQUE, la posición inicial de la subvariable BOSQUE será 7, pues la B de BOSQUE es la séptima letra de GUARDABOSQUE. Algunas versiones de BASIC carecen de la función INSTR, pero poseen una función similar denominada INDEX. Ésta es la forma de utilizar INSTR (o INDEX) para localizar un espacio dentro de una variable:

```
10 REM BUSCANDO LA POSICION DE UN ESPACIO EN
   UNA VARIABLE
20 LET A$ = "ORDENADOR PERSONAL"
30 LET P = INSTR(A$, " ")
40 PRINT P
50 END
```

Observe que el segundo argumento de la función INSTR (línea 30) es " ". Las comillas encierran un espacio: el carácter que se ha de buscar. El programa imprimirá 10 como el valor de P, dado que el espacio se halla en el décimo lugar de la variable. Intente deducir qué se imprimiría si se modificara la línea 30 del siguiente modo:

```
LET P = INSTR(A$, "C")
```

Por último, una función muy práctica que se utiliza con la sentencia PRINT. Veamos lo que sucede al ejecutar este programa:

```
10 PRINT "ESTA LINEA NO ESTA SANGRADA"
20 PRINT TAB(5); "ESTA LINEA ESTA SANGRADA"
30 END
```

¿Puede ver lo que sucedió? La segunda línea se empezó a imprimir cinco espacios a la derecha del margen izquierdo. TAB es análoga al tabulador de una máquina de escribir. Aquí le ofrecemos otro breve programa que emplea la función TAB:

```
10 REM UTILIZANDO LA FUNCION TAB
20 PRINT "INTRODUZCA EL VALOR DE TAB"
30 INPUT T
40 LET W$ = "TABULACION"
50 PRINT TAB(T); W$
60 END
```

Ahora puede volver al programa de clasificación de nombres y comprobar cómo se emplean en él algunas de estas funciones.

Ejercicios

■ **Bucles 1** ¿Qué se imprimirá al ejecutar este programa?

```
10 LET A = 500
20 FOR L = 1 TO 50
30 LET A = A -1
40 NEXT L
50 PRINT "EL VALOR DE A ES"; A
```

Complementos al BASIC

En el Spectrum reemplazar por el número de TAB.

El Spectrum no dispone de ninguna de estas órdenes, pero usted puede crear sus propias versiones de las mismas mediante la sentencia DEF FN; de modo que agregue a su programa las siguientes líneas:

```
9900 DEF FN L$(X$,N) = X$(TO N)
9910 DEF FN R$(X$,N) =
   = X$(LEN(X$) - N + 1 TO)
9920 DEF FN M$(X$,P,N) = X$(P TO
   P + N - 1)
```

Ahora


```
FN L$(X$,N) sustituye a LEFT$(X$,N)
FN M$(X$,P,N) sustituye a MID$(X$,P,N)
FN R$(X$,N) sustituye a RIGHT$(X$,N)
```

No disponen de esta función el Spectrum, el VIC-20, el Commodore 64 y el Oric-1, pero es posible sustituirla. Supongamos que la línea original fuera:

```
20 LET P = INSTR(A$,"BOSQUE")
```

Reemplácela por

```
20 LET X$ = A$: LET Z$ = "BOSQUE"
   GOSUB 9930: P = U
```

y agregue estas líneas:

```
9929 STOP
9930 LET U = 0: LET X = LEN(X$):
   LET V = LEN(Z$)
9940 FOR W = 1 TO L - V + 1: IF
   MID$(X$,W,V) = Z$ THEN LET U = W
9950 IF U <> 0 THEN LET W =
   = L - V + 1
9960 NEXT W: RETURN
```

En el Spectrum, sustituya MID$(X$,W,V), en la línea 9940, por FN M$(X$,W,V) y consulte el "Complementos al BASIC" anterior

■ **Bucles 2** ¿Qué se verá en pantalla si se ejecuta este programa?

```
10 REM
20 REM ESTE ES UN BUCLE DE TIEMPOS
30 REM COMPROBAR CUANTO DURA
40 REM
50 PRINT "START"
60 FOR X = 1 TO 5000
70 NEXT X
80 PRINT "STOP"
90 END
```

■ **Bucles 3** ¿Qué resultado se imprimirá si ejecuta este programa y, al requerírsele, digita el número 60?

```
10 PRINT "PIENSE UN NUMERO Y DIGITELO"
20 INPUT N
30 LET A = 100
40 FOR L = 1 TO N
50 LET A = A + 1
60 NEXT L
70 PRINT "AHORA EL VALOR DE A ES"; A
80 END
```

■ **Bucles 4** ¿Qué sucedería al ejecutar este programa?

```
10 PRINT "ME GUSTA EL BASIC"
20 GOTO 10
30 END
```

■ **Bucles 5** ¿Qué se vería en pantalla si se ejecutara este programa?

```
10 FOR Q = 1 TO 15
20 PRINT "ESTOY ALGO TONTO"
30 NEXT Q
40 END
```

■ **Read-Data 1** ¿Qué resultado se imprimiría?

```
10 READ X
20 READ Y
30 READ Z
40 PRINT "COMPARAMOS LA SENTENCIA 'READ'"
50 DATA 50,100,20
60 PRINT X + Y + Z
```

■ **Read-Data 2** ¿Qué se imprimiría en pantalla si se ejecutara este programa?

```
100 FOR L = 1 TO 10
110 READ X
120 PRINT "X = "; X
130 NEXT L
140 DATA 1,2,3,5,7,11,13,17,19,23
```

Las respuestas, en el próximo capítulo.

Respuestas a los "Ejercicios" de las páginas 101 y 102
Variables

A (B6) ~~X~~ D$ ~~Y~~ X$ (A12) (D9) (Q81) (Q5) ~~G~~ H$

Aritmética 1

```
10 LET B = 6
20 PRINT B
```

Aritmética 2

```
10 LET A = 5
20 LET B = 7
30 LET C = 9
40 LET D = A + B + C
50 PRINT D
```

Aritmética 3
17

Aritmética 4
25
25

Comparaciones 1
5

Comparaciones 2
601 (los enteros se suponen)

Comparaciones 3
10000

Print 1

```
PRINT "EL VALOR DE T ES";T
```

Print 2

```
640 PRINT "LO SIENTO, PERO SU PUNTUACION DE";S; "ES
      DEMASIADO BAJA"
```

Print 3
Se trataba de un error deliberado. El punto y coma al final de la línea producirá un error de sintaxis cuando se ejecute. El programa debe decir:

```
200 LET A$ = "¿MI COMPUTER?"
210 LET B$ = "¿LE GUSTA?"
220 PRINT B$;A$
```

Y entonces el resultado sería:
¿LE GUSTA MI COMPUTER?

Entrada 1
6

Entrada 2
POR FAVOR DIGITE SU NOMBRE
HOLA (SU NOMBRE) SOY SU ORDENADOR

Tenga en cuenta que las respuestas a las "variables" serán diferentes en algunas máquinas que no admiten más de un carácter alfabético (es decir, que no admiten sufijo numérico)

Varillas mágicas

Un lápiz óptico es un accesorio que sirve para dibujar o marcar en la pantalla. Vamos a explicar cómo funcionan estos extraordinarios dispositivos

La "aversión a los teclados" es una de las causas más comunes por las que la gente se muestra reacia a familiarizarse con los ordenadores, ya sea en su casa o en la oficina. Debido a que el teclado se asemeja al de una máquina de escribir y a que estas personas no saben mecanografía y, más aún, comprueban que el teclado contiene diversas teclas con signos desconocidos, temen hacer el ridículo. La solución a este problema es el lápiz óptico (junto a otros dispositivos tales como entrada de voz), que posee asimismo otras funciones.

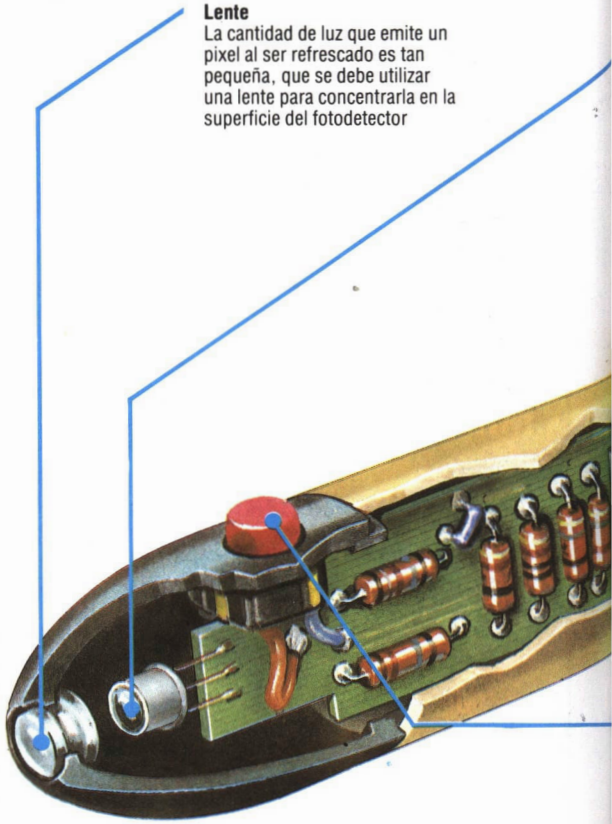
Un lápiz óptico es un dispositivo cilíndrico (su nombre deriva de su gran parecido con un lapicero corriente) de uno de cuyos extremos sale un cable en espiral similar al de los teléfonos. En el otro extremo de este cable hay un enchufe que se conecta en la parte posterior del ordenador. Cuando se apunta directamente a la pantalla con el lápiz óptico (en algunos ordenadores el lápiz ha de tocar ésta para activar un interruptor situado en su interior), el ordenador detecta la posición exacta que está señalando el lápiz.

Lo que sucede en realidad es que un fotodetector localizado en la punta del lápiz responde con un impulso eléctrico cuando pasa por el punto de luz que continuamente explora toda la pantalla para crear la imagen. El sistema de circuitos en el interior del chip controlador del video calcula dónde estaba el punto de exploración en el momento en que el lápiz óptico emitiera su señal.

El lápiz óptico, por lo tanto, se utiliza básicamente para seleccionar un ítem visualizado en pantalla. Sabiendo qué punto está señalando el lápiz, el ordenador deduce el carácter o el símbolo del que forma parte ese punto. Muchos programas de aplicaciones incorporan "menús" en su funcionamiento. Un menú es simplemente una lista de opciones que se visualiza en la pantalla, entre las cuales el usuario debe hacer su elección, como haría con el menú de un restaurante. En un programa de economía doméstica, el menú sería:

1) Efectuar un pago
2) Examinar el estado de cuentas bancario
3) Aceptar un recibo
   etcétera.

Normalmente el usuario debería indicar la acción que ha escogido pulsando una tecla (1, 2 o 3) o digitando una palabra de orden. Con un lápiz óptico, todo cuanto ha de hacer será señalar la opción requerida. El ordenador por lo general responde haciendo centellear esa opción determinada, para indicar que ha aceptado la entrada. Algunos programas más sofisticados funcionan casi completamente mediante este tipo de menús (se los suele describir como programas de opciones de trabajo), en cuyo caso el usuario sólo toca el teclado cuando se requiere una información verdadera, como por ejemplo el nombre y la dirección de alguien.



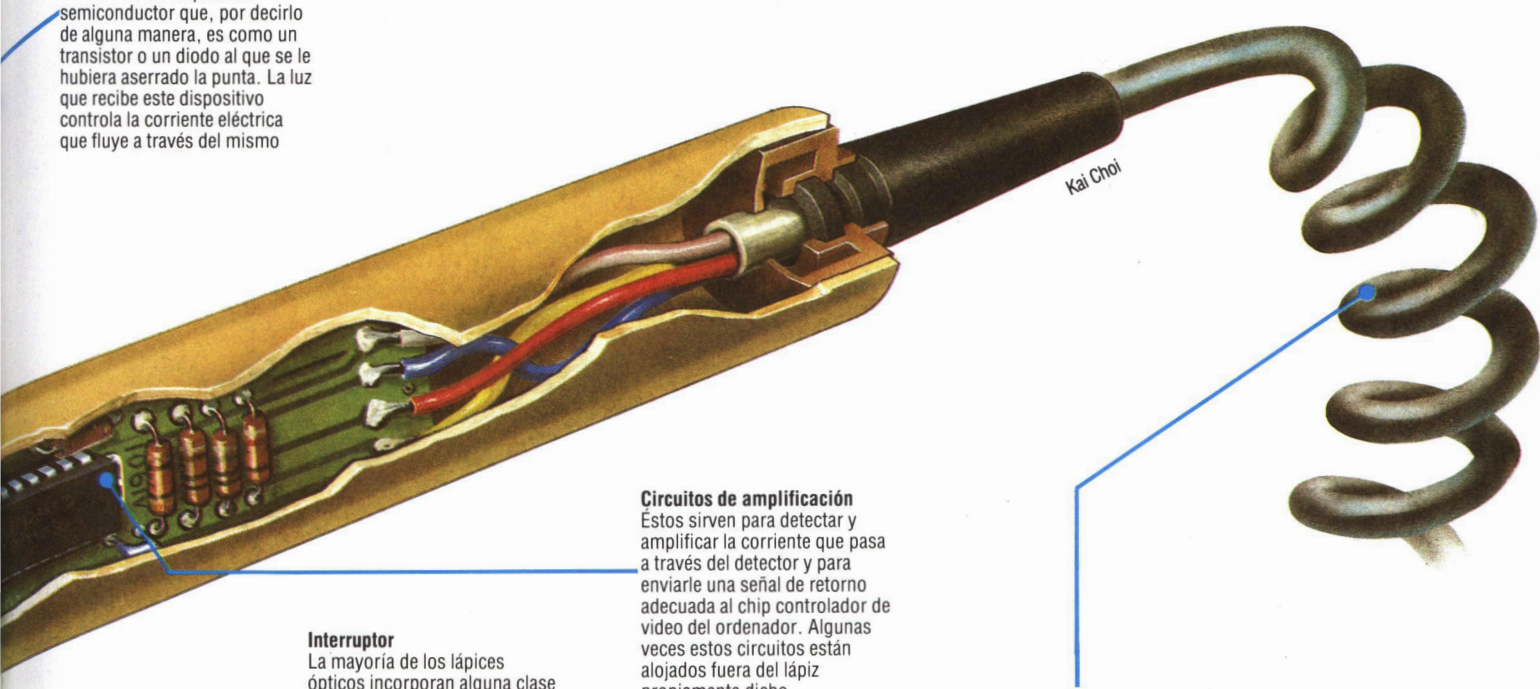
Rutinas especiales

La dificultad que suponen los programas de este tipo es que se deben escribir pensándolos especialmente para que trabajen con el lápiz óptico y no desde el teclado. En realidad la cuestión estriba en escribir una pequeña rutina que asuma las coordenadas que indican la posición normal del lápiz óptico desde el controlador de video y que determine cuál de las opciones se produce en la pantalla en esa posición. Lamentablemente, son pocos los fabricantes de software que proporcionan versiones de sus programas que puedan ejecutarse con lápiz óptico.

Sin embargo, además de utilizarse para seleccionar ítems, los lápices ópticos se pueden emplear para crear imágenes en pantalla. La mayoría de los ordenadores personales para los cuales existen lápices ópticos poseen un paquete de programas apto para tal fin. Al usuario se le presenta una pantalla en blanco (sobre la cual él puede dibujar un diagrama, un dibujo o un esbozo) y una sección separada (normalmente a lo largo de la parte inferior de la pantalla) que le proporciona una serie de funciones especiales para ayudarlo en el proceso de creación. Una de éstas puede ser una paleta de colores, que funciona igual que una paleta de óleos como la que emplean los pintores. El lápiz óptico se coloca sobre el próximo color requerido, y el lugar por donde después se pase el lápiz por la pantalla principal dejará una línea de ese color.

De la parte inferior de la pantalla, utilizada como paleta, el usuario también puede seleccionar distintas

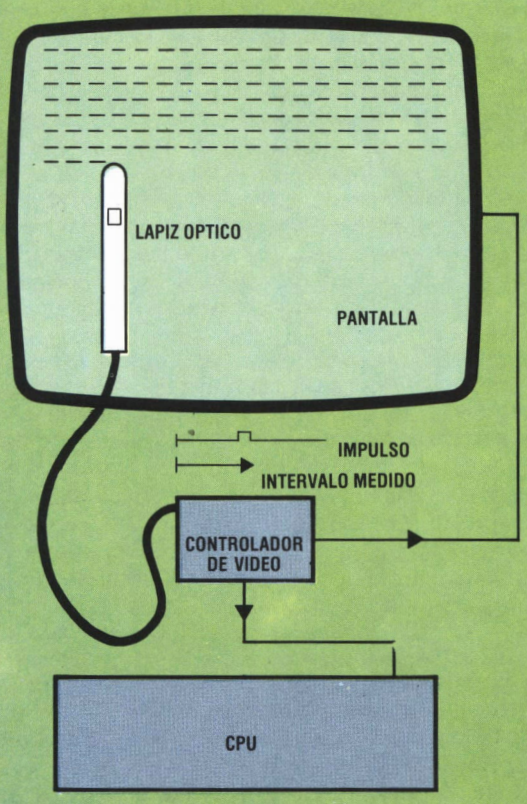
Fotodetector
Se trata de un dispositivo semiconductor que, por decirlo de alguna manera, es como un transistor o un diodo al que se le hubiera aserrado la punta. La luz que recibe este dispositivo controla la corriente eléctrica que fluye a través del mismo



Circuitos de amplificación
Éstos sirven para detectar y amplificar la corriente que pasa a través del detector y para enviarle una señal de retorno adecuada al chip controlador de video del ordenador. Algunas veces estos circuitos están alojados fuera del lápiz propiamente dicho

Interruptor
La mayoría de los lápices ópticos incorporan alguna clase de interruptor, a veces para operar manualmente, pulsándolo con un dedo, y otras veces para activar presionando el lápiz óptico contra la pantalla. El interruptor es necesario para que el lápiz no reaccione a la luz (por ejemplo, a la luz de la habitación) cuando no se lo está utilizando para seleccionar un ítem en la pantalla

Cable
Este cable en espiral al estilo de los del teléfono se enchufa directamente en la parte posterior del ordenador y desde allí conecta con el chip controlador de video



Liz Dixon

Cómo funciona
El funcionamiento del lápiz óptico suele depender del chip controlador de video situado en el interior del ordenador. Este chip, explorando la pantalla muchas veces por segundo con líneas horizontales, produce imágenes de video en el televisor o en el monitor (esta acción se denomina *raster scan*). Los televisores españoles tienen 625 líneas en la pantalla y la frecuencia de exploración es de 50 por segundo. La salida del lápiz óptico también se deriva directamente a este chip. Cuando se señala un ítem de la pantalla con el lápiz óptico y se activa el interruptor, el controlador de video se limita a medir el tiempo que media desde la última vez que comenzó a trazar las líneas en la parte superior de la pantalla y el instante en que recibió el impulso proveniente del lápiz óptico. A partir de esta información calcula sobre qué línea está situado el lápiz y a qué distancia se halla esta línea. Esta información se convierte luego en las coordenadas X e Y que necesita el programa

anchuras de trazo y texturas y, asimismo, puede dibujar círculos y cuadrados. En resumen, todo cuanto exponíamos en "El artista electrónico" se puede realizar con un lápiz óptico y, probablemente, mucho más rápido que mediante el teclado.

Asimismo, están comenzando a aparecer en el mercado juegos que emplean lápices ópticos. Capturar monstruos extraterrestres es mucho más fácil con un lápiz óptico que con el teclado, de manera que los juegos han de incorporar otros grados de dificultad. Otros juegos más sosegados, como el ajedrez informatizado, también se pueden beneficiar de la utilización del lápiz óptico: uno señala adónde quiere desplazar el caballo y el ordenador se ocupa de todo lo demás.

Quizá el grupo más numeroso de usuarios de lápices ópticos sea, no obstante, el de ingenieros y diseñadores. Los sistemas de CAD (Computer Aided Design: diseño auxiliado por ordenador) se emplean bastante en los anuncios publicitarios de nuevos modelos de coches; en realidad, son sistemas informatizados como tantos otros, pero con un software especializado y gráficos de gran calidad. Si se utiliza un sistema CAD para diseñar un nuevo dispositivo electrónico, la pantalla incorporará la representación de todos los componentes a los que el diseñador pueda desear acceder; éste puede "escoger" cualquiera de ellos y colocarlo en el lugar de la pantalla que crea conveniente.

El lápiz óptico constituye uno de los mejores ejemplos de accesorio para ordenador que, además de poseer gran valor práctico, resulta fácil y divertido de emplear.

Planificando el futuro

	A	B		TOTALES
1	ENERO	FEBRERO	1	
2	42,41	18,75	2	388,4
3	160,35	149,89	3	1732,7
4			4	

Campos de juego
Una hoja electrónica está dividida en líneas y columnas; la intersección de una línea con una columna se denomina *campo* o *celda* y se puede direccionar a partir de sus coordenadas (A1, B3, etc.). Cada campo puede contener un título (p. ej., ENERO), un número (p. ej., 149,89) o una fórmula. El campo N2 contiene la fórmula 'SUM(A2: M2)', que es la suma de las cifras de la línea superior, de enero a diciembre. El resultado de este cálculo está visualizado: 388,4. Observe que se ha dividido la hoja electrónica en dos ventanas, de manera que las cifras de marzo-diciembre no aparecen visibles

Un programa de "hoja electrónica" puede ayudar a darle a su ordenador una utilidad lucrativa. Sus principales aplicaciones son planificación, presupuesto y previsión financiera

Se ha calculado que los gerentes dedican aproximadamente el 30 % de su tiempo a la elaboración de presupuestos, una actividad que siempre implica considerar las respuestas a muchas preguntas del tipo "¿qué ocurriría si...?" Tradicionalmente se utilizaba para ello una hoja de papel, por lo general equivalente al tamaño de una doble página de *Mi Computer*. En ella se trazaban verticalmente una docena o más de columnas, cada una de las cuales se encabezaba con el número de un mes, y todos los tipos de gastos uno debajo del otro sobre un costado. En cada columna se incluían los gastos mensuales en las diversas categorías. Siguiendo este método, uno podía sumar cada columna para obtener los gastos totales mes por mes, y sumar individualmente todas las líneas para hallar el total gastado durante un año entero en cada rubro.

El problema se planteaba cuando uno había planificado gastar demasiado (o, peor aún, muy poco) y debía volver atrás y alterar una gran cantidad de cifras, y volver a calcular en función de ellas los totales de las líneas y las columnas.

Utilizando un programa de hoja electrónica, uno puede volver a calcular la página entera de cifras cada vez que se altera un único elemento básico. Si, por ejemplo, ha variado el costo del transporte del mes de enero, esta variación alterará los gastos totales de ese mes; así como el total de los gastos correspondientes a ese rubro, con sólo tocar una tecla. ¡No hay que sorprenderse de que los paquetes de hojas electrónicas sean el tipo de programas que más se venden en todo el mundo!

Al igual que la mayoría de los programas comerciales, las hojas electrónicas por lo general se escriben con *overlays*, es decir, que en realidad no todo el programa reside realmente en la máquina todo el tiempo. Si se considera que el programa está dividido en subrutinas, la subrutina que no se requiera en la operación normal no será recuperada del almacenamiento de apoyo (disco o cinta) hasta que se la "llame". El sistema operativo *colocará* entonces esa subrutina *sobre* la que se ha vuelto redundante (de ahí el nombre *overlay*: sobreponer). Como se puede imaginar, este método para ampliar la memoria disponible es verdaderamente muy útil, pero implica que a menudo uno debe esperar mientras la información se transfiere desde el medio de almacenamiento de apoyo a la memoria principal.

Los paquetes de hojas electrónicas (por lo general se los puede reconocer por su nombre, ya que la mayoría de las veces terminan con la partícula *calc*) existen para una gran variedad de ordenadores personales y de gestión empresarial. El más popular en Gran Bretaña, en función de su volumen de ventas, es el Visicalc, escrito originalmente para el Apple II y que salió a la venta a mediados de 1979. El mercado de software para microordenadores reacciona con mucha rapidez ante el éxito obtenido por uno de sus productos, y este caso no fue una excepción. Antes de que uno pronunciara las palabras "hoja electrónica", ya se hallaban a la venta, a lo largo de todo el país, programas para cualquier clase de máquina, apropiados o no.

Para que una hoja electrónica sea realmente útil ha de satisfacer dos requisitos: dimensiones (no necesariamente las que se ven en la pantalla, porque, como veremos más adelante, lo que se contempla es sólo una "ventana" de la totalidad) y una buena gama de órdenes para control y formato.

Esto significa que existen severas limitaciones en cuanto a las clases de máquinas que tienen posibilidades de ejecutar un programa de hoja electrónica aprovechándolo al máximo. Por regla general, las exigencias mínimas son 32 Kbytes de RAM y una pantalla de 80 caracteres en el caso de una aplicación de gestión, si bien una pantalla de 40 caracteres probablemente será suficiente si se trata de fines domésticos.

Los usuarios de ordenadores personales comprobarán que muchos de los paquetes disponibles para máquinas basadas en cassette, como el Spectrum y el Vic-20, si bien por su propia naturaleza son de dimensiones y potencial limitados, también resultarán muy útiles.

Dado que las hojas electrónicas poseen la capacidad de responder a preguntas del tipo "¿qué ocurriría si...?", es obvio que se pueden emplear para elaborar modelos simples informatizados. Allí dábamos un ejemplo del trabajo del analista de sistemas; si se diera el caso de que usted utilizara un paquete de hoja electrónica, comprendería enseguida la evidente necesidad de efectuar una planificación similar e igualmente cuidadosa. Hemos comentado que las bases de datos consisten en una cantidad de información no clasificada que se ordena de acuerdo a los requerimientos del usuario cuando se recupera la información. Existen procesadores de textos (otro software de gran demanda) que le permiten al usuario desplazar palabras o bloques enteros de texto en el momento de ejecución. Pero las hojas electrónicas son un poco diferentes, en el sentido de que en realidad exigen por parte del usuario un proceso de planificación.

La línea inferior
El cursor es el bloque rectangular que normalmente ocupa el campo N2. Si se digita algo, lo digitado aparecerá en el campo donde estaba situado el cursor. El contenido completo de ese campo también se visualizará en la línea de órdenes de la hoja electrónica, que en este caso está en la parte inferior de la página

FORMULA: SUM (A2:M2)

Por ejemplo, si está utilizando una hoja electrónica para analizar sus gastos, quizá le interese agrupar todos los gastos relacionados con la vivienda (pagos de alquiler o de hipoteca, impuestos, seguro, etc.) y luego emplear el resultado de ese cálculo en una tabla mayor dentro de la misma hoja. Habrá de ser muy cuidadoso al sumar todos los gastos relacionados con la casa antes de trasladar ninguna cifra a la tabla mayor.

Cada dirección individual de la hoja electrónica, denominada celda, se direcciona y localiza en función de sus coordenadas X e Y. Horizontalmente se utilizan las letras A - Z, AA - AZ, e incluso BA - BM, para cubrir la anchura total posible de la hoja, que en las versiones más corrientes es de 65 celdas. Verticalmente se pueden emplear los números del 0 al 255. Cada celda puede contener una etiqueta (por ejemplo "ventas" o "ganancia"), un valor al que se le da entrada como resultado de un cálculo, o bien derivado del mismo (como 1 000), o la fórmula para efectuar ese cálculo, como B4 + B6 * B5. Las fórmulas, debido a que suelen exceder las dimensiones de visualización del recuadro, normalmente se visualizan en una línea separada en la parte superior de la pantalla. Al comenzar una hoja electrónica nueva las dimensiones de la celda estarán preestablecidas, quizás en ocho o nueve dígitos o caracteres. Esto se conoce como dimensiones por omisión. Por lo general, es posible acortar o alargar las celdas para adaptarlas al tipo de cálculo que se esté realizando. Algunos paquetes permiten que la columna situada más hacia la izquierda (normalmente sus títulos o descripciones) sea más ancha que las demás. Y no es necesario decidir de inmediato el tamaño que se desea darles a las celdas. La mayoría de las versiones son susceptibles de ampliar o reducir, incluso después de haber colocado información en ellas. En caso de que se redujeran las dimensiones por debajo de la longitud del contenido, la parte no visualizada no se borra sino que solamente queda oculta a la vista.

El último de los componentes principales de la hoja es la línea de orden, que aparece en la parte superior o inferior de la pantalla en respuesta a la tecla de "orden": /, por ejemplo. Estas órdenes se emplean para dar su formato y manipular el trazado de la hoja en sí misma, no de la información, si bien puede llegar a incidir en la forma en que se presenta ésta. La mayor parte de las versiones más populares de programas para hojas electrónicas permiten efectuar una gran variedad de operaciones con la base de datos. Se puede, por ejemplo, borrar, trasladar o copiar líneas o columnas enteras, y además dividir la ventana para que se visualicen juntas partes de la hoja que normalmente están muy alejadas entre sí como para aparecer juntas en la ventana; después es posible también hacer mover en espiral (mover a través) estas ventanas por separado.

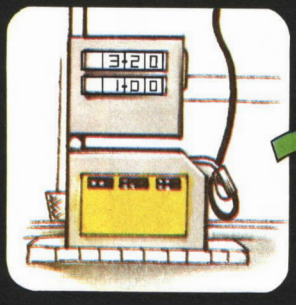
El movimiento entre las celdas se suele realizar mediante la tecla de control del cursor; no obstante, otra tecla de orden permite saltar hasta otra celda determinada. Cargar y guardar, borrar y proteger, todo ello se efectúa empleando las teclas de órdenes, y en este punto vale la pena volver a recalcar la importancia de guardar regularmente el trabajo que uno haya realizado. Componer una hoja electrónica es una tarea que toma mucho tiempo, por lo regular mucho más que el que se necesita para dar entrada a los datos. Por regla general, *siempre* se ha de guardar una hoja antes de empezar a dar entrada a los datos. Luego, en el caso de que se incurriera en un error garrafal, el haberla conservado le evitará una pérdida irreparable.

Los resultados se transmiten a la impresora empleando una tecla de orden, pero el usuario ha de asegurarse de definir la parte de la hoja que desea imprimir mediante la utilización de los parámetros. Así como la pantalla es una ventana, o un fragmento de la totalidad de la hoja, la misma función desempeñará la hoja impresa a través de una impresora de salida. Si desea imprimir una hoja más ancha que su impresora, lo aconsejable es hacer dos impresiones y luego unir las dos páginas.

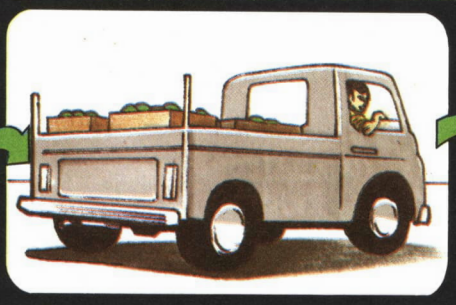
Como ya hemos apuntado, el empleo de ventanas permite que se visualicen simultáneamente dos partes diferentes de la hoja. También se pueden imprimir las hojas con ventana dividida. Esto es especialmente útil al dar entrada a la información, puesto que uno puede hacer una referencia a una entrada anterior. La mayoría de los paquetes le permiten al usuario "retener" la línea superior o la primera línea, o ambas a la vez, que él crea por algún motivo, tomando en cuenta que por lo general éstas contienen los títulos o etiquetas.

Hasta este momento hemos venido considerando la hoja como una tabla a la cual uno sólo puede acceder en serie (un ítem después del otro, a lo largo de una línea o de una columna); pero si uno no desea seguir el trazado normal, sumando líneas o columnas, no existe ningún impedimento para trazar la hoja de cualquier otra forma si procediendo de esa manera se facilita la

¿Qué ocurriría si...?
Si se modifica el contenido de cualquier campo, la hoja electrónica volverá a calcular automáticamente todos los campos que dependan de esa cifra. La velocidad y sencillez de este proceso estimula al usuario a verificar la validez de sus proyectos, para comprobar lo que sucedería con los beneficios globales si variaran determinadas condiciones. Tomemos el ejemplo de un vendedor de frutas...



Si el precio de la gasolina aumentara en un X %...



Luego los costes mensuales de transporte se incrementarían en un Y %...

Esto, a su vez, haría subir el precio al por mayor de las manzanas...

Y ello incidiría directamente en el consumidor a través del aumento de los precios...

Kevin Jones

Mark Watkinson

fluidez de pensamiento del usuario. No obstante, conviene recordar que este tipo de sofisticación dentro de la base de datos requerirá un análisis de los sistemas mucho más exhaustivo que el que normalmente se lleva a cabo.

Las hojas electrónicas para gestión empresarial, como Visicalc, Supercalc y Masterplan, le ofrecen al usuario la posibilidad de pasar información desde la hoja electrónica a paquetes de gestión para bases de datos o tratamiento de textos, y existen muchos programas complementarios para que las salidas se realicen en una variedad de formas gráficas: circulares, por ejemplo, o de barras.

Ya hemos mencionado una de las posibles aplicaciones de las hojas electrónicas para el usuario de ordenadores personales: el análisis de los gastos y el presupuesto del hogar. Otra aplicación muy adecuada para el hogar sería en la instalación de la calefacción central, para la cual se han de considerar gran cantidad de variables: el tipo de combustible que se empleará, la cantidad y la clase de radiadores, la salida de la caldera, etc. De hecho, utilizar una hoja electrónica es de gran ayuda para cualquier proceso que implique tomar una serie de decisiones, en especial porque el usuario está casi obligado a contemplar todas y cada una de las posibilidades.

Quizá la característica más notable de una hoja electrónica para gestión empresarial ejecutada en una máquina adecuada, sea su asombrosa velocidad de funcionamiento. Ésta es una función de la programación en código de lenguaje máquina y no es nada sorprendente que la velocidad de un paquete escrito en BASIC para un pequeño ordenador personal pueda ser, en comparación, motivo de alarma.

Tal vez resulte interesante considerar algunos de los problemas con los que uno se podría enfrentar si intentara escribir un programa de este tipo en BASIC, aunque sólo fuera como un medio para llegar a comprender la complejidad que entrañaría dicha tarea.

Para empezar, cada celda se habría de definir de tres maneras. La celda debería ser capaz de retener datos de "variables alfanuméricas", como "enero" o "tarifas"; debería ser capaz de retener datos numéricos para su utilización en operaciones aritméticas, como por ejemplo, el total de las cuotas pagadas en enero; y también podría ser necesario que retuviera una fórmula que, en esencia, fuera una línea del código de programación, como "cuotas anuales/12" para obtener la cuota mensual. Luego las dimensiones de cada celda deberían ser susceptibles de ampliarse y reducirse, pero sin perder su parte menos significativa, para lo cual todas ellas deberían duplicarse: una aparecería en la pantalla y la otra, que siempre retendría toda la información, quedaría oculta dentro del programa.

Como se puede apreciar, sólo la manipulación de la información es en sí misma una tarea complicada; y recuerde que los paquetes más sofisticados ¡pueden constar de hasta 16 000 celdas individuales! Las técnicas que se utilizan para escribir esta clase de programas se parecen mucho a las que se emplean para escribir intérpretes para lenguajes como el BASIC o el FORTH, y también se hace uso de técnicas similares para el software de gestión por base de datos.

Todo esto viene al caso para explicar el porqué del elevado costo de estos programas empresariales escritos con una finalidad concreta. Un paquete como el Visicalc o el Supercalc puede ser muy caro, pero a la hora de contemplar la posibilidad de comprarlo, quizá el punto a considerar sea el ahorro que supone su utilización. La relativamente sencilla aplicación que realizamos anteriormente, la del director que recopilaba presupuestos, por ejemplo, puede representar un ahorro de quizá entre un 15 y un 20 % en un solo año. El costo del software constituye una parte insignificante en relación al monto total de inversión sin contar el ahorro de tiempo y energía que trae aparejado su aplicación. Por cierto, el paquete Visicalc probablemente haya sido el mejor amigo de los agentes de ventas del Apple.

Ventanas al mundo
Así como podemos desplazar el cursor por la pantalla hacia la derecha o hacia la izquierda, hacia arriba o hacia abajo, mediante el control del teclado, en una hoja electrónica podemos mover la pantalla a través de la hoja. Esta sofisticación permite visualizar una superficie mucho mayor que la de la pantalla.
Algunos ordenadores portátiles con pantalla de dimensiones limitadas, como el Epson HX-20 y el Osborne-1, utilizan el mismo sistema para todas sus operaciones

Juegos de aventuras

El ordenador puede ser una fuente de diversión. En un juego de aventuras, uno no es espectador: es el protagonista

Pronuncie la palabra "aventura" y la mayoría de la gente pensará en un libro, una película, un programa de televisión e incluso, quizá, en una experiencia personal. Pero muchas personas pensarán en los ordenadores, porque existe multitud de propietarios de ordenadores para quienes la palabra Aventura (con A mayúscula) se encuentra asociada a una clase de juegos por ordenador muy específica.

Para hacernos una idea de en qué consisten las aventuras por ordenador, comparémoslas con los libros. Cuando se lee una historia de aventuras se disfruta con los peligros, los misterios y los sucesos emocionantes; pero el protagonista es otra persona. En una aventura por ordenador, en cambio, uno no es un espectador, sino que forma parte de la acción. Como principal protagonista de la historia, se encuentra inmerso en la acción y es uno mismo quien está viviendo la experiencia.

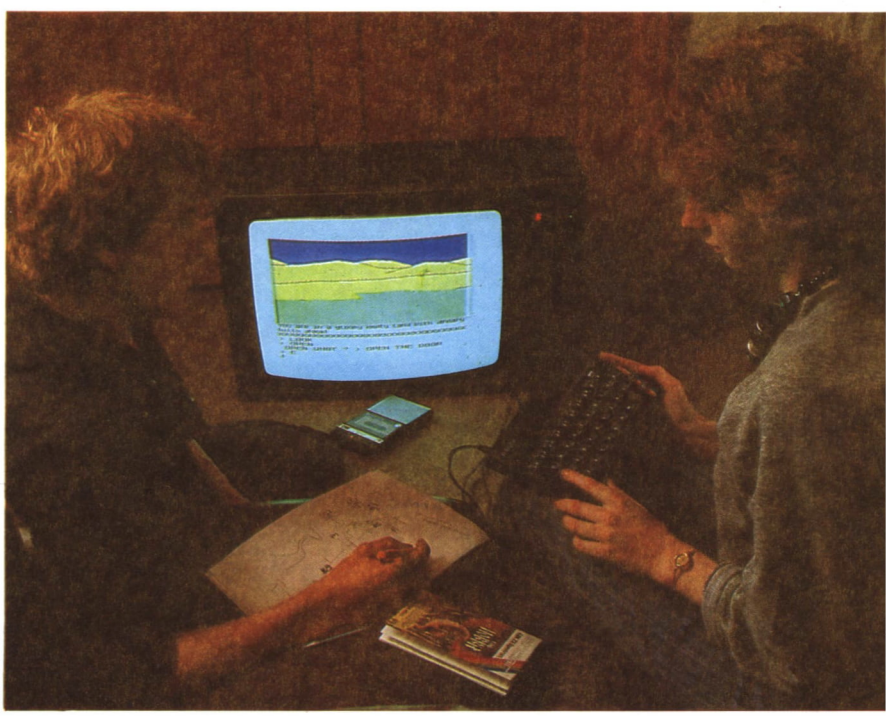
En un libro, el lector no puede modificar el curso de los acontecimientos de la narración. Los sucesos siempre son los mismos y se producen en un orden determinado, y por más que se los vuelva a leer una y otra vez, nada variará en lo más mínimo. En cambio, en una aventura por ordenador sus decisiones, pareceres y acciones determinan la forma en que se desarrolle el argumento. Puede haber cualquier número de variaciones en cuanto al orden de los acontecimientos y muchos finales diferentes, algunos agradables y otros no tanto. En un juego de aventuras lo esencial es que uno es parte activa, aunque rodeado del confort y la seguridad de su propio hogar.

Cada aventura se desarrolla en un ambiente diferente. Éste puede ser un extraño mundo subterráneo, una feria al aire libre, una ciudad fantasma, otro planeta, una tierra mítica; en realidad, cualquier lugar. La acción se puede situar en el pasado, en el presente o en el futuro.

Por lo general la aventura posee una trama coherente que incluye un objetivo final. Por ejemplo, la meta puede ser huir de un planeta desconocido, encontrar y destruir a un genio del mal, rescatar a una princesa, hallar un tesoro o esclarecer un crimen y arrestar al culpable.

Por muy interesantes que le parezcan estos ingredientes, el verdadero placer de jugar a un juego de aventuras reside en solucionar los enigmas. Éstos constituyen una parte esencial de la aventura. Se puede encontrar un enigma en una de cuatro situaciones. La situación más común consiste en tener que resolverlo para poder seguir avanzando; por ejemplo, al hallarse ante el peligro de un puente a punto de derrumbarse. En otro sentido, el enigma suele ser una pista falsa: después de haber atravesado un desfiladero con el fin de abordar a la extraña figura que lo había estado observando desde la otra orilla, usted sólo encuentra ante sí un enorme espejo. Hay enigmas cuya resolución ayuda a completar el juego exitosamente, aunque no sean un factor esencial: por ejemplo, el descubrimiento de un pasaje secreto que conduce hasta una ciudad de enanos viciosos. Puede ser una cuestión de vida o muerte: usted se encuentra solo y abandonado en el saliente de una montaña, y no existe forma alguna de seguir ascendiendo o de retroceder, y no tiene ni comida ni bebida.

Los enigmas se resuelven utilizando el sentido común y no requieren ninguna experiencia ni conocimientos especiales. No obstante, el aventurero ha de estar alerta, ya que las claves para la solución siempre se han de hallar en el texto o bien se deben deducir del mismo. En los juegos de aventuras bien escritos no existen elementos dejados al azar, excepto en dosis muy pequeñas.



Ian Dobbie

Al jugar a una aventura probablemente encontrará objetos, mensajes y personajes que parecerán irrelevantes en el contexto de la historia. No obstante, siempre debe tener presente que en esta clase de juegos prácticamente todo tiene un fin determinado, incluso aunque, en ocasiones, esa finalidad específica sea la de despistarlo. ¿Qué significado puede tener un montón de botellas de ron rotas? ¿Cómo debe uno interpretar una voz profunda que le dice «REPITA»? ¿Qué utilidad práctica puede ofrecer un lodazal maloliente? ¿Por qué ese felpudo estaba clavado al suelo si no había nada debajo de él? Todos estos enigmas están incluidos en varios juegos de aventuras y desempeñan un papel esencial en la trama. Cuando se encuentra un objeto por primera vez, independientemente de lo común o extraño que resulte, es muy improbable que

Un papel preponderante
Los juegos de aventuras, como *Dungeons and dragons* (Calabozos y dragones), existen ya desde hace muchos años; pero en la actualidad el ordenador personal los ha puesto al alcance de la mano de una gama de jugadores potenciales mucho más amplia. Los juegos de aventuras empiezan a competir, en cuanto a popularidad, con los juegos recreativos. Aunque la mayoría de ellos aún son para un solo jugador, pueden ser una absorbente fuente de recreación para todos los miembros de la familia. El jugador asume el papel de protagonista

Lectura de un mapa
En un juego de aventuras, el jugador o el personaje que éste representa se ha de mover en un vasto territorio o ambiente; es muy corriente que exista una red de pasadizos subterráneos. Algunos fabricantes incluyen pistas en los manuales de instrucción para aquellos jugadores que, llegado un momento, se queden absolutamente atascados; pero la única forma de descubrir el trazado consiste en abrirse camino a través de él. Para completar un juego de este tipo se pueden invertir semanas enteras, de manera que es imprescindible llevar un boceto esquemático que le indique en qué lugares ha estado y los obstáculos que ha encontrado a su paso

se tenga de inmediato la certeza de su importancia; sin embargo, es seguro, como dos y dos son cuatro, que el jugador lo necesitará mucho antes de lo que cree.

Muchos juegos de aventuras poseen un pequeño laberinto en el que cada sitio o habitación se describe en idénticos términos. La única manera fiable de trazar un mapa de ruta a través de este tipo de laberintos, consiste en imitar a Hansel y Gretel, es decir, dejar caer pequeños objetos en el sendero que se sigue, con el fin de poder identificar cada cuarto. Esta estratagema se ha hecho tan conocida que algunos programadores han ideado problemas adicionales; por ejemplo, que un ladrón vaya tras el aventurero y, con la mayor discreción, cambie la disposición de los objetos que sirven de orientación.

En algunas aventuras, si bien para completar el juego exitosamente se han de resolver todos los enigmas y superar todos los escollos, el orden en que se resuelvan los misterios y se alcancen los objetivos carece de importancia. Este tipo de juegos se opone a aquellos en los cuales existe un único camino para llegar a un desenlace triunfal.

Una buena aventura puede ocuparle horas, incluso semanas de juego hasta que se descubren todos sus secretos. Entonces, cuando decida suspender el juego, estará capacitado para guardarlo en la fase en que se encuentre, en una cassette o en un disco. Este recurso también resulta útil cuando se ha llegado a una etapa peligrosa de la aventura, porque continuar avanzando impertérrito a la deriva hacia un grupo de orcas, provisto apenas de una lámpara y una botella de agua, es proceder con excesiva temeridad. Un aventurero prudente "congelará" el juego antes de tener que vérselas con las orcas. Luego, si éstas deciden terminar con los días del aventurero, por lo menos existirá la posibilidad de reanudar el juego después e intentar un curso de acción diferente, lo cual resultará mucho menos desalentador que volver a empezar la aventura desde el principio.

Incluso aunque el jugador sea muerto, ello no implica necesariamente un final definitivo. Algunos autores permiten que el difunto resucite, a menudo en medio de una humareda color naranja, a veces perdiendo puntos o algunas de sus posesiones, y siempre apareciendo en un lugar más bien indeseable, como, por ejemplo, el infierno o en medio de la nada.

¿Cómo participar y comunicarse con el programa? Éste puede dirigirse directamente al jugador o bien éste puede ser representado por un "muñeco", personaje al que se controla mediante órdenes. El ordenador actúa al mismo tiempo como intérprete de los deseos del aventurero y como narrador. El jugador da entrada a las órdenes a través del teclado y éstas se visualizan en la pantalla.

Algunas aventuras visualizan en pantalla sólo el texto, otras únicamente gráficos, el resto, una mezcla

David Higham

de ambos. Los efectos sonoros se utilizan por lo general en los juegos de aventuras exclusivamente con gráficos. Las aventuras de texto se pueden comparar con un libro sin ilustraciones, en el cual los lugares, objetos y acontecimientos se describen mediante palabras. En la mayor parte de los juegos que combinan texto y gráficos, éstos sirven de complemento a las descripciones del texto y a menudo son escenas estáticas de objetos y lugares. Estas escenas pueden ser desde unos sencillos trazados lineales hasta imágenes muy detalladas. En las aventuras con gráficos, éstos normalmente se utilizan en mapas estilizados o imágenes del terreno, así como para representar los interiores de las construcciones. Los personajes y los objetos se representan mediante símbolos o figuras en miniatura. En este caso, la acción del jugador por lo general queda limitada a una pequeña cantidad de órdenes, básicamente pulsaciones individuales de teclas para hacer mover y controlar al personaje.

La visualización de texto en una aventura comprende, por lo común, tres elementos: dónde se halla el jugador, qué puede ver y a dónde puede ir. Por ejemplo, el texto en pantalla puede decir: «Usted se encuentra en la espesura de un bosque. Sobre su cabeza, el cielo no es más que una serie de manchas entre el denso follaje. Un sendero muy transitado lleva hacia el este y hacia el oeste. Delante, hacia el norte, se abre un abismo. En el suelo está clavada una espada; enroscada en ella hay una serpiente verde». Al jugador se le describen los alrededores, se le dan algunas de las direcciones que puede tomar y se le indican los objetos que están a la vista.

Las órdenes constan, por lo general, de dos palabras, un verbo seguido de un sustantivo, si bien las aventuras más sofisticadas incluyen oraciones completas. Algunos de los verbos estándar son GET (coger), DROP (soltar), PUSH (empujar), PULL (tirar), THROW (arrojar), LIGHT (encender), KILL (matar), EAT (comer) y DRINK (beber). GO NORTH (ir hacia el norte), por ejemplo, sería la forma normal de especificar un movimiento, si bien la mayoría de los juegos de aventuras permiten abreviar este tipo de órdenes, y así, es muy corriente el empleo de iniciales, como W para GO WEST (ir hacia el oeste).

EXAMINE (examinar) es un verbo esencial; a menudo es el medio para adquirir mayor información. En el ejemplo anterior, EXAMINE SNAKE (examine la serpiente) podría significar: "Se trata de una culebra de hierba", o tal vez: "La serpiente advierte que usted avanza hacia ella y lo ataca". Algunos verbos no requieren un sustantivo.

INVENTORY (inventario) se emplea para decirle lo que está llevando. Algunos objetos se pueden colocar dentro de otros (agua en una botella o un hacha en un saco, por ejemplo), mientras que otros se pueden llevar encima (por ejemplo, un anillo o una capa).

Con frecuencia se emplea SCORE (puntuación) para que el jugador sepa qué progresos ha hecho en alcanzar los objetivos. Digitar HELP (ayuda) puede facilitar una pista para superar una dificultad, pero generalmente le aconseja que siga intentándolo. En ocasiones, en respuesta a una orden determinada, se le puede decir al aventurero: "¡No puede hacer eso... todavía!", lo que implica que una combinación verbo-sustantivo dará un resultado concreto pero no en ese momento y, quizá, no en ese lugar. Una parte del desafío consiste en averiguar los verbos y los sustantivos que son relevantes para la aventura. Las palabras y las combinaciones que ésta no reconoce suelen dar como respuesta algo como "No le comprendo".

Ian McKinnell

En busca del tesoro

El Hobbit es un juego de aventuras con gráficos para el Sinclair Spectrum y toma su nombre de la novela de J. R. R. Tolkien, un ejemplar de la cual se proporciona con la cassette. Usted asume el papel de Bilbo, viajando a través de Middle Earth y encontrándose con muchos de los personajes y situaciones del libro, en busca del dragón y su tesoro

La mayoría de los editores de aventuras facilitan hojas de pistas para quienes se quedan trabados, ofreciendo sutiles claves para seguir adelante.

Algunos juegos de aventuras son demasiado largos para la memoria del ordenador. Para superar este inconveniente se proporcionan en disco; el programa principal se carga en la memoria al principio y los textos y gráficos se toman del disco o se devuelven a él a medida que se los va necesitando. Pero gracias a las inteligentes técnicas para comprimir grandes cantidades de texto en una memoria restringida, ahora un juego de aventuras largo se puede almacenar en la memoria de un ordenador, lo que significa que estas aventuras se pueden suministrar en cinta de cassette.

Existen aventuras para virtualmente toda clase de micros. Cualquiera le proporcionará una grata experiencia y puede ser el comienzo de un hobby.

¡Felices aventuras!

Ian Dobbie

Resolviendo un caso
Deadline (Plazo fatal) es una variante temática de las aventuras. Usted encarna el papel de un detective que debe esclarecer un asesinato. Su archivo contiene, por ejemplo, los resultados de la autopsia, algunas píldoras que se hallaron junto al cuerpo y otras notas relativas al caso. El juego no se desarrolla en tiempo real (dos semanas se considera un muy buen tiempo), pero cada acción que usted empreda, como ir de un cuarto a otro o interrogar a un sospechoso, representa unos minutos vitales que se descuentan del tiempo límite de 12 horas que se le ha dado para resolver el caso

Números al azar

Siguiendo con nuestro estudio de las funciones del BASIC llegamos a la función RND, que produce números al azar (o casi al azar) para su utilización en juegos o en programas estadísticos

Ahora que ya hemos visto cómo trabajan varias de las funciones de BASIC, examinaremos una de las que más se utilizan: la función RND. RND se usa para generar números al azar. También se emplea en los juegos cada vez que existe un elemento de casualidad.

Lamentablemente, RND es una de las palabras más "variables" del BASIC. La descripción que nosotros hagamos de ella puede diferir respecto a la función que tenga destinado realizar en algunos micros personales. Aclaremos, entonces, las diferencias existentes entre el BASIC utilizado en nuestro curso de programación BASIC y otros BASIC.

La mayoría de nuestros programas se basan en el BASIC Microsoft (o MBASIC). Microsoft es una empresa norteamericana y su BASIC fue uno de los primeros que estuvieron disponibles a nivel masivo. El BASIC es un lenguaje que no está estandarizado oficialmente, pero el de Microsoft se aproxima, dentro de lo posible, a lo que sería una versión estandarizada. Existen muchas versiones elaboradas a partir de la de Microsoft, y esta empresa tiene encomendado producir otras para varios ordenadores populares.

La diferencia fundamental entre el MBASIC y la mayoría de las versiones más recientes es que ahora los ordenadores personales son capaces de realizar excelentes gráficos, facultad que no poseían cuando se desarrolló el MBASIC. Otras versiones de BASIC incluyen generalmente un número de órdenes y sentencias para gráficos. Para obtener el máximo rendimiento de las prestaciones del ordenador, es necesario aprovechar en profundidad su capacidad para realizar gráficos, y ello requiere un exhaustivo estudio del manual por parte del usuario.

De los diversos BASIC que proporcionan los ordenadores personales más populares, probablemente el BASIC del Sinclair (utilizado en el ZX81 y en el Spectrum) y el BASIC del BBC sean los que más difieren del MBASIC. La versión del Texas Instruments (empleada en el TI99/4A) también incorpora una cantidad de diferencias significativas. En la medida de lo posible, en los recuadros de "Complementos al BASIC" le indicamos cómo debe modificar nuestros programas, y se

debe remitir a estos recuadros en caso de que encuentre alguna dificultad al ejecutar los programas.

Como hemos mencionado antes, la función RND varía de una versión a otra. Compruebe en su manual de BASIC qué cometido cumple en su versión. Nosotros vamos a ilustrar su utilización a partir de un sencillo juego de dados. Al igual que en programas anteriores, hemos realizado la mayor parte del trabajo en forma de subrutinas. Esta técnica ofrece la ventaja de hacer que los programas resulten más fáciles de leer, de escribir y de depurar.

El programa principal comienza con la sentencia RANDOMIZE en la línea 20. La mayoría de las versiones de BASIC, pero no todas, necesitan de esta sentencia para "redistribuir" la función RND. En realidad es bastante difícil lograr que los ordenadores produzcan números auténticamente al azar. Sin esta operación de redistribución, se produciría la misma secuencia de números supuestamente al azar cada vez que se aplicara RND. La línea 50 remite luego a una subrutina que utiliza RND para asignarle a la variable D un número al azar. La forma en que la hemos empleado es:

```
320 LET D = INT (10 * RND)
```

Ésta es la línea que, con toda probabilidad, el usuario habrá de modificar cuando dé entrada al programa. En "Complementos al BASIC" se proporcionan detalles de cómo funcionan las distintas versiones de RND, de manera que veamos qué es lo que está sucediendo en este BASIC Microsoft. La función RND usa una expresión (entre paréntesis, como es normal en estas funciones) como una opción para alterar ligeramente la secuencia de números producida. Sin ninguna expresión (por ejemplo, LET A = RND), el valor de A será un número entre 0 y 1. Nosotros no deseamos un número inferior a 1, por lo cual multiplicamos el número por 10. Esto se puede hacer de la siguiente manera: LET A = 10 * RND. Si, en aras de la demostración, RND hubiera devuelto el valor 0,125455, ahora el valor de A sería 1,25455.

Para eliminar la fracción del número y conservar sólo la parte entera, utilizamos la función INT (entero) de esta manera: LET A = INT (10 * RND). Algunas versiones de BASIC permiten especificar el límite máximo de los números generados al azar en la expresión encerrada entre paréntesis después de RND. Por ejemplo, el BASIC del Dragon imprimirá un número entero entre 1 y 6 en respuesta a PRINT RND(6).

Dado que con nuestro BASIC Microsoft no podemos hacer esto, comprobamos si los números devueltos son mayores que 6 o menores que 1, porque esos números no sirven para un juego de dados. Esto se realiza en las líneas 330 y 340:

```
330 IF D > 6 THEN GOTO 320
340 IF D < 1 THEN GOTO 320
```

Si D quedara fuera de los límites entre 1 y 6, GOTO hará que el programa salte hacia atrás y lo intente otra vez.

Habiendo escogido para D un valor al azar entre 1 y 6, la subrutina de tirar los dados vuelve (RETURN) al programa principal. Éste imprime el mensaje SU NUMERO ES UN, seguido de la imagen de un dado. Observe cómo se selecciona la imagen apropiada de un dado. Esto se efectúa en la subrutina de SELECCION. Por ejemplo, si el dado (y por tanto D) es un 1, la línea 410 llama a la subrutina que comienza en la línea 530, de manera que:

```
410 IF D = 1 THEN GOSUB 530
```

Esta subrutina no es más que una serie de sentencias PRINT diseñada para producir gráficos muy rudimentarios en la pantalla. Bien puede ser que el BASIC del ordenador del usuario posea gráficos de pantalla mucho mejores y, si éste es el caso, lo mejor sería reemplazar nuestras subrutinas por las sentencias para gráficos apropiadas.

Una vez que el programa ha elegido un dado al azar para el usuario, repetirá luego el proceso para escoger y visualizar un dado para el ordenador. La parte del programa que decide quién ha ganado se ha incorporado al programa principal; también se la podría haber escrito en forma de subrutina, pero no habría valido la pena, puesto que sólo tiene cuatro líneas. La línea 200 compara M (mis dados) con C (los del ordenador) para ver si son iguales. Si lo son, a la variable alfanumérica S$ se le asignan las palabras ES UN EMPATE. La línea 210 compara para comprobar si M es mayor que C. Si lo es, le asigna a S$ las palabras HA GANADO USTED. La línea 220 comprueba si M es menor que C. En ese caso, le asigna a S$ las palabras HA GANADO EL ORDENADOR. La línea 240 simplemente imprime el resultado y el juego acaba. Aunque este programa es bastante largo, en esencia es muy sencillo. Utiliza solamente una función, RND, no tiene bucles, ni variables subíndice, ni ninguna otra complicación excepto algunas sentencias IF...THEN.

Dado que la función RND es tan variable y que algunas versiones de BASIC (la Microsoft, por ejemplo) requieren la sentencia RANDOMIZE para generar una nueva secuencia de números al azar, ¿existe algún procedimiento para que podamos generar números auténticamente al azar (es decir, impredecibles) sin usar estas funciones? Sí, existen varias técnicas.

Una de las funciones que hasta ahora no habíamos examinado es la función INKEY$ (que corresponde a *inkey-string*). Cada vez que se encuentra con la palabra INKEY$, el programa inspecciona el teclado para ver si se ha pulsado alguna tecla. El programa no espera a que se dé entrada a un carácter, como lo hace cuando se emplea la orden INPUT. Por tanto la orden INKEY$ normalmente se utiliza dentro de un bucle. El programa entonces explora continuamente el teclado, esperando alguna entrada. Por lo general en el bucle se incluye alguna comparación para que éste termine cuando se dé entrada a un carácter apropiado. Esto permite escribir un programa que forme un bucle para contar que concluirá cuando se haya digitado un carácter específico. ¿Qué sucedería si empleáramos este programa?

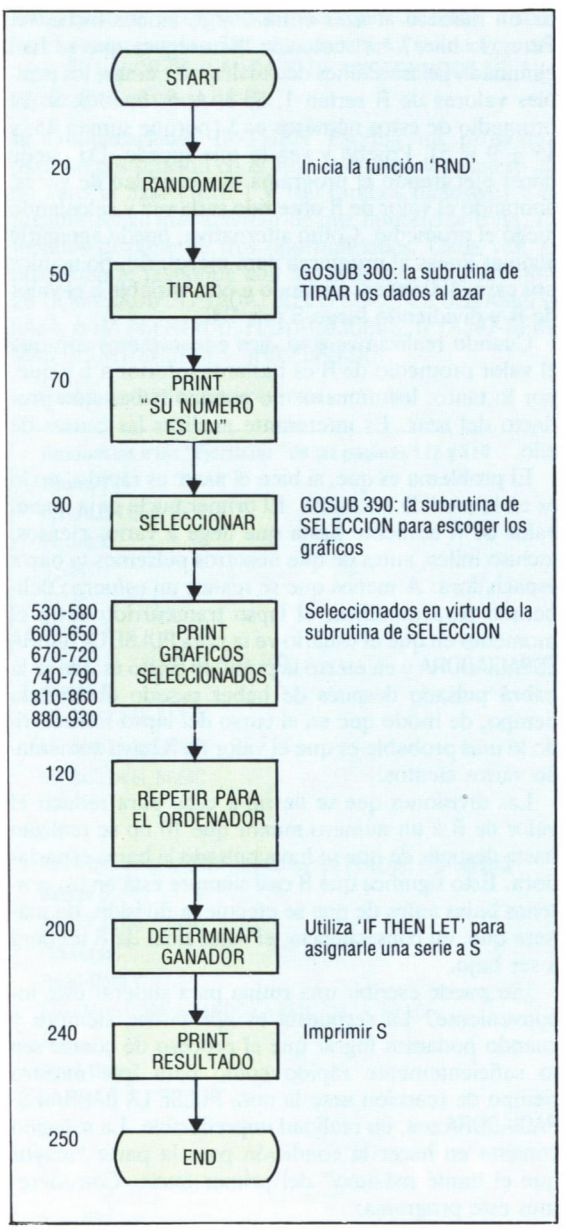
```
10 PRINT "PULSE LA BARRA ESPACIADORA"
20 FOR X = 0 TO 1
30 LET R = R + 1
40 LET A$ = INKEY$
50 IF A$ = " " THEN GOTO 80
60 LET X = 0
70 NEXT X
80 FOR Q = 0 TO 1
90 IF R < 10 THEN GOTO 130
100 LET Q = 0
110 LET R = R/10
120 NEXT Q
130 PRINT INT (R)
140 END
```

¿Será R un número al azar? Debería serlo, de modo que procedamos a analizar el programa y comprobemos por qué.

La línea 10 imprime la nota PULSE LA BARRA ESPACIADORA. Antes de que podamos responder a esta nota, el programa ha dado entrada el bucle FOR X = 0 TO 1 en la línea 20. Puede que 0 y 1 parezcan límites extraños para el bucle, pero enseguida veremos cómo se utiliza esta estructura. La línea 30 le asigna a la variable R el valor 1 la primera vez que se efectúa el bucle. La línea 40 le asigna a la variable alfanumérica A$ el carácter digitado en el teclado, cualquiera que haya sido. Esto se hace empleando la función INKEY$. Si el usuario pulsara la letra R, a A$ se le asignaría R. La línea 50 compara A$ para ver si es un espacio (que en BASIC se representa como un espacio encerrado entre comillas dobles: " "). Si A$ es un espacio, el programa se bifurca valiéndose de la sentencia GOTO; pero si A$ no es un espacio, el programa continúa por la siguiente línea.

Se trata de la línea 60, que dice LET X = 0. Ahora X es el índice del bucle. La sentencia NEXT X de la línea 70 hace que el programa retorne al comienzo del bucle, en la línea 20. Dado que X se ha reestablecido en 0, el bucle la repite. De esta manera, el bucle FOR X = 0 TO 1 se repetirá indefinidamente, en la medida en que fracase la comparación IF A$ = " ".

Si en algún momento se pulsa la barra espaciadora,



Flujo del programa
El diagrama de flujo muestra de forma simplificada las acciones principales que realiza el programa. A la izquierda se indican los números de las líneas correspondientes y, a la derecha, se dan unas breves notas explicativas. No se trata de un diagrama de flujo completo, ya que no incluye muchas de las "decisiones" y bifurcaciones del programa

a A$ se le asignará el carácter que represente un espacio, y el programa se bifurcará hasta la línea 80 y el bucle no se repetirá más.

Pero ¿qué sucede mientras el bucle se repite? A cada repetición del bucle, la línea 30 aumenta el valor de R. La primera vez que se realiza, R se establece en 1, la segunda vez se establece en 1 + 1 y así sucesivamente. Cuando el bucle se interrumpe a raíz de la comparación de A$, podemos leer R para ver hasta dónde hemos contado.

Sin embargo, los ordenadores funcionan a gran velocidad, de modo que, para cuando pulsemos la barra espaciadora, R podría haber llegado a varios centenares. ¿Qué haremos si deseamos que los valores de R se hallen sólo entre 1 y 10? La línea 80 introduce otro bucle que nos permite comparar R y dividirla por 10 en el caso de que sea mayor que 10. En la medida en que R sea mayor que 10, fallará la comparación de la línea 90, el valor de Q se reestablecerá en 0 y se repetirá el bucle. La línea 110 divide por 10 el valor de R, pero el resultado no se imprimirá en tanto no se dé la condición de que el valor de R se haya reducido a un número menor que 10. La línea 30 asegura que el valor de R nunca podrá ser 0.

En teoría, entonces, este programa debería producir un número al azar entre 1 y 9, ambos inclusive. Pero ¿lo hace? La sentencia INT asegura que se han eliminado las fracciones decimales; por tanto, los posibles valores de R serían 1, 2, 3, 4, 5, 6, 7, 8, 9. El promedio de estos números es 5 (porque suman 45, y 45 ÷ 9 = 5). Pruebe y vea lo que sucede. Lo puede hacer ejecutando el programa una cantidad de veces, anotando el valor de R obtenido cada vez y calculando luego el promedio. Como alternativa, puede agregarle algunas líneas al programa para ejecutarlo, pongamos por caso, 100 veces, sumando a otra variable S el valor de R y dividiendo luego S por 100.

Cuando realizamos esto, nos encontramos con que el valor promedio de R es bastante inferior a 5 y que, por lo tanto, los números no pueden haber sido producto del azar. Es interesante analizar las causas de ello.

El problema es que, si bien el BASIC es rápido, no lo es en la medida necesaria. El primer bucle deja que el valor de R aumente hasta que llega a varios cientos, incluso miles, antes de que nosotros pulsemos la barra espaciadora. A menos que se realice un esfuerzo deliberado para modificar el lapso transcurrido entre el momento en que el usuario ve la nota PULSE LA BARRA ESPACIADORA y en efecto la pulsa, lo cierto es que se la habrá pulsado después de haber pasado demasiado tiempo, de modo que en el curso del lapso transcurrido lo más probable es que el valor de R haya aumentado varios cientos.

Las divisiones que se llevan a cabo para reducir el valor de R a un número menor que 10 no se realizan hasta después de que se haya pulsado la barra espaciadora. Esto significa que R casi siempre está en las centenas bajas antes de que se efectúe la división, de manera que, en consecuencia, el valor final de R tenderá a ser bajo.

¿Se puede escribir una rutina para superar este inconveniente? La respuesta es afirmativa, siempre y cuando podamos lograr que el proceso de contar sea lo suficientemente rápido como para que nuestro tiempo de reacción ante la nota PULSE LA BARRA ESPACIADORA sea, en realidad impredecible. La solución consiste en hacer la condición para la parte "mayor que el límite máximo" del primer bucle. Consideremos este programa:

```
10 REM JUEGO DE DADOS -- PROGRAMA PRINCIPAL
20 RANDOMIZE
30 REM SU TURNO
40 REM GOSUB RUTINA DE'TIRAR'
50 GOSUB 300
60 LET M = D
70 PRINT "SU NUMERO ES UN"
80 REM GOSUB RUTINA DE'SELECCION'
90 GOSUB 390
100 PRINT
110 REM TURNO DEL ORDENADOR
120 REM GOSUB RUTINA DE'TIRAR'
130 GOSUB 300
140 LET C = D
150 PRINT "EL NUMERO DEL ORDENADOR ES UN"
160 REM GOSUB RUTINA DE'SELECCION'
170 GOSUB 390
180 PRINT
190 REM  ¿QUIEN GANO?
200 IF M = C THEN LET S$ = "EMPATE"
210 IF M > C THEN LET S$ = "GANO USTED"
220 IF M < C THEN LET S$ = "GANO EL ORDENADOR"
230 REM PRINT RESULTADO
240 PRINT S$
250 END
260 REM
270 REM
280 REM
290 REM
300 REM SUBRUTINA DE TIRAR LOS DADOS AL AZAR
310 REM
320 LET D = INT(10* RND)
330 IF D > 6 THEN GOTO 320
340 IF D < 1 THEN GOTO 320
350 RETURN
360 REM
370 REM
380 REM
390 REM SUBRUTINA DE SELECCION
400 REM
410 IF D = 1 THEN GOSUB 530
420 IF D = 2 THEN GOSUB 600
430 IF D = 3 THEN GOSUB 670
440 IF D = 4 THEN GOSUB 740
450 IF D = 5 THEN GOSUB 810
460 IF D = 6 THEN GOSUB 880
470 RETURN
480 REM
490 REM
500 REM
510 SUBRUTINAS DE 'GRAFICOS'
520 REM
530 PRINT "          "
540 PRINT "  ┌─────┐ "
550 PRINT "  │     │ "
560 PRINT "  │  *  │ "
570 PRINT "  │     │ "
580 PRINT "  └─────┘ "
590 RETURN
600 PRINT "          "
610 PRINT "  ┌─────┐ "
620 PRINT "  │    *│ "
630 PRINT "  │     │ "
640 PRINT "  │*    │ "
650 PRINT "  └─────┘ "
660 RETURN
670 PRINT "          "
680 PRINT "  ┌─────┐ "
690 PRINT "  │    *│ "
700 PRINT "  │  *  │ "
710 PRINT "  │*    │ "
720 PRINT "  └─────┘ "
730 RETURN
740 PRINT "          "
750 PRINT "  ┌─────┐ "
760 PRINT "  │*   *│ "
770 PRINT "  │     │ "
780 PRINT "  │*   *│ "
790 PRINT "  └─────┘ "
800 RETURN
810 PRINT "          "
820 PRINT "  ┌─────┐ "
830 PRINT "  │*   *│ "
840 PRINT "  │  *  │ "
850 PRINT "  │*   *│ "
860 PRINT "  └─────┘ "
870 RETURN
880 PRINT "          "
890 PRINT "  ┌─────┐ "
900 PRINT "  │*   *│ "
910 PRINT "  │*   *│ "
920 PRINT "  │*   *│ "
930 PRINT "  └─────┘ "
940 RETURN
```

```
10 PRINT "PULSE LA BARRA ESPACIADORA"
20 FOR X = 0 TO 1
30 LET R = R + 1
40 IF R > 9 THEN LET R = 1
50 IF INKEY$ = "  " THEN GOTO 80
60 LET X = 0
70 NEXT X
80 PRINT R
90 END
```

En este programa, R nunca puede ser menor que 1 ni mayor que 9. Para cuando se pulse la barra espaciadora (y lo reconozca la función INKEY$ de la línea 50), R tendrá un valor cualquiera comprendido entre 1 y 9, ambos inclusive.

Este programa fue comprobado mil veces y se halló para R un valor promedio de 5,014. Puesto que el promedio perfecto sería 5 y que el margen de error es sólo un 0,28 % superior, esto sugeriría que de hecho el programa genera un número al azar que se aproxima bastante al promedio teórico. La cuestión estriba, por supuesto, en que incluso cuando un programa parece razonable sobre el papel, luego, en la práctica, puede tener algunos fallos imprevistos. Este motivo justifica el que siempre valga la pena realizar una comprobación a conciencia.

Algunos lectores habrán observado que estos programas para números al azar se pueden acortar mediante la utilización de varias sentencias GOTO en lugar del bucle FOR...NEXT. En próximos capítulos de nuestro curso de programación BASIC explicaremos las razones por las cuales hemos preferido evitar las sentencias GOTO.

Complementos al BASIC

RANDOMIZE

En el micro BBC y el Oric-1, elimine la línea 20 y sustituya la 320 por:

```
320 LET D = INT (10 * RND(1))
```

RND

En el Dragon-32, suprima la línea 20 y sustituya la 320 por:

```
320 LET D = RND(6)
```

y elimine las líneas 330 y 340.

En el Lynx, sustituya la línea 20 por:

```
20 RANDOM
```

En el Vic-20 y el Commodore 64, reemplace la línea 20 por:

```
20 LET X = RND(-TI)
```

y sustituya la línea 320 por:

```
320 LET D = INT(10 * RND(1))
```

En el Spectrum, la palabra RANDOMIZE aparece abreviada en el teclado (RAND), pero en la pantalla se leerá RANDOMIZE

INKEY$

En el Oric-1 y en el Lynx, reemplace INKEY$ por KEY$.
En el Vic-20 y el Commodore 64, sustituya la línea 40 por:

```
40 GET A$
```

Luego reemplace la línea 50 por:

```
50 GET A$: IF A$ = "  " THEN GOTO 80
```

En el micro BBC, sustituya INKEY$ por INKEY$(10). El número entre paréntesis indica el tiempo, expresado en centésimas de segundo, durante el cual el sistema aguardará a que se pulse una tecla; así que para obtener una respuesta rápida deberá usar un número bajo, y viceversa

Ejercicios

■ **Función RND** Modifique el último programa del texto para que dé un número al azar entre 1 y 6 (ambos inclusive).

■ **Bucle y promedio** Agregue algunas líneas al último programa del texto con la finalidad de hacer que se repita cien veces y a continuación dé el promedio de los cien resultados.

■ **Sustitución por una subrutina** Sustituya las líneas 50 y 130 del programa principal (es decir, la subrutina de tirar los dados al azar) por un GOSUB que invoque a su "generador de números al azar" existente en el primer ejercicio.

■ **INKEY$** Utilizando la función INKEY$, ¿cómo escribiría usted un programa que leyera cualquier tecla digitada en el teclado e imprimiera: LA TECLA QUE USTED HA PULSADO ES: * (* representa la tecla pulsada por usted).

■ **Bucle de tiempos** Escriba un bucle de tiempos (un bucle "para contar") y utilice la función INKEY$ para hallar cuál es el valor más alto que alcanza una variable al cabo de 10 segundos (necesitará un reloj). Escriba el programa de modo que la última salida impresa diga: EL VALOR DE R AL CABO DE 10 SEGUNDOS ES: * (* representa el valor de R).

■ **Comparaciones IF-THEN** Escriba un programa para juegos sencillo en el cual el ordenador genere un número al azar entre 1 y 100 (ambos inclusive) y el jugador tenga que adivinar de qué número se trata. El jugador dispone de tres oportunidades. Cada vez, el programa le responde con los mensajes SU ELECCION ES DEMASIADO ELEVADA, SU ELECCION ES DEMASIADO BAJA, o HA ACERTADO, FELICITACIONES, o YA NO TIENE MAS OPORTUNIDADES. ¡HA PERDIDO!

Las respuestas, en el próximo capítulo.

Respuestas a los "Ejercicios" de las páginas 118 y 119

Bucles 1

EL VALOR DE A ES 450

Bucles 2

START
STOP

Bucles 3

AHORA EL VALOR DE A ES 160

Bucles 4

ME GUSTA EL BASIC
ME GUSTA EL BASIC
ME GUSTA EL BASIC
:
:

Hasta que borre (RESET) o interrumpa (BREAK) el programa

Bucles 5

ESTOY ALGO TONTO
(15 veces)

Read-Data 1

ESTAMOS COMPARANDO LA SENTENCIA 'READ'
170

Read-Data 2

X = 1
X = 2
:
:
X = 23

La traducción alternativa

Los ordenadores "piensan" según el código máquina; los programadores prefieren escribir en un lenguaje de alto nivel como el Basic. Compiladores e intérpretes ofrecen métodos de traducción que difieren de unos a otros

Al principio los ordenadores no tenían teclado. Las instrucciones del programa tenían que introducirse paso a paso situando cada uno de los ocho interruptores en las posiciones *up* o *down* para representar una sola operación. Estas dos combinaciones constituían el código de la máquina.

Era lógico, entonces, reemplazar los interruptores por un teclado tipo máquina de escribir y sustituir el código por palabras de uso corriente. El resultado fue el lenguaje de "alto nivel", el BASIC, por ejemplo, que ha desplazado a los códigos de lenguaje máquina de bajo nivel.

Como procesadores, sin embargo, los ordenadores no cambiaron, sino que continuaron funcionando con los interruptores originales (y aún lo siguen haciendo). Por ello los programadores tuvieron que desarrollar software escritos en la notación original de bajo nivel para traducir esos programas de alto nivel a una forma que los procesadores pudieran manipular. Estos programas de bajo nivel recibieron la denominación de *compiladores* o *intérpretes*, de acuerdo a su método de traducción.

En informática (como en cualquier otra disciplina), todo aumento de potencia o velocidad debe pagarse —en dinero, tiempo o libertad de acción—. Así sucede con los intérpretes y compiladores. Conjuntamente suministran todas las traducciones de programa que se necesitan. Los intérpretes son más adecuados para unas áreas y los compiladores para otras, pero cada uno paga por sus ventajas con desventajas compensadoras.

Los intérpretes, que normalmente se montan en los ordenadores personales, son la forma más barata de traducir programas de alto nivel a un lenguaje que el ordenador pueda entender. No emplean mucha memoria y dejan más espacio para los programas.

Los micros de precio más asequible utilizan casi siempre un intérprete BASIC: se escribe un programa en BASIC, se teclea RUN, y el programa funciona o se detiene e indica un error del sistema, algo parecido a:

```
ERROR DE SINTAXIS EN LINEA 123
```

Entonces de digita LIST, se halla el error, se corrige, se pulsa RUN y el programa funcionará o se volverá a parar, y así sucesivamente. Hay que tener en cuenta que algunos de los intérpretes BASIC más sofisticados verifican los errores de sintaxis al introducir cada línea.

Lo anterior puede haberse realizado cientos de veces sin haber pensado ni un solo momento en el intérprete. Su virtud principal es precisamente el ser un dispositivo invisible que permite trabajar en el programa sin tener que preocuparse de en qué lugar de la memoria se encuentra localizado o cómo ejecutarlo: el programa está en la punta de los dedos, y puede procesarse, listarse o editarse de inmediato.

El intérprete suele ser fácil de usar, siempre que no sea muy sofisticado: cada vez que se digita RUN, el intérprete tiene que encontrar en la memoria el programa en BASIC, traducirlo y realizarlo línea a línea. Si el programa contiene este bucle:

```
400 LET N = 0
500 PRINT N
600 LET N = N + 1
700 IF N < 100 THEN GOTO 500
```

el intérprete debe traducir y realizar las líneas 500 a 700 cien veces, como si nunca las hubiera procesado.

Los compiladores son diferentes. Son caros, difíciles de escribir, y ocupan y usan mucha memoria. Por lo general forman parte de un software soportado en discos; por tanto se necesita un sistema caro.

Lo que ofrecen es flexibilidad, potencia y velocidad; frente a las cuatro líneas de BASIC anteriores, un compilador las traduciría de una vez, luego realizaría ese código cien veces.

Esto permite ahorrar bastante tiempo, pero tiene un precio. Supóngase que se posee un compilador BASIC y se quiere introducir y procesar un programa en este lenguaje.

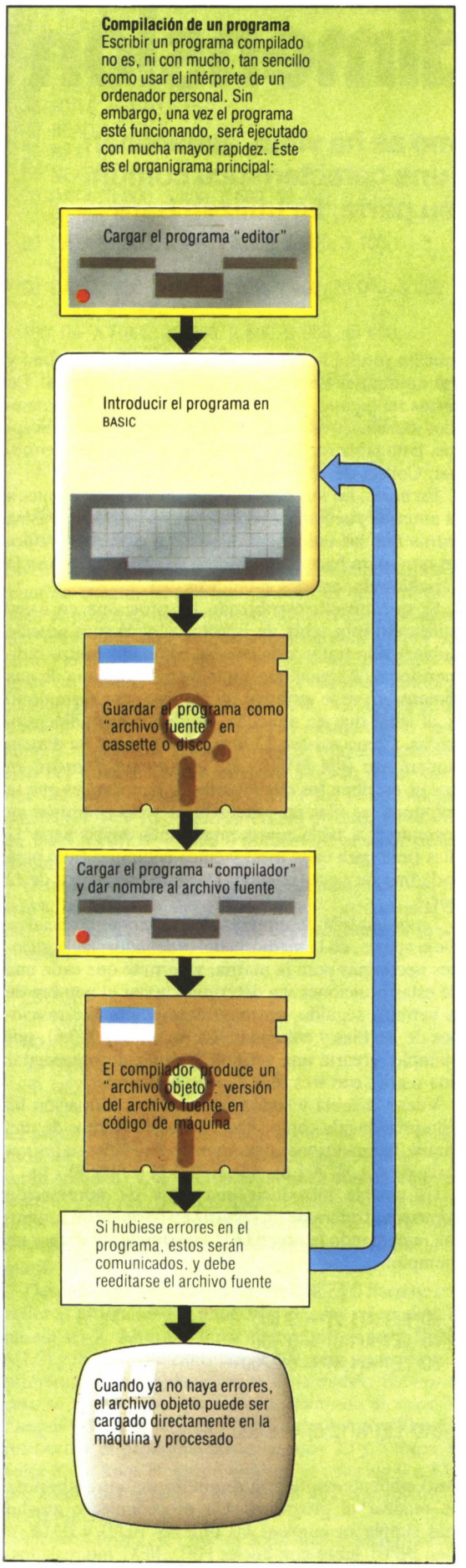
Primero se carga y se pasa un programa de creación de archivos (llamado *editor*), que permite introducir el programa mediante el teclado y guardarlo en disco como *archivo fuente*.

A los archivos se les debe dar un nombre para poder encontrarlos una vez se los ha creado (al igual que sucede con un archivo clásico); por lo tanto, el programa editor pide que se les dé una denominación. Estos nombres por lo general constan de dos partes: la primera es una etiqueta, cualquier nombre que se elija (por ejemplo, MYPROG), y la segunda normalmente es un código de tres letras que indica la naturaleza de los datos contenidos en el archivo; este código se denomina *extensión*. En BASIC, podría tener el código BAS. El archivo fuente está ahora en disco bajo el nombre de MYPROG. BAS. Entonces, al escribir el usuario:

```
COMPILE MYPROG. BAS
```

hará que el ordenador cargue y ejecute el compilador BASIC con un archivo denominado MYPROG. BAS.

Compilación de un programa
Escribir un programa compilado no es, ni con mucho, tan sencillo como usar el intérprete de un ordenador personal. Sin embargo, una vez el programa esté funcionando, será ejecutado con mucha mayor rapidez. Éste es el organigrama principal:



El compilador tarda unos segundos, según sea su longitud, en traducir el programa a un *archivo objeto*, que guarda un disco bajo el nombre de MYPROG.OBJ. La extensión OBJ indica que éste es un archivo objeto, es decir, una traducción a código de máquina de un archivo fuente.

Mientras el compilador traduce el archivo, verifica asimismo sus errores de sintaxis. Si encuentra alguno, enviará un mensaje similar a éste:

```
100 REED X:IF X=3(N + 2) LET P=Q
     1              2       3
FATAL ERROR:-
1) //REED// UNRECOGNISED COMMAND
2) //(//ILLEGAL OPERATOR HERE
3)??'THEN' OR 'GOTO' EXPECTED HERE
```

Este mensaje se produce para cada línea que contiene un error. En otras palabras, la información sobre el error es mucho más extensa que la que proporciona un intérprete BASIC.

Llegados a este punto, hay que volver a cargar y pasar el editor, extraer del disco el archivo fuente, realizar los cambios e intentar compilar otra vez. Si no hay más errores, se puede escribir:

```
RUN MYPROG
```

y el programa funcionará, tal como se espera, o no. En esta etapa ya no hay ningún error de sintaxis, porque el programador ya los ha corregido; no obstante, puede darse el caso de que éste quisiera introducir cambios en el programa, para realizar lo cual este último se carga y se pasa el editor, se cambia el archivo fuente, se recopila... y así sucesivamente.

Las virtudes de un compilador no son evidentes en la fase de desarrollo del programa, aunque las informaciones sobre errores son valiosas. Éstos empiezan a adquirir su real importancia cuando ya se tiene un programa de trabajo y se digita la orden RUN.

Los programas compilados son rápidos. Suelen serlo de 5 a 50 veces más que los programas por intérprete, según sea la eficacia del compilador, aunque la velocidad de ejecución del programa compilado tiene su contrapartida en la fase del desarrollo del programa.

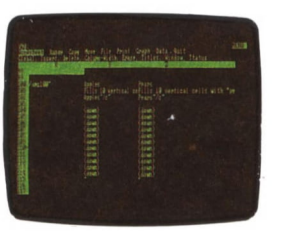
Si se comparan compiladores e intérpretes con respecto a las secuencias descritas más arriba, los compiladores quedan en desventaja, puesto que éstos, por lo general, están escritos para máquinas más potentes y menos especializadas, en las cuales se pueden escribir y procesar los programas en otros muchos lenguajes.

El COBOL (lenguaje de alto nivel, para escribir programas de proceso de datos comerciales para control de cuentas, nóminas e inventarios), por ejemplo, fue inventado pensando en utilizar un compilador, mientras que el BASIC requiere realmente un intérprete.

Una vez se ha desarrollado y compilado un programa, el archivo fuente ya no se necesita más que como referencia. En consecuencia, el programa fuente puede ser totalmente comentado y escrito con el propósito de que sea legible, mientras que el archivo objeto puede ser mucho más pequeño y ocupar menos espacio en el disco y la memoria.

El hecho de que los archivos objeto creados por un compilador se compongan de código de máquina ilegible, puede ser, sorprendentemente, una ventaja. Si se comercializa software, no se venderá el archivo fuente, sino sólo el archivo objeto, lo cual hará que sea mucho más difícil hacerlo objeto de piratería, copiarlo o alterarlo.

Lento
En un juego de aventuras, la velocidad no es primordial, y la mayor parte del programa consiste en la manipulación de un texto. Éste está escrito en BASIC, y es interpretado mientras se procesa



Más rápido
Muchos programas de gestión (en particular, hojas electrónicas) son difíciles de escribir en código máquina, debido a que contienen muchas operaciones matemáticas. Sin embargo, un lenguaje de intérprete sería demasiado lento. Por ello, frecuentemente, están escritos en BASIC y luego son compilados

El más rápido
Para los juegos recreativos de acción rápida, que requieren gráficos, incluso un programa compilado sería moroso. Estos paquetes deben escribirse directamente en código de máquina: una tarea lenta y engorrosa

Una nueva dimensión

Las matrices unidimensionales, como se ha visto, almacenan cierta cantidad de datos que tienen una característica común. Las matrices bidimensionales, por su parte, se utilizan para confeccionar tablas y cuadros

Hasta ahora se han considerado dos tipos de variables: simples y subíndice. Las primeras son semejantes a posiciones de memoria, en las cuales pueden almacenarse y manipularse números (o series de caracteres) refiriéndose a la variable "etiqueta". Las variables simples únicamente pueden almacenar un valor o serie y tienen nombres de variable "simple" —por ejemplo, N, B2, X, Y3—. Las variables subíndice, a veces llamadas matrices unidimensionales, pueden almacenar una lista entera de valores o series. El número de valores o series que tienen cabida se especifica al principio del programa mediante la orden DIM. Por ejemplo, DIM A(16) significa que la matriz denominada A puede contener 16 valores independientes. Debe tenerse en cuenta, sin embargo, que muchos lenguajes BASIC aceptan A(0) como primer elemento, y por consiguiente DIM A(16) define en realidad a 17 elementos. Para hacer referencia a estas "posiciones" se debe emplear el subíndice apropiado. PRINT A(1) imprimirá el primer elemento de la matriz: LET B = A(12) asigna el valor del decimosegundo elemento de la matriz a la variable B; LET A(3) = A(5) traspasa el valor del quinto elemento al tercero.

Algunas veces, sin embargo, es necesario poder manejar datos cuya mejor forma de representación es mediante tablas. Adviértase que tienen una gran semejanza a una hoja electrónica. Los datos a representar pueden ser muy diversos, abarcando desde los resultados de los partidos de fútbol hasta la clasificación de ventas de unos grandes almacenes por artículos y departamentos. Como ejemplo de una tabla de datos típica, véase el siguiente desglose de los gastos de una familia durante el período de un año:

	Alquiler	Teléfono	Electricidad	Alimentación	Coche
ENE	20 000	3 800	3 500	52 700	8 300
FEB	20 000	3 500	2 900	51 000	6 800
MAR	20 000	3 400	3 100	49 900	7 000
ABR	20 000	3 300	3 000	50 000	9 100
MAYO	20 000	3 400	4 300	47 000	7 900
JUN	20 000	3 600	3 000	48 600	6 100
JUL	20 000	3 500	3 200	47 000	8 500
AGO	20 000	3 400	3 100	49 500	7 400
SEPT	20 000	3 300	3 000	46 300	6 900
OCT	20 000	3 700	3 800	52 800	7 000
NOV	20 000	3 900	3 800	53 000	7 500
DIC	20 000	4 100	4 700	55 900	8 600

El disponer la información de esta manera permite que sea tratada de forma relativamente simple. Es fácil, por ejemplo, hallar el total de los gastos del mes de marzo con sólo sumar todas las cantidades que figuran en la línea correspondiente a este mes. Igual de sencillo resulta hallar los costes anuales de teléfono y del automóvil sumando las columnas verticales. De forma semejante, no resulta difícil conocer los promedios mensuales o anuales de gastos por dichos conceptos. Esta tabla recibe el nombre de matriz bidimensional. Consta de 12 filas y 5 columnas.

En BASIC, las matrices bidimensionales semejantes a la anterior pueden ser representadas casi de la misma forma que las unidimensionales. La diferencia radica en que, para hacer referencia a su situación, ahora la variable requiere dos subíndices.

Si se estuviera escribiendo un programa en BASIC utilizando esta tabla de información, lo más sencillo hubiera sido tratar toda la tabla como una matriz bidimensional. Al igual que a una matriz ordinaria de una dimensión, se le atribuye un nombre, por ejemplo A. Y al igual que en aquéllas, será necesario DIMensionarlas. Como existen 12 filas y 5 columnas, se dimensionará así: DIM A(12,5). Es importante el orden en que se escriben los dos subíndices; la norma es que se coloquen las filas en primer lugar y las columnas en segundo. La tabla representada más arriba tiene 12 filas (una para cada mes) y cinco columnas (una para cada tipo de gasto), siendo por tanto una matriz de 12 por 5.

La orden DIM desempeña dos funciones esenciales. Deja aparte, en la memoria del ordenador, las posiciones necesarias para la matriz, y permite que cada una de estas posiciones sea determinada por el nombre de la variable seguido, entre paréntesis, por las posiciones de las filas y columnas. La orden DIM X(3,5), por ejemplo, crearía una variable X capaz de representar una matriz con tres filas y cinco columnas.

Véase la tabla y supóngase que la información ha sido introducida como si fueran los elementos de una matriz bidimensional denominada A. Hállense los valores de A(1,1), A(1,5), A(2,1), A(3,3) y A(12,3).

Es posible introducir una tabla de información como si se tratara de una matriz en parte de un programa, empleando la orden LET; a continuación damos un ejemplo:

```
30 LET A(1,2) = 3 800
40 LET A(1,3) = 3 500
50 LET A(1,4) = 52 700
 :
 :
610 LET A(12,5) = 8 600
```

Pero evidentemente ésta es una forma muy laboriosa de realizar el programa. Un procedimiento mucho más simple es emplear las órdenes READ y DATA, o bien INPUT junto con bucles FOR...NEXT intercalados.

Así, pues, veamos cómo podría hacerse utilizando la orden READ:

```
10 DIM A(12,5)
20 FOR F = 1 TO 12
30 FOR C = 1 TO 5
40 READ A(F,C)
50 NEXT C
60 NEXT F
70 DATA 20 000, 3 800, 3 500, 52 700, 8 300,
   20 000, 3 500, 2 900
80 DATA 51 000, 6 800, 20 000, 3 400, 3 100,
   49 900, 7 000, 20 000
90 DATA 3 300, 3 000, 50 000, 9 100, 20 000, 3 400,
   4 300, 47 000
100 DATA 7 900, 20 000, 3 600, 3 000, 48 600,
    6 100, 20 000, 3 500
110 DATA 3 200, 47 000, 8 500, 20 000, 3 400,
    3 100, 49 500, 7 400, 20 000
120 DATA 3 300, 3 000, 46 300, 6 900, 20 000,
    3 700, 3 800
130 DATA 52 800, 7 000, 20 000, 3 900, 3 800,
    53 000, 7 500
140 DATA 20 000, 4 100, 4 700, 55 900, 8 600
150 END
```

En este programa existen varios puntos importantes a tener en cuenta. El primero de ellos es que DIM está justo al principio del programa. Una orden DIM debe efectuarse una sola vez en un programa y, por lo tanto, lo normal es colocarla cerca del principio o antes de realizar ningún bucle. El segundo punto a considerar es que hay dos bucles FOR...NEXT, uno para situar la "fila" y otro para hacer lo mismo con la "columna". Estos dos bucles no están uno a continuación del otro, sino que uno está alojado en el interior del otro. Ténganse en cuenta los límites elegidos. FOR F = 1 TO 12 incrementará el valor de la fila de 1 a 12; FOR C = 1 TO 5 aumentará el valor de la columna de 1 a 5.

Exactamente en el centro del segundo bucle —que está alojado en el interior del primero— está situada la orden READ. La parte decisiva del programa es la siguiente:

```
20 FOR F = 1 TO 12
30 FOR C = 1 TO 5
40 READ A(F,C)
50 NEXT C
60 NEXT F
```

La primera vez que se procese, después de haber realizado las líneas 20 y 30, los valores asignados a F y C serán ambos 1, y, por lo tanto, la línea 40 será equivalente a READ A(1,1). La primera cantidad de la orden DATA es 20 000, y, en consecuencia, este valor será asignado a la primera fila y a la primera columna de la matriz.

Después que haya sucedido esto, NEXT C hace retroceder el programa a la línea 30 y el valor asignado a C pasa a ser 2. Ahora la línea 40 es equivalente a READ A(1,2) y la siguiente cantidad, 3 800, será asignada a la primera fila y segunda columna de la matriz. Este proceso se repite hasta que C haya alcanzado el valor 5. Después de ello, la orden NEXT F de la línea 60 hace retroceder el programa hasta la línea 20 y F toma el valor 2. La línea 30 ajusta otra vez el valor de C a 1 y, por consiguiente, ahora la línea 40 será equivalente a READ A(2,1).

De esta forma, los bucles anidados en esta manera son muy útiles, pero es necesario poner mucha atención. Cada bucle debe estar alojado completamente en el interior del otro y debe observarse cuidadosamente la correlación de las órdenes NEXT. Obsérvese cómo el primer bucle, FOR F, incluye la segunda orden NEXT. Cuando existen dos bucles, uno de ellos alojado en el interior del otro, el primero recibe el nombre de bucle exterior, y el segundo, el de bucle interior. Todo el bucle interior será siempre completado antes de que sea incrementado el índice del bucle exterior. Es posible introducir bucles en el interior de otros hasta la "profundidad" requerida por el programa, pero éste puede resultar complejo y difícil de seguir. Es una mala práctica colocar en el interior de los bucles instrucciones que hagan desviar el programa principal; asimismo debe evitarse emplear GOTO.

Veamos la orden DATA. Obsérvese que, para separar los datos de las cantidades, se emplean comas, pero que no tiene que haber coma antes de la primera cantidad o después de la última. Entre cada cantidad se han insertado espacios, pero esto no es lo normal. Al introducir los datos, es fácil cometer errores, que luego son difíciles de descubrir.

Se pueden utilizar tantas órdenes DATA como sean necesarias. Cada línea nueva debe empezar con una DATA. Cada vez se lee una cantidad, empezando por el principio de la primera DATA y continuando hasta que se hayan leído todas. Compruébese que el número de datos sea correcto, pues, en caso contrario, al pasar el programa se obtendrá una información errónea.

El programa presentado hasta ahora, realmente, no hace más que transformar los datos y disponerlos en forma de una matriz bidimensional. Después de introducir el programa y pulsar RUN, aparentemente no sucederá nada, y lo único que se verá en la pantalla será el esquema BASIC. Para comprobar que los datos están bien situados, deben probarse algunas órdenes PRINT. (En BASIC, una orden es una palabra clave que puede ejecutarse inmediatamente, sin tener que utilizar el programa, y, por tanto, no necesita un número de línea. Por ejemplo, LIST, RUN, SAVE, AUTO, EDIT y PRINT.) PRINT A(1,1) <CR> hará que el número 20 000 aparezca en la pantalla. ¿Qué se imprimirá con las siguientes órdenes?

```
PRINT A(12,1)
PRINT A(1,5)
PRINT A(5,1)
PRINT A(5,5)
```

Para lograr que el programa sea de utilidad, es necesario ampliarlo. En su forma actual constituye una base adecuada para un "programa principal". Para usarlo como parte de un programa mayor y más útil, pueden escribirse módulos como si constituyeran subrutinas, que serán empleadas por diversos GOSUB, insertados en puntos adecuados antes de la orden END.

En los primeros pasos del diseño de un programa de gastos domésticos, es mejor empezar con una simple descripción escrita de los requisitos generales. Puede decidirse que se quiere obtener el total de gastos y los promedios por meses, o bien para cada categoría (electricidad, por ejemplo). Se pueden desarrollar los detalles de cómo derivar estos resultados a una etapa posterior. Si se tiene que efectuar una elección en el programa sobre qué subrutinas se desea que sean realizadas, probablemente se requerirá la guía de un "menú", que controle directamente las subrutinas apropiadas. Un primer bosquejo del programa, en

esta fase de desarrollo, puede ser semejante al siguiente:

```
PROGRAMA PRINCIPAL
(ENTRADA DE DATOS)

MENU
(SELECCION DE SUBRUTINAS)

END
```

Al desarrollar el programa con algo más de precisión, se verá la necesidad de emplear subrutinas para calcular totales por meses (TOTALMES) o por categorías (TOTALCAT), los gastos promedios mensuales (PROMMES) y los promedios anuales por categorías (PROMCAT). La razón de utilizar nombres de una sola palabra para denominar estas subrutinas es ayudar a planear el programa sin tener que preocuparse, en esta fase, de detalles como dar un número de línea. Si lo pensamos detenidamente, quizá decidamos que incluso la parte principal del menú del programa debería tratarse como una subrutina con el fin de mantener la sección principal del programa como un módulo independiente. La siguiente etapa para perfilar el programa constará de los siguientes pasos:

```
PROGRAMA PRINCIPAL (ENTRADA DE DATOS)
MENU (LLAMADA A SUBRUTINAS)
END

**SUBRUTINAS**

1 MENU
2 TOTALES
3 PROMEDIOS

(2) TOTALES
4 TOTALMES
5 TOTALCAT

(3) PROMEDIOS
6 PROMMES
7 PROMCAT
```

Este esquema de programa muestra que la subrutina de MENU ofrece una elección entre TOTALES y PROMEDIOS. Ambas serán subrutinas. La subrutina de TOTALES ofrece una nueva elección entre TOTALMES y TOTALCAT. Éstas serán las subrutinas que efectuarán los cálculos reales.

La subrutina de PROMEDIOS permite elegir entre PROMMES y PROMCAT, las cuales, asimismo, constituyen las subrutinas que efectuarán los cálculos adecuados. En este punto, debe ser posible ver si el "programa" hará lo que se le pide, sin desarrollar ningún código real (programa detallado escrito en BASIC). Si hasta aquí se han realizado todas las etapas correctamente, ya se está en condiciones de abordar la escritura de los módulos (subrutinas). El único cambio que debe hacerse en el programa principal será una llamada de subrutina antes de END, es decir, habría que añadir:

```
145 GOSUB **MENU**
```

Obsérvese que se continúa empleando "nombres" para designar las subrutinas, en vez de asignarles números de línea. Numerosos lenguajes, el PASCAL, por ejemplo, permiten designar con nombres a los subprogramas, pero, en cambio, ello no es posible en la mayoría de las versiones de BASIC, que, por el contrario, necesitan utilizar números de línea. Sin embargo, tomaremos en consideración estos detalles más adelante.

Veamos cómo puede escribirse la subrutina de MENU (se han omitido los números de línea pero el usuario puede añadirlos si desea realizar este programa).

```
REM SUBRUTINA **MENU**
PRINT "¿DESEA T(OTALES) O P(ROMEDIOS)?"
PRINT "DIGITAR T O P"
INPUT L$
IF L$ = "T" THEN GOSUB *TOTALES*
IF L$ = "P" THEN GOSUB *PROMEDIOS*
RETURN
```

Nota: Las llamadas a subrutinas se están acotando entre asteriscos: *— —*. Al programar, en vez de ello, se deberán emplear números de línea, que podrán incorporarse cuando se esté en condiciones de conocer cuáles son.

Imaginemos que se teclea T, inicial de TOTALES. Entonces el programa pedirá la subrutina de TOTALES. Ésta presentará otro menú, semejante al siguiente:

```
REM SUBRUTINA DE **TOTALES**
PRINT "¿DESEA TOTALES DE?"
PRINT "¿M(ES) O C(ATEGORIA)?"
PRINT "DIGITAR M O C"
INPUT L$
IF L$ = "M" THEN GOSUB *TOTALMES*
IF L$ = "C" THEN GOSUB *TOTALCAT*
RETURN
```

Supongamos que se elige M, indicativo de TOTALMES. Veamos cómo se puede escribir un módulo para calcular los gastos totales para cualquiera de los meses del año.

```
REM SUBRUTINA DE **TOTALMES**
REM CALCULA LOS GASTOS TOTALES DE
REM CUALQUIER MES
PRINT "ELEGIR MES"
PRINT "1-ENE 2-FEB 3-MAR 4-ABR 5-MAYO"
PRINT "6-JUN 7-JUL 8-AGO 9-SEPT"
PRINT "10-OCT 11-NOV 12-DIC"
PRINT "DIGITAR EL NUMERO DEL MES"
LET T = 0
INPUT M
FOR C = 1 TO 5
LET T = T + A(M,C)
NEXT C
PRINT "LOS GASTOS TOTALES DEL MES"
PRINT "NUMERO";M;"SON";T
RETURN
```

Se digita el número que representa el mes, y la orden INPUT asigna el número a la variable M(MES). M se emplea para especificar el subíndice de la "fila" de la matriz bidimensional A. El bucle FOR-NEXT incrementa el valor de C (columna) desde uno hasta cinco y, por consiguiente, la primera vez que se pasa el bucle, si se había elegido tres, es decir, el mes de marzo, la orden LET será equivalente a LET T = T + A(3,1). La próxima vez que se utilice el bucle, será equivalente a LET T = T + A(3,2), y así sucesivamente.

En este capítulo le encomendamos a usted la tarea de escribir las otras subrutinas o tratar de solucionar los otros ejercicios. Las matrices bidimensionales son ideales para cualquier programa que contenga tablas

Respuestas a los "Ejercicios" de la página 131

Función RND

```
40 IF R > 6 THEN LET R = 1
```

Bucle y promedio

```
5  FOR L = 1 TO 100
   :
   :
80 LET T = T + R
90 NEXT L
100 LET A = T/100
110 END
```

Sustitución por una subrutina

Suprima las líneas 5, 80, 90, 100 y 110 en la solución anterior. Cambie las líneas 10 a 70 por (pongamos por caso) las 1000 a 1070. Verifique que la línea 40 es igual que en la función RND de la solución anterior. Luego añada 1080 RETURN. Incorpore el resultado en el programa principal. Cambie las líneas 50 y 130 para que se lea 50 GOSUB 1000 y 130 GOSUB 1000.

INKEY$

```
10 PRINT "PULSE CUALQUIER TECLA"
20 LET A$ = INKEY$
30 IF A$ = " " THEN GOTO 20
40 PRINT "LA TECLA QUE USTED HA PULSADO ES";A$
50 END
```

Bucle de tiempos

```
5  PRINT "PULSE LA BARRA ESPACIADORA DESPUES DE
   10 SEGUNDOS"
10 FOR L = 0 TO 1
20 LET R = R + 1
30 IF INKEY$ = " " THEN GOTO 60
40 LET L = 0
50 NEXT L
60 PRINT "EL VALOR DE R DESPUES DE
   10 SEGUNDOS ES";R
70 END
```

Comparaciones IF... THEN

```
10 GOSUB 1000
20 PRINT "ADIVINE EL NUMERO"
30 FOR G = 1 TO 50
40 INPUT N
50 IF N > R THEN GOTO 110
60 IF N < R THEN GOTO 130
70 IF N = R THEN GOTO 150
80 NEXT G
90 PRINT "YA NO TIENE MAS OPORTUNIDADES.
   ¡HA PERDIDO!"
100 GOTO 500
110 PRINT "SU ELECCION ES DEMASIADO
    ELEVADA"
120 GOTO 80
130 PRINT "SU ELECCION ES DEMASIADO BAJA"
140 GOTO 80
150 PRINT "HA ACERTADO, FELICITACIONES"
500 END
1000 REM **SUBRUTINA ALEATORIA**
     (Introduzca aquí su subrutina)
1020 RETURN
```

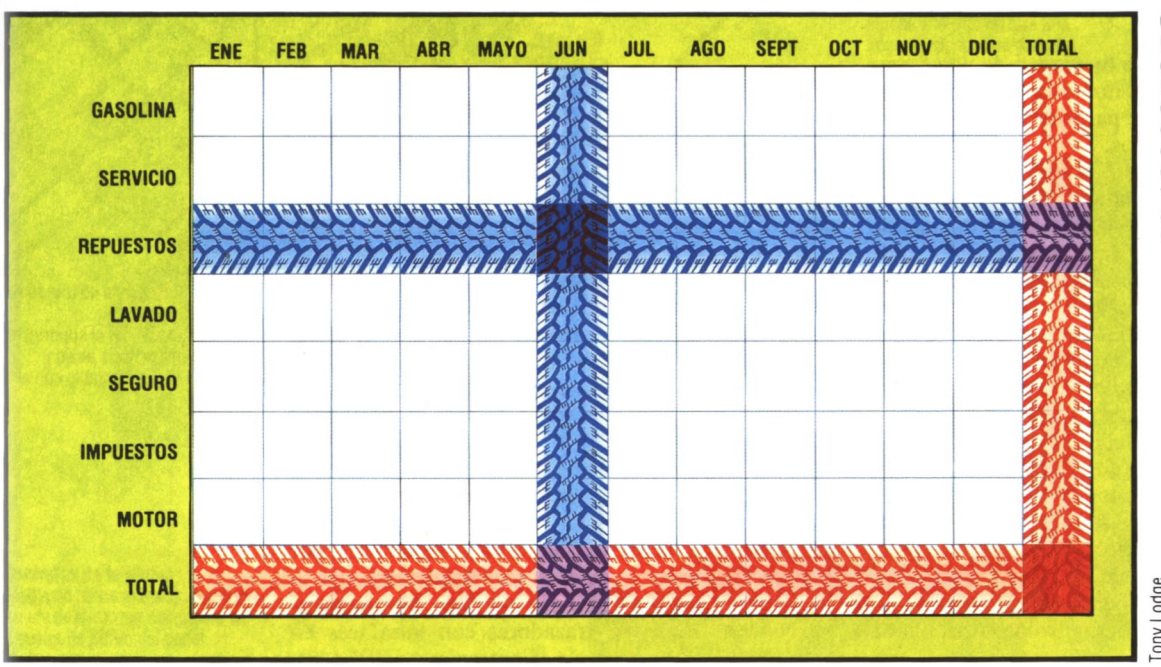
de datos, ya sean éstos estadísticos, financieros o de cualquier otro tipo.

Ejercicios

■ **Asignación de valores** Escriba un programa que asigne valores a los elementos ("gasolina", "servicio", etc.) de la matriz representada en la parte inferior de esta página. A continuación, escriba una subrutina que requiera un mes y uno de los elementos e imprima el contenido del cuadrado determinado por ambos. Por último, escriba una subrutina que calcule el total de cada columna y sitúe el resultado al pie del cuadrado, haga lo mismo con cada fila y calcule el total final, que deberá colocar en el cuadrado inferior derecho.

■ **Bugs** El programa siguiente no funcionaría adecuadamente y produciría un mensaje erróneo. Descubrir el error y efectuar la corrección pertinente.

```
10 DIM A(3,4)
20 FOR F = 1 TO 3
30 FOR C = 1 TO 4
40 READ A(F,C)
50 NEXT C
60 NEXT F
70 FOR X = 1 TO 3
90 FOR Y = 1 TO 4
100 PRINT A(Y,X)
110 NEXT Y
120 NEXT X
130 DATA 2,4,6,8,10,12,14,16,18,20,22
140 END
```



Gastos de un coche
La figura muestra una cuadrícula de 8 × 13 cuadrados. Las filas representan elementos diferentes del coste de mantenimiento de un coche, y las columnas los diferentes meses del año. Realizar el ejercicio "Asignación de valores" para calcular el coste anual de mantenimiento de un coche

Tony Lodge

Trazadores de gráficos

Mediante estos dispositivos, un ordenador puede producir gráficos de alta calidad. Funcionan con plumas de punta de fibra y algunos de ellos pueden cambiar automáticamente de color

La capacidad de crear copias impresas de los diagramas que aparecen en la pantalla de un ordenador es un requisito esencial para muchas personas que utilizan la máquina profesionalmente. Ingenieros, científicos, diseñadores y hombres de negocios necesitan diagramas y cuadros con un grado de precisión que no pueden suministrar las impresoras convencionales. El único dispositivo que puede crear esas imágenes es un trazador de imágenes, y, hasta hace poco tiempo, éste resultaba excesivamente caro para el usuario del ordenador personal.

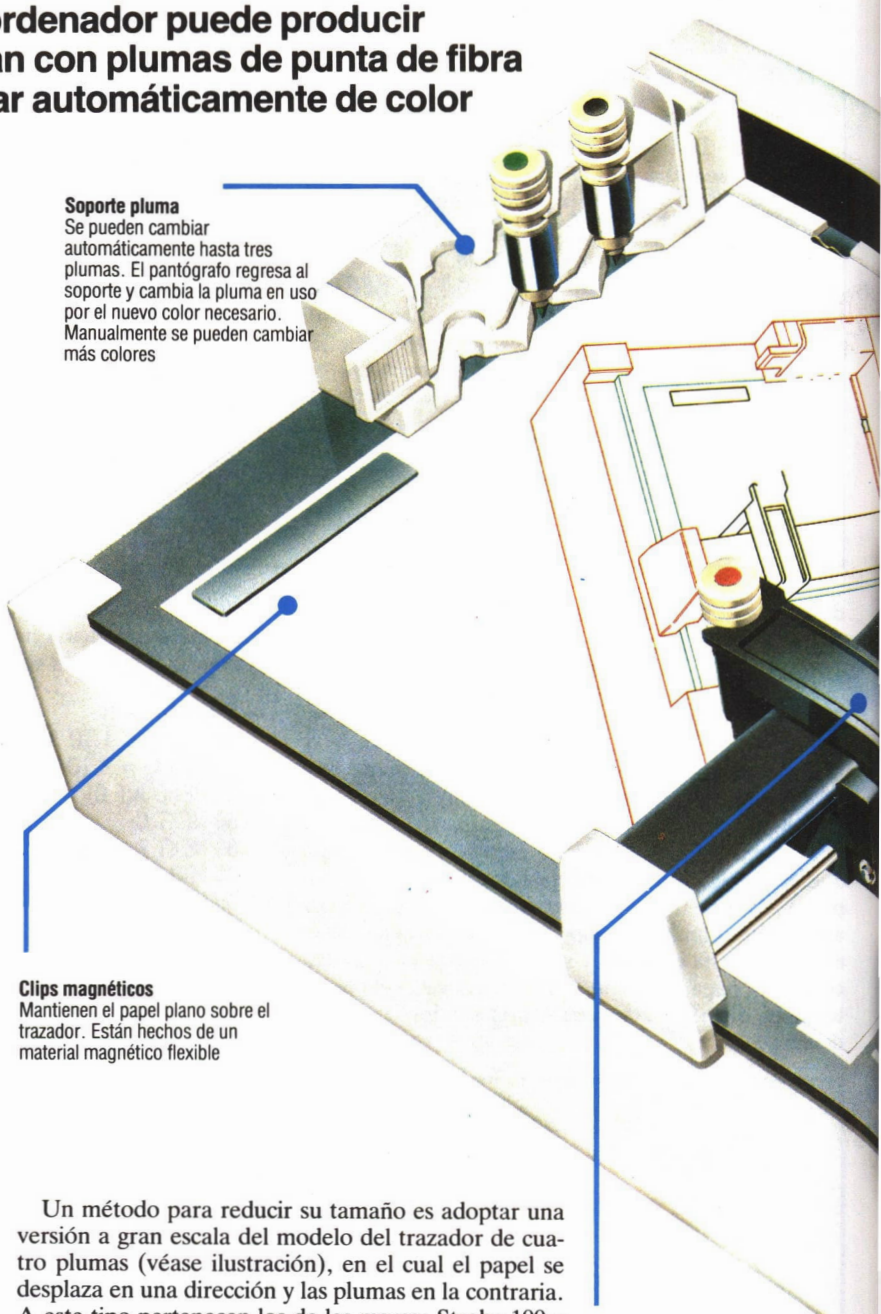
Sin embargo, con la introducción de dispositivos como el trazador de gráficos-impresora de cuatro plumas utilizado por el Tandy/CGP-115 y el Oric MCP-40, un terminal de gráficos puede estar al alcance de una gran parte de los usuarios. Recientemente ha aparecido en el mercado una gama completa de trazadores de gráficos con unas características que antes sólo podían ofrecer al usuario máquinas cuyo precio era prohibitivo.

La necesidad de utilizar un trazador de gráficos, o *plotter*, generalmente está determinada por el tipo de trabajo a que está destinado el ordenador. Un ingeniero o un proyectista necesitarán dibujos precisos de equipos y montajes, en cambio un hombre de negocios deseará cuadros y gráficos que muestren los volúmenes de ventas. Realizar esto con impresoras convencionales es un proceso muy laborioso y los resultados sólo aparecerán en blanco y negro. La otra única opción de bajo coste es efectuar una fotografía de la pantalla, pero si bien esto puede ser suficiente para gráficos comerciales, no reúne los requisitos mínimos de precisión que pedirían un arquitecto o un diseñador.

Los trazadores de gráficos funcionan de una forma completamente distinta a las impresoras: trazan líneas entre dos puntos en lugar de partir de formas preestablecidas o modelos de puntos.

El principio básico con el que funcionan todas las marcas consiste en un sistema de coordenadas X, Y. Al igual que una gráfica puede ser trazada definiendo las coordenadas por las que debe pasar la línea, también una figura puede ser descompuesta en una serie de coordenadas. Para poder unir estas coordenadas con el fin de recrear la figura, tiene que existir alguna forma de movimiento. Por ello se fija la pluma a un pantógrafo que puede desplazarse en el sentido de las abscisas X (de izquierda a derecha) al mismo tiempo que la pluma se mueve a lo largo del pantógrafo en el sentido de las ordenadas Y (de arriba abajo).

El tipo tradicional de trazador de gráficos se conoce con el nombre de *lecho plano*, debido a que el papel es fijado a una placa plana, sobre la cual se desplaza el pantógrafo, tal como se puede apreciar en la ilustración. Esto tiene el inconveniente de que el trazador debe ser, como mínimo, tan grande como la hoja de papel.



Soporte pluma
Se pueden cambiar automáticamente hasta tres plumas. El pantógrafo regresa al soporte y cambia la pluma en uso por el nuevo color necesario. Manualmente se pueden cambiar más colores

Clips magnéticos
Mantienen el papel plano sobre el trazador. Están hechos de un material magnético flexible

Portaplumas
La pluma que se está utilizando se fija (en este caso magnéticamente) en el soporte, que se desplaza hacia abajo y sitúa la pluma en contacto con el papel

Un método para reducir su tamaño es adoptar una versión a gran escala del modelo del trazador de cuatro plumas (véase ilustración), en el cual el papel se desplaza en una dirección y las plumas en la contraria. A este tipo pertenecen los de las marcas Strobe 100 y Hewlett Packard Sweetlips. El movimiento del papel debe ser controlado con la misma precisión que el desplazamiento del pantógrafo en los del tipo de lecho plano, y esto se logra con un motor paso a paso. Este tipo de motor posee unas características muy especiales, y cada vez que recibe un pulso de la potencia aplicada, realiza una fracción de giro. Se aplica principalmente a unidades de disco, en las cuales controla la posición del cabezal sobre la superficie del disco, y en robótica.

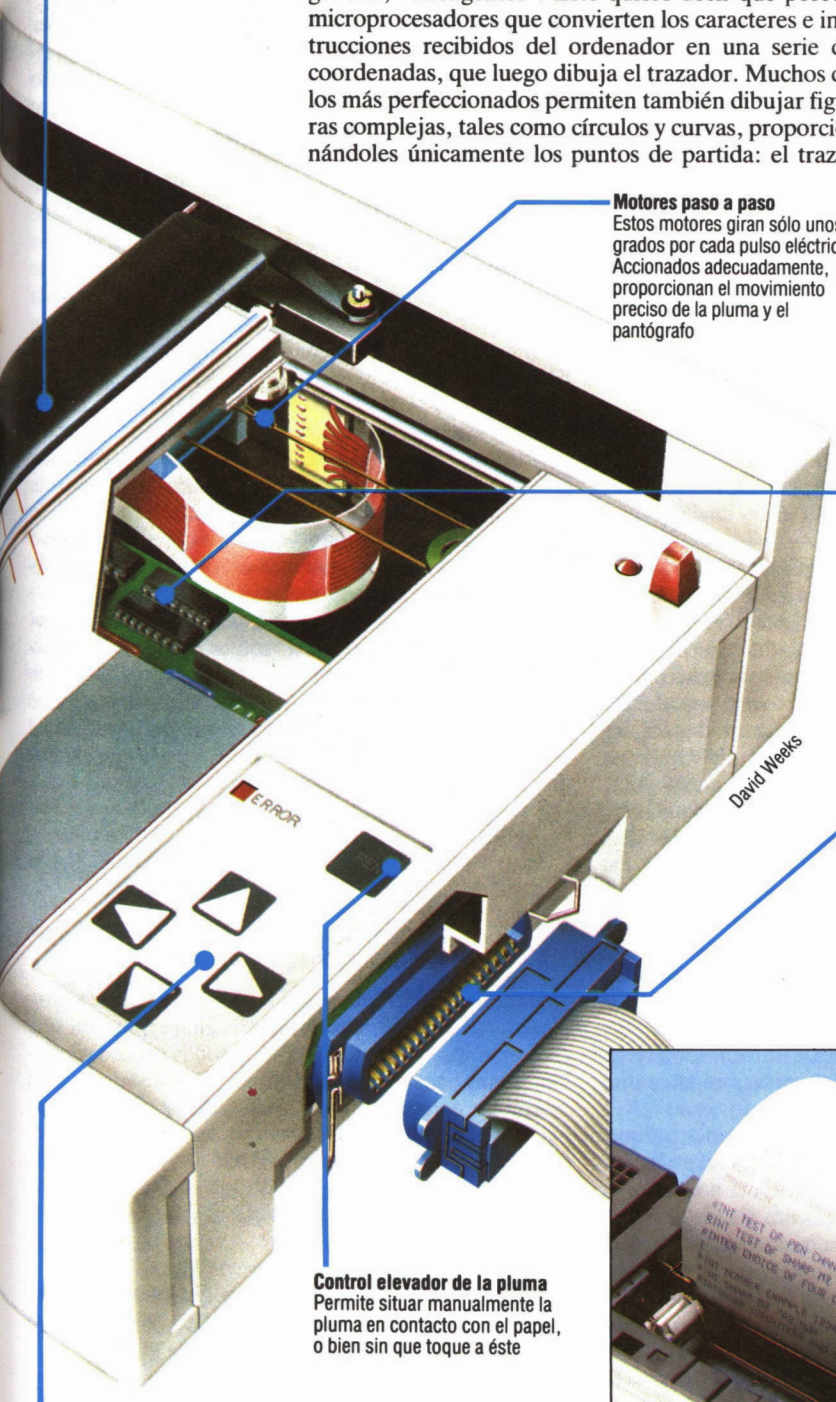
La conexión de un trazador a un ordenador es, por lo general, similar a conectar una impresora, por lo menos en lo que se refiere a la interface. Normalmente, pueden adquirirse trazadores con interfaces en serie (RS232) o en paralelo (Centronics o IEEE488),

que pueden enchufarse a la conexión utilizada por las impresoras. A menudo, su programación es algo más complicada. En vez de sólo enviarle los resultados de un programa que debe ser impreso, ahora será necesario suministrarle también información sobre cómo deben ser presentados. Por lo común, esto se realiza de una forma muy parecida a la empleada para formar un diagrama en la pantalla.

Debido al complicado sistema que utilizan los trazadores para realizar su cometido, éstos son, por regla general, "inteligentes . Esto quiere decir que poseen microprocesadores que convierten los caracteres e instrucciones recibidos del ordenador en una serie de coordenadas, que luego dibuja el trazador. Muchos de los más perfeccionados permiten también dibujar figuras complejas, tales como círculos y curvas, proporcionándoles únicamente los puntos de partida: el trazador hará el resto. El rotulado de los gráficos y diagramas y el coloreado de diagramas de barras y de segmentos circulares son, frecuentemente, procesos automáticos, factor que hace la programación mucho más sencilla.

Muchos trazadores se suministran junto con el software, lo cual les permite estar conectados directamente al interior del programa, en vez de ser como una copia sobre papel de la pantalla. Si no se dispone de este tipo de programas, el usuario debe desarrollar las rutinas necesarias para transformar la información de la pantalla en los códigos adecuados para guiar al trazador. Algunos de ellos no disponen de juegos de caracteres y, por consiguiente, deben crearse incluso los códigos para las letras y números. En cambio, esto permite al usuario crear sus propios caracteres y tipos.

Una vez que se ha generado una figura, ésta puede trazarse en cualquier posición, orientación o tamaño, con lo cual puede obtenerse una colección de formas para diferentes usos. Con frecuencia son muy útiles las rutinas para trazar círculos, curvas y figuras en secciones de gráficas, especialmente en el campo de los gráficos comerciales. Sin embargo, los principios de creación de un dibujo a partir de las coordenadas de la pantalla son exactamente los mismos que para crear la figura sobre el papel y, por lo tanto, la programación es bastante sencilla.



Pantógrafo
Puede situarse en cualquier punto a lo largo de la hoja (eje de las X) y el portaplumas es desplazado según el eje de las Y. La combinación de los movimientos de izquierda a derecha y de atrás adelante permite llegar a cualquier punto de la página

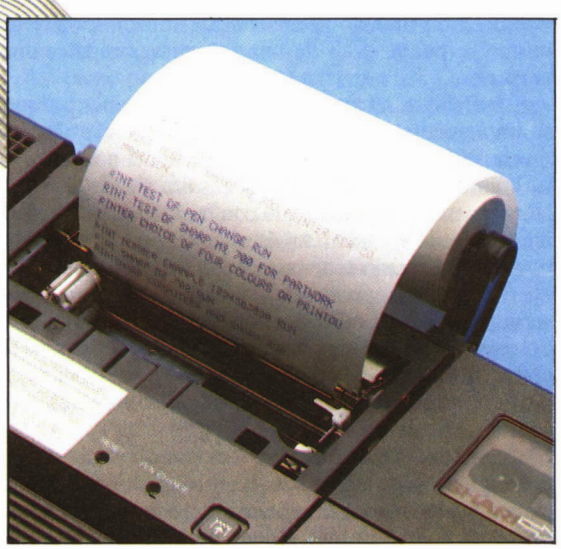
Motores paso a paso
Estos motores giran sólo unos grados por cada pulso eléctrico. Accionados adecuadamente, proporcionan el movimiento preciso de la pluma y el pantógrafo

Tablero de circuitos
Normalmente, los plotters son dispositivos "inteligentes" (pueden dárseles órdenes de alto nivel, por ejemplo: "dibujar una circunferencia de radio y centro determinados", y el trazador mueve la pluma para dibujar ésta). El tablero de circuitos dispone de su propio microprocesador, ROM y RAM

Conexión interface
Los trazadores se conectan al ordenador mediante una interface estándar, por ejemplo la RS232 (en serie) o Centronics (en paralelo). Para el ordenador significa lo mismo que una impresora, aunque para su funcionamiento serán necesarias órdenes distintas

Control elevador de la pluma
Permite situar manualmente la pluma en contacto con el papel, o bien sin que toque a éste

Controles de la pluma
Mediante estos mandos, la pluma puede desplazarse manualmente a cualquier punto del papel



Trazador de gráficos-impresora de cuatro plumas

Este mecanismo atrajo la atención de la industria cuando apareció por primera vez en la impresora Sharp CE-150. Sus hermanos mayores: el Tandy's CGP-115 y el Oric MCP-40, ayudaron a que el coste de las impresoras en color fuera asequible para el usuario del ordenador doméstico.
Como todas las buenas ideas, el sistema es conceptualmente muy simple. Un rollo de papel es empujado a través del mecanismo por medio de un piñón. El papel es desplazado, hacia adelante y hacia atrás, en intervalos muy precisos, mientras un portaplumas que sujeta los cursores de los cuatro colores se desplaza por la superficie, de izquierda a derecha y viceversa.
El portaplumas, para crear el texto o el gráfico, gira hasta enfrentar el color correcto y luego la pluma es presionada contra el papel. Las líneas horizontales se obtienen por el desplazamiento de la pluma y con el papel inmóvil; las verticales, con el papel en movimiento y la pluma estática. Las combinaciones de ambos movimientos producen diagonales o curvas. La calidad de la impresión es notable, pero la limitada anchura del papel la hace inadecuada para procesamiento de textos u otros fines profesionales

Información clasificada

Al diseñar cualquier programa que suponga almacenamiento o manipulación de información, es importante atender a la estructura, de modo que no sea preciso duplicar ningún dato

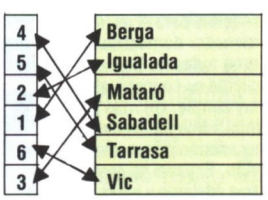
El camino más corto
Una empresa tiene intereses en seis poblaciones y todos los meses ha de enviar un camión a recorrerlas. El ordenador de la empresa puede elaborar la ruta más conveniente.

Ciudad$(1)	"Berga"
Ciudad$(2)	"Igualada"
Ciudad$(3)	"Mataró"
Ciudad$(4)	"Sabadell"
Ciudad$(5)	"Tarrasa"
Ciudad$(6)	"Vic"

Los nombres de las ciudades están incluidos en un archivo secuencial en cinta por orden alfabético. Se leen por este orden en el archivo de la matriz Ciudad$( ). El ordenador calcula que el orden más adecuado es:

Sabadell	Ciudad$(4)
Tarrasa	Ciudad$(5)
Igualada	Ciudad$(2)
Berga	Ciudad$(1)
Vic	Ciudad$(6)
Mataró	Ciudad$(3)

En lugar de volver a almacenar los nombres de las ciudades por ese orden, utilizando más memoria, el ordenador almacena solamente los números de las ciudades en la matriz Ciudad$( ), de la siguiente manera:



La lista de números es un índice de la matriz Ciudad$( ). Se podrían crear otros índices de este tipo para la matriz, cada uno de los cuales ordenaría los nombres de distinta manera. La ventaja de confeccionar índices como éste es que no se debe clasificar ni duplicar el archivo original: sólo se necesita clasificar y almacenar los índices

Toda persona que haya consultado alguna vez un sistema de índice por fichas (como el archivo de una biblioteca, por ejemplo) sabe la gran utilidad que ofrece. Si a usted en alguna ocasión se le ha caído al suelo un fichero, sabrá también que esas mismas fichas, pero desordenadas, se convierten en objetos sin ninguna utilidad. Una biblioteca contiene una vasta cantidad de información, pero a menos que se distribuyan los volúmenes según algún tipo de orden, su valor como sistema de información es prácticamente nulo.

La esencia de un sistema de información no reside en la información en sí misma sino más bien en la forma en que ésta esté organizada. Tomemos, por ejemplo, este dato de una guía comercial:

Pérez, J., Calle F, Barcelona.

En sí mismo, el valor de esta información es limitado; sin embargo, si se sabe que proviene de la sección "Ferreterías" de la guía, adquiere un nuevo significado, determinado por la estructura en la cual se halla incluida.

La estructura más sencilla de información es la que se ofrece en forma de archivo: una serie de datos con ciertas características en común. El nombre del archivo revela algo acerca de la información que contiene, y poner toda la información junta bajo un mismo título hace que su empleo resulte mucho más sencillo. El archivo se puede tratar como una gran unidad de información o como una agrupación particular de unidades más pequeñas. Un libro es un archivo; una novela, por ejemplo, se lee normalmente como una totalidad, mientras que un libro de cocina por lo general se lee como un conjunto de recetas individuales.

Si el archivo es grande, encontrar una información determinada puede suponer tener que comenzar por el primer dato del archivo e ir explorando cada dato hasta hallar el deseado. Esto se conoce como *búsqueda* o *acceso secuencial*, y el archivo organizado de este modo es un *archivo secuencial*. Un programa de televisión es un archivo secuencial de información y lo mismo se puede decir de una conversación entre dos personas.

Los archivos secuenciales son muy comunes porque su implementación es útil y económica y además, en muchos sentidos, reflejan los métodos del pensamiento humano. No obstante, su consulta puede resultar lenta y pesada, de modo que con frecuencia se dividen internamente en subarchivos que se pueden hallar de manera directa sin tener que buscar en todo el archivo. En otro tiempo, los libros eran simples archivos secuenciales, pero sufrieron una transformación a raíz del invento de los capítulos, los números de página y el índice. Los capítulos son los subarchivos del libro, y las páginas son los subarchivos tanto de los capítulos como del libro.

Se denomina *archivo de acceso directo* a aquel que no requiere búsqueda secuencial. Un álbum de canciones en cinta magnética es un archivo secuencial, mientras que el mismo álbum pero en un disco de larga duración es un archivo de acceso directo: hallar una canción en la cinta requiere comenzar por el principio y bobinar hacia adelante, mientras que en el caso del disco se puede acceder directamente a cada surco moviendo el brazo del tocadiscos sobre el LP y colocándolo al comienzo del surco deseado.

El acceso directo depende de que uno sepa dónde se encuentra lo buscado: en un libro el índice puede señalar en qué página comienza un capítulo o se halla determinada ilustración. Saber con exactitud dónde están las cosas significa trabajo (y, por lo tanto, cuesta dinero). Confeccionar el índice de un libro representa un trabajo extra para el autor o el editor, y cabe en lo posible que la información que proporciona el libro no justifique ese gasto.

Los ordenadores procesan grandes cantidades de información a gran velocidad, utilizando una variedad de estructuras de datos. Éstos se han de almacenar de forma permanente en cinta magnética o en disco de algún modo estructurado (generalmente, en archivo secuencial), pero de manera muy diferente en la memoria principal del ordenador.

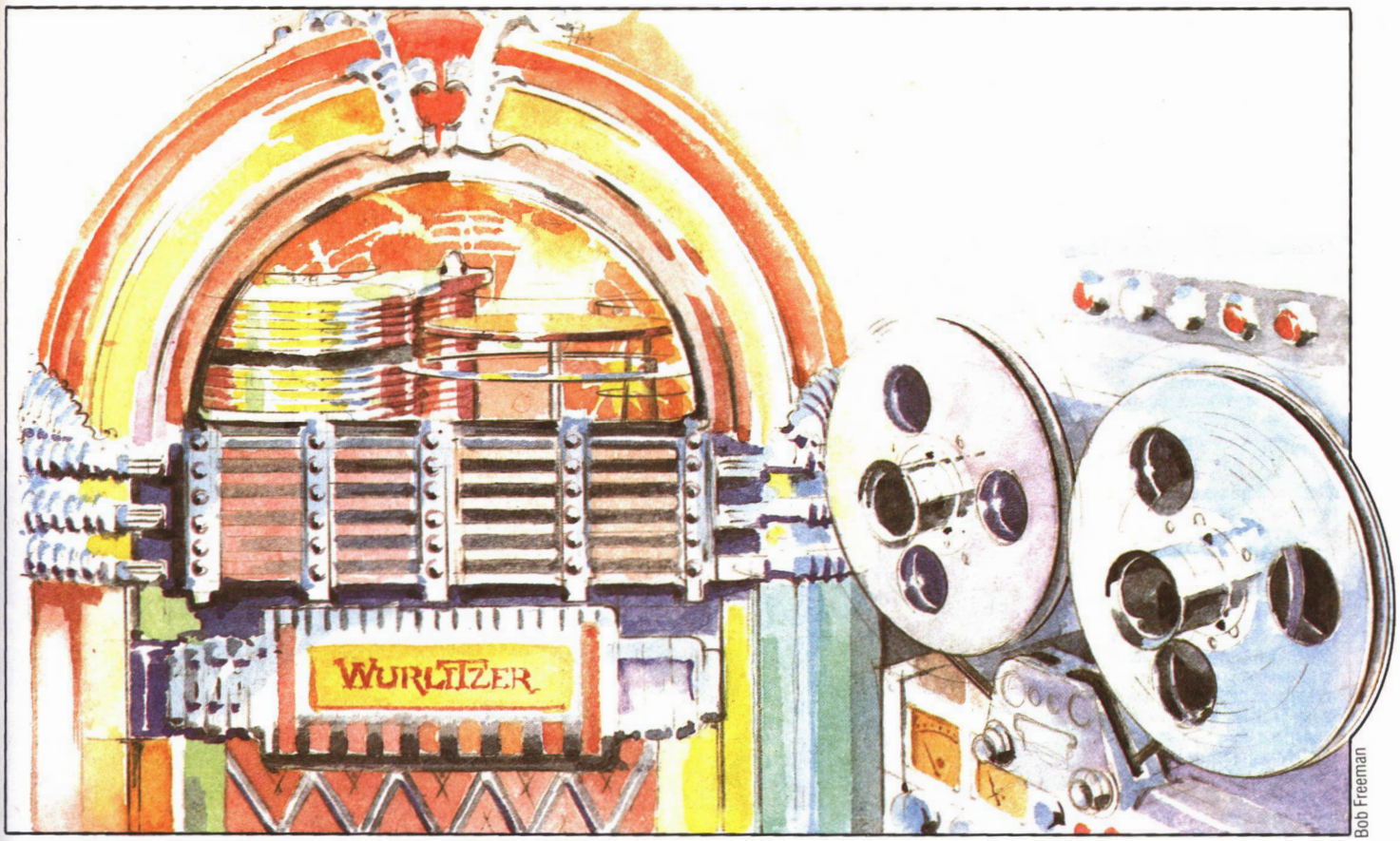
Supongamos que una distribuidora tiene las direcciones de todas las tiendas a las que vende sus productos en un archivo secuencial en cinta y que desea que el ordenador imprima las listas de entrega para que dispongan de ellas los conductores que llevan a cabo el reparto a las tiendas. El archivo en cinta podría ser entonces de la siguiente manera:

Casa Bermúdez	Calle C
González Hnos.	Calle D
Modas Rowena	Calle A
Montes de Gredos	Calle F
Roig y Pujol, S.A.	Calle B
Viuda de Zárate	Calle E

Cuando se lea el archivo de la cinta a la memoria principal, cada nombre y dirección se almacenará en una dirección numerada individual, y todas estas direcciones juntas constituirán un bloque de memoria con un nombre propio, de modo que en la memoria el archivo es así:

NOMBRE DEL BLOQUE:	Tiendas
1) Casa Bermúdez	Calle C
2) González Hnos.	Calle D
3) Modas Rowena	Calle A
4) Montes de Gredos	Calle F
5) Roig y Pujol, S.A.	Calle B
6) Viuda de Zárate	Calle E

Ahora se puede acceder individualmente a cada uno de los datos refiriéndose al bloque y a la dirección correspondientes dentro del mismo. Tiendas (4), por ejemplo, contiene Montes de Gredos, Calle F. Esta estructura se denomina matriz y es la estructura de datos utilizada más comúnmente por los ordenadores para el procesamiento interno de datos. Es como un libro



Bob Freeman

En el surco correcto
Un juke-box contiene 200 canciones grabadas en 100 discos y pesa 80 kg. Para seleccionar una canción hay que pulsar tres teclas. El tiempo promedio entre SELECT y PLAY es de 15 segundos. En una grabadora de cinta, un carrete de cinta magnética puede contener las mismas 200 canciones y pesa sólo 10 kg. Para seleccionar cualquier canción hay que rebobinar la cinta, pulsar PLAY y esperar. Entre SELECT y PLAY median alrededor de 1 500 segundos.
Un juke-box es una máquina de acceso directo: es rápida, bastante especializada y cara. Por el contrario, una grabadora de cinta es un aparato de acceso secuencial: lento, mucho menos especializado que el juke-box, pero de precio más asequible. Una grabadora de cassette conectada a un microordenador es un dispositivo de acceso secuencial, mientras que una unidad de disco flexible es un dispositivo de acceso directo, aun cuando se pueda utilizar para almacenar archivos secuenciales

con una información determinada en cada página. Observe que esta sencilla estructura modifica de inmediato la forma en que observamos los datos. Un archivo de nombres y direcciones reconocibles se ha convertido en un bloque de datos anónimos. Los ordenadores no necesitan saber qué significa un dato determinado, sino sólo dónde está y qué deben hacer con él.

Los datos de la matriz Tiendas ( ) están clasificados alfabéticamente, pero es poco probable que este orden sea el más adecuado para visitar las tiendas. Supongamos que el ordenador determina que el mejor orden de entrega es:

1) Montes de Gredos	Calle F
2) Casa Bermúdez	Calle C
3) Modas Rowena	Calle A
4) González Hnos.	Calle D
5) Viuda de Zárate	Calle E
6) Roig y Pujol, S.A.	Calle B

Este plan de entrega se puede almacenar en otra matriz, pero ello implicaría almacenar dos veces en la memoria la misma información. Los propietarios de micros sabrán que la RAM es limitada, y puede ser inadecuado o bien imposible duplicar los datos de esa manera; de modo, entonces, que se necesita otro método.

Si los datos verdaderos se reemplazan por los números que ostentaban en la matriz Tiendas ( ), el plan de entrega sería el siguiente:

NOMBRE DEL BLOQUE: Entregas
1) 4
2) 1
3) 3
4) 2
5) 6
6) 5

Lo anterior significa en realidad: "Vaya primero a la tienda cuyas señas están almacenadas en Tiendas (4), vaya luego a Tiendas (1), después a Tiendas (3)...", y así sucesivamente. La única información significativa del plan es el orden en que los conductores que efectúan el reparto de la mercancía deben visitar las tiendas, de modo que eso es lo único que se ha de almacenar en la nueva matriz, Entregas.

Entregas ( ) es ahora un índice de la matriz Tiendas ( ) con el fin de determinar las entregas. Al imprimir este plan, el ordenador utilizará los números de la matriz Entregas ( ) para imprimir los nombres y las direcciones de la matriz Tiendas ( ) en el orden correcto.

En este sencillo ejercicio, la información (los nombres y direcciones de las tiendas) se ha manipulado pero no se ha modificado en razón de las distintas estructuras de datos que se le han impuesto. Una estructura de datos no modifica el contenido de éstos, sino que les otorga un significado al asociarlos de forma ordenada con otros datos.

Así como podemos reordenar la matriz Tiendas ( ), clasificándola esta vez de acuerdo a la matriz Entregas ( ), de la misma manera podemos confeccionar otros índices con otras finalidades. Cuando hablábamos de las bases de datos, comentamos que cierta información se podía seleccionar tomando como referencia indicadores incluidos en cada registro individual. De este modo podemos "colocar" en cada registro del archivo Tiendas ( ) un indicador que señale su lugar en el plan de entregas. Podemos ampliar el registro aún más para incluir, por ejemplo, un indicador de un archivo de pedidos fijos; Casa Bermúdez, pongamos por caso, siempre encarga 50 camisas de manga larga, 15 de manga corta, etc. El departamento de producción podrá entonces extraer del archivo la información relativa sólo al número de camisas que necesita confeccionar.

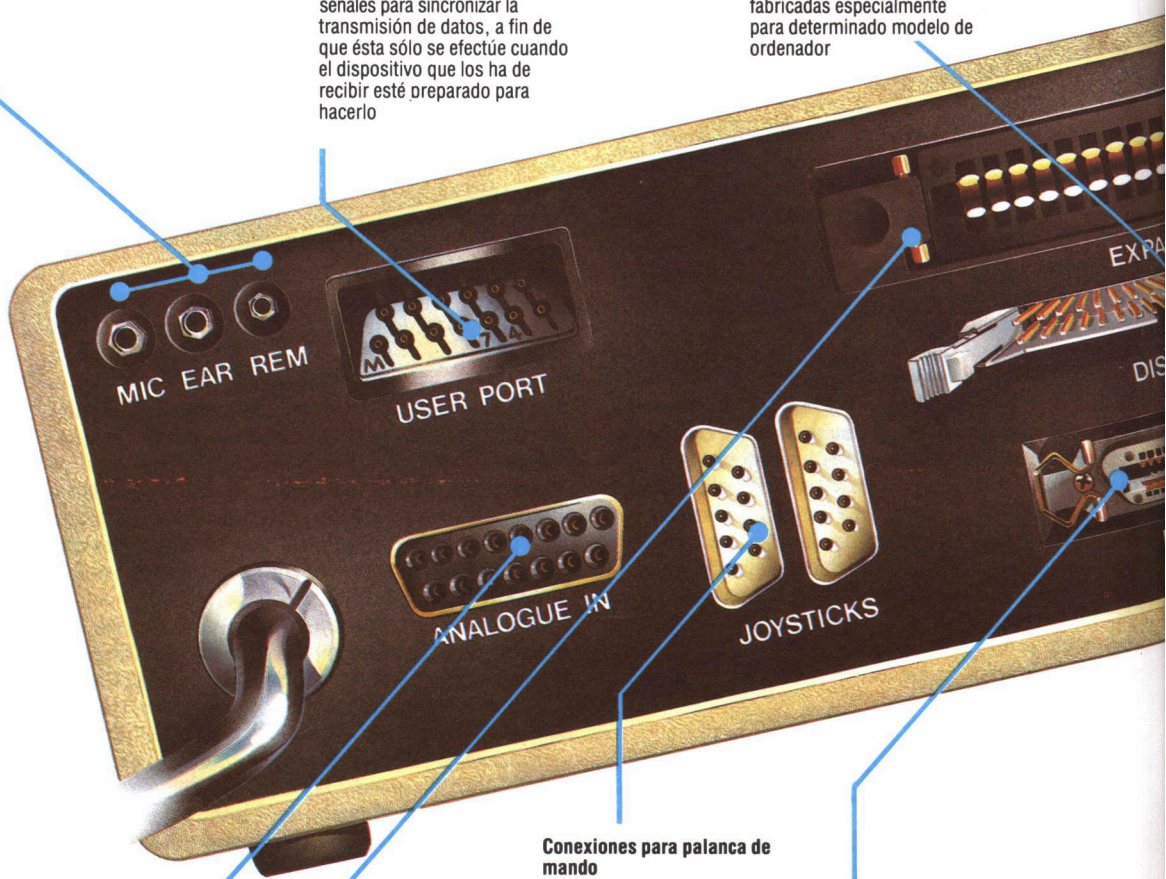
Conexiones útiles

No se puede enchufar una clavija cuadrada en un agujero redondo; tampoco es posible conectar entre sí dos dispositivos para ordenador si no tienen interfaces compatibles

Interface para cassette
La conexión para cassette o interface de que disponen la mayoría de los ordenadores personales es en realidad un tipo de interface en serie. Debido a que los datos se han de grabar en cintas de cassette normales, utilizando frecuencias de audio, no se pueden conseguir altas velocidades de transferencia de datos. El sistema de circuitos de la interface toma los bytes de datos para grabar de la memoria y convierte cada uno de ellos en un flujo de bits. Al cargar las cintas en la memoria, el sistema de circuitos de la interface decodifica los tonos y los unos y ceros resultantes se ensamblan en bytes de ocho bits para su almacenamiento en la memoria. Las interfaces para cassette de la mayoría de los micros personales son universales, en el sentido de que se puede utilizar cualquier grabadora de cinta normal obteniendo los mejores resultados. El conector empleado no está estandarizado, pero los de uso más común son los conectores DIN o minienchufes.

Conexión en paralelo
Ésta es una interface en paralelo con fines generales para conectar a un micro con dispositivos periféricos. Los ocho bits de cada byte transmitido se envían juntos (en paralelo) a través de ocho cables. Se proporcionan otras señales para sincronizar la transmisión de datos, a fin de que ésta sólo se efectúe cuando el dispositivo que los ha de recibir esté preparado para hacerlo

Interface para unidad de disco
Normalmente las unidades de disco se conectan a los ordenadores utilizando una interface en paralelo. No existe ninguna estandarización y, por regla general, sólo se pueden conectar las unidades de disco fabricadas especialmente para determinado modelo de ordenador



Entrada analógica
Existente por lo general sólo en los ordenadores más caros diseñados con fines educativos, una entrada analógica es útil para conectar el ordenador con dispositivos de laboratorio como sensores de luz o indicadores eléctricos de temperatura. La interface constará sólo de una o más líneas capaces de aceptar y leer un voltaje dentro de una escala determinada. El usuario debe asegurarse de que el ordenador no se conecta a un voltaje que supere dicha escala, lo cual entrañaría peligro

Conexión para ampliación de memoria
Por lo general sustenta la mayor parte de las líneas, si no todas, que vienen directamente desde el microprocesador, es decir, los buses de direcciones, de datos y de control. Es aquí donde se debe enchufar la memoria adicional y, en algunos ordenadores, también los periféricos del fabricante. Puede acoplarse mediante un conector terminal PCB, aunque en algunos casos puede tratarse de un enchufe capaz de acoger un conector terminal, como el de un cartucho para juegos (que es en realidad una forma de ampliación de la ROM)

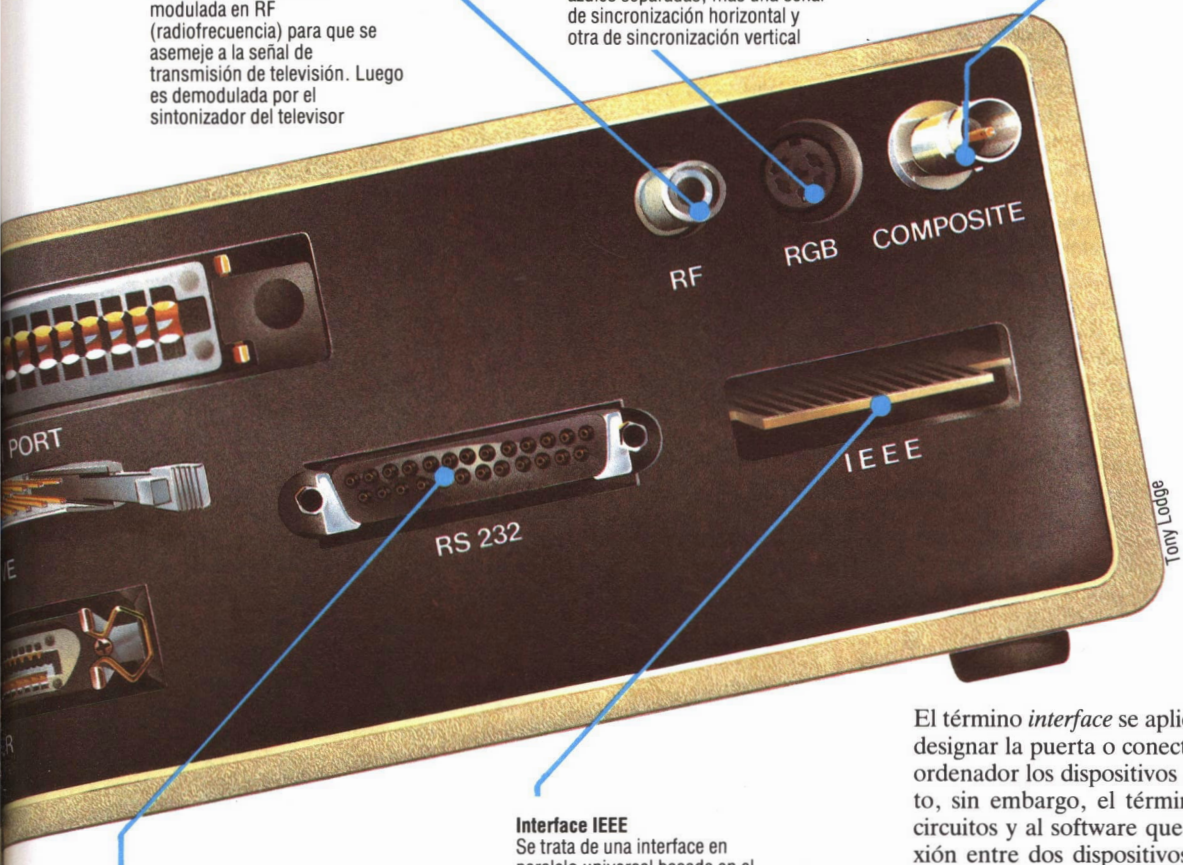
Conexiones para palanca de mando
No hay ninguna interface estándar para las palancas de mando, aunque muchos fabricantes han adoptado el modelo de Atari. La mayoría de dichas interfaces poseen simplemente cinco líneas activas (una desde cada interruptor de las cuatro modalidades del movimiento de la palanca de mando, y la quinta para el botón de disparo). Sin embargo, las palancas analógicas requieren una interface diferente, capaz de aceptar una amplia escala de voltajes para indicar la posición exacta de la palanca.
La mayoría de los ordenadores disponen de más de una conexión para palanca de mando, si bien en ocasiones varios dispositivos comparten el mismo enchufe

Interface para impresora
Las interfaces para impresoras están relativamente estandarizadas según un sistema desarrollado por la Centronics Corporation, de modo que no es difícil conseguir una impresora con interface Centronics que funcione con la interface para impresora de la mayoría de los ordenadores. Los niveles de señales, así como las funciones de señales, están asimismo estandarizados en 0 y 5 voltios para el 0 y el 1 binarios respectivamente. Los conectores utilizados y la asignación de las señales a las diversas patillas no están estandarizados; por este motivo el usuario tal vez precise de un cable especial para conectarle una impresora al ordenador

Interface para video (RF)
Todos los ordenadores personales están construidos para ser conectados a una unidad de visualización en video, que en la mayoría de los casos es un televisor corriente, en color o en blanco y negro. Si se ha de utilizar el enchufe normal del televisor para la entrada de antena, la señal de video primero ha de ser modulada en RF (radiofrecuencia) para que se asemeje a la señal de transmisión de televisión. Luego es demodulada por el sintonizador del televisor

Interface RGB
Se pueden obtener aún mejores resultados si se mantienen separados los elementos que requiere el monitor de video. La interface RGB proporciona señales de video rojas, verdes y azules separadas, más una señal de sincronización horizontal y otra de sincronización vertical

Interface para video (composite)
Algunos televisores y muchos monitores de video incorporan una entrada de video compuesta que evita la etapa de demodulación de RF, con lo cual se consigue producir imágenes de mayor calidad. En la salida del ordenador están presentes todos los elementos de una señal de video estándar (señales de crominancia, luminancia, sincronización, etc.), pero esta señal "compuesta" no requiere una posterior modulación o procesamiento por parte del sintonizador del televisor



Interface en serie
A diferencia de muchas otras, la interface en serie RS232 está, a nivel teórico, definida con toda precisión según la normativa de la Asociación de Industrias Eléctricas. Ésta especifica el tipo de conector a utilizar (un conector "miniatura D" de 25 patillas), las señales destinadas a cada patilla y los niveles de señales. Lamentablemente, son pocos los fabricantes que se atienen a la normativa y con determinados ordenadores puede resultar difícil hacer funcionar dispositivos en serie. De las muchas patillas que presenta la interface estándar, normalmente sólo se utilizan tres: la 2, para transmitir los datos en serie desde el ordenador al periférico; la 3, para recibir los datos transmitidos desde éste, y la 7, la señal a tierra. Tanto los dispositivos de transmisión como los de recepción se han de ajustar para que la velocidad de transmisión de los datos y el formato de los datos transmitidos sean los mismos

Interface IEEE
Se trata de una interface en paralelo universal basada en el bus de interface Hewlett Packard y adoptada ahora como estándar. La normativa está muy bien definida, tanto física como eléctricamente. A diferencia de otras interfaces para datos (como, por ejemplo, el Centronics en paralelo y el RS232 en serie), que sólo se pueden conectar a un dispositivo a la vez, el bus IEEE se puede acoplar simultáneamente con hasta 15 aparatos (incluyendo el propio ordenador). Los dispositivos que incorporan interfaces IEEE incluyen impresoras, unidades de disco flexible, plotters, generadores de señales, voltímetros y otros equipos de comprobación. Como se presta muy bien para emplearlo con equipos de medición y pruebas, el bus IEEE se utiliza mucho en los laboratorios y establecimientos industriales. En la actualidad son muy pocos los ordenadores personales que ofrecen interfaces IEEE y algunos de ellos utilizan una conexión en el tablero de circuito impreso en lugar del conector IEEE estándar

El término *interface* se aplica con cierta vaguedad para designar la puerta o conector en que se le enchufan al ordenador los dispositivos externos. En sentido estricto, sin embargo, el término se refiere al sistema de circuitos y al software que permiten efectuar la conexión entre dos dispositivos cualesquiera relacionados con el ordenador.

En su interior, el ordenador se comunica enviando datos a través de "buses", grupos de conductores en paralelo, cada uno de los cuales conduce una única señal binaria. En la mayor parte de los microordenadores existen tres buses internos: un bus de datos de 8 bits, un bus de direcciones de 16 bits y un bus de control, cuyas señales, por lo común constan de entre 5 y 12 bits que indican el estado normal de la CPU. Algunas de las señales de control le avisan a la memoria y a los dispositivos periféricos si la CPU desea recibir datos (leer) o depositar datos (escribir). Otras transmiten información desde el exterior a la CPU, informándole, por ejemplo, de que un dispositivo periférico tiene que dar entrada a determinados datos y requiere atención.

Internamente, el ordenador por lo general manipula la información compuesta ya sea de 8 bits o de 16 bits a la vez. Por tanto, si la CPU desea recuperar los datos de la dirección de memoria 65535 (o FFFF, expresada en hexadecimal), establecerá todos los 16 cables del bus de direcciones en uno para identificar esa localización. Si el contenido de esta dirección de memoria resultara ser 182 (B6, en hexadecimal), este dato se colocaría en el bus de datos como los ocho dígitos binarios 10110110.

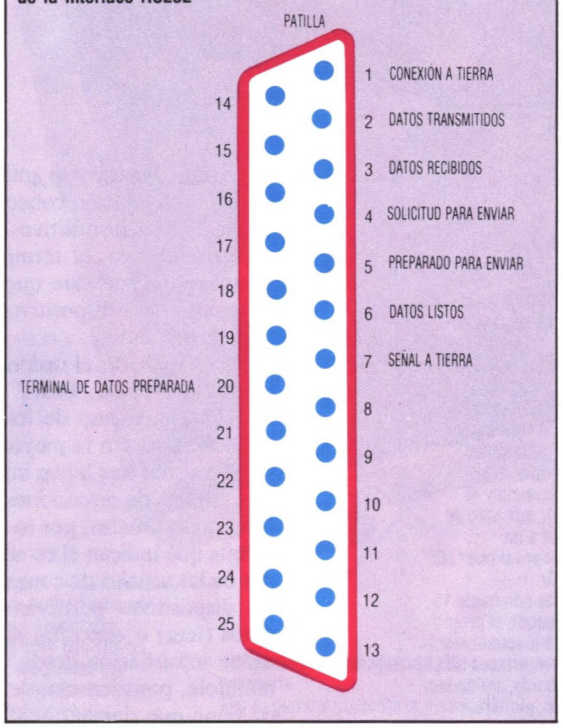
Cuando se transfieren datos de esta manera, de a 8 o 18 bits a la vez, se dice que la transferencia es "en

paralelo". Muchos dispositivos periféricos también están diseñados para transmitir o recibir datos en paralelo. Los interfaces que se facilitan para los periféricos de este tipo se denominan *interfaces en paralelo* y la mayoría de los ordenadores proporcionan al menos un conector específicamente para dispositivos "en paralelo".

No todos los dispositivos periféricos pueden recibir o transmitir datos en paralelo. Algunos se valen de un solo cable para comunicarle al ordenador un bit a la vez. Internamente siguen utilizando los datos en forma de bytes de 8 o 16 bits, pero los bits de cada byte se transmiten o se reciben de a uno cada vez, empezando por el bit "menos significativo" del byte y terminando con el bit "más significativo". Cada byte se divide en un flujo de bits, enviados uno después del otro, y vueltos a ensamblar en un byte al otro extremo, utilizando unos circuitos especiales convertidores de paralelo a en serie y viceversa.

Se puede hacer que tanto las interfaces en serie como las interfaces en paralelo transmitan información ya sea desde el ordenador o hacia el mismo. Los ordenadores suelen incluir otras interfaces, que o bien siempre envían la información hacia afuera (por ejemplo, el circuito de salida del televisor) o siempre aceptan la información entrante (por ejemplo, las conexiones para palancas de mando).

Funciones estandarizadas para las patillas de la interface RS232



Interfaces en serie

Existe una interface en serie estandarizada, la RS232, para la cual están definidos con todo detalle los niveles de señales y la asignación de patillas. Hasta el tipo de conector está especificado. Lamentablemente, son pocos los fabricantes que se adhieren por completo a la normativa y hacer funcionar las conexiones en serie puede resultar difícil. La RS232 es una interface en serie "bidireccional", con una patilla (terminal) para transmitir datos desde el ordenador y otra destinada a recibirlos.

Los datos se envían de a un bit cada vez por vía del terminal de "datos transmitidos" y se reciben a través del terminal de "datos recibidos". El flujo de bits puede asumir diversos formatos estandarizados, pero no importa cuál de ellos se utilice siempre y cuando ambos dispositivos, el de recepción y el de transmisión, empleen el mismo.

Dado que cada byte de información se envía hacia afuera como un flujo de bits en serie, el software que controla a la interface ha de tener algún modo de indicar cuándo comienza el primer bit del dato y cuándo ha terminado el último bit. El procedimiento utilizado más comúnmente tiene un único "bit de inicio" (0, en lógica de Boole), seguido de ocho bits de datos, seguidos de un único "bit de parada" (un 1 lógico).

Es necesario especificar de antemano la velocidad a la cual se transmiten los datos, ya que, de lo contrario, se podría malinterpretar el patrón de impulsos que representa en forma de ceros y unos a los ocho bits de datos. Esta velocidad de transmisión de datos se denomina *velocidad baudio*, en honor del francés Baudot, que la inventara en el siglo XIX. Las velocidades baudio oscilan entre 75 y 9 600 baudios. Estas cifras corresponden a los 75 y 9 600 bits por segundo que se transmiten, y como normalmente hay 10 bits (incluyendo los bits de inicio y de parada) para cada carácter, la velocidad de transmisión de caracteres equivale a la décima parte de la velocidad baudio.

Interfaces en paralelo

La interface en paralelo transmite o recibe la información de a un byte cada vez, pero además de las ocho "líneas de datos" necesita también proporcionar otras señales para que el ordenador y el periférico sepan cuándo es posible transmitir datos y cuándo no. El tipo más común de interface en paralelo es la "Centronics" (llamada así en razón de la fábrica norteamericana de impresoras Centronics Corporation); pero este supuesto modelo estándar no está muy afianzado. El tipo de conector que se usa y la asignación de señales a las diversas patillas presentan muchas diferencias de un fabricante a otro. La mayoría de interfaces Centronics proporcionan al menos las señales siguientes:

De DATO 0 a DATO 7	Ocho cables para transportar los ocho bits del byte que se está transmitiendo
ADK	Una señal de entrada para el ordenador que indica que el dispositivo receptor está preparado para aceptar datos
GND	El cable "a tierra" que proporciona una referencia común de 0 voltios tanto al ordenador como al dispositivo periférico
BUSY	Una señal desde el dispositivo periférico hasta el ordenador que indica que el periférico no puede aceptar información
STROBE	Una señal de salida desde el ordenador que le indica al periférico que la información está preparada y debe leerla

Muchos otros dispositivos, aparte de las impresoras, adoptan la interface en paralelo casi estandarizada de Centronics, y es probable que para conectarlas al ordenador sólo haya que comprar un cable especial para conexión o acondicionar uno el mismo usuario.

Multiplicación

Los ordenadores dan respuestas muy rápidas a complejos problemas aritméticos, abordándolos del modo más sencillo

En la última parte de nuestro curso acerca del sistema binario, descubrimos cómo suman los ordenadores. A continuación estudiaremos el proceso de la multiplicación.

Si tuviera que multiplicar 14 por 12, una forma sencilla de hacerlo sería mediante una suma múltiple, por ejemplo 14+14+14+14+... (12 veces). Dado que, en cierto sentido, la multiplicación es una suma repetida, con toda seguridad este método funcionará. (Ésta era, precisamente, la forma en que multiplicaban los primeros ordenadores.)

Sin embargo, este procedimiento es tosco y ocupa mucho tiempo, de modo que los diseñadores de estas maravillosas máquinas tuvieron que desarrollar un método más eficaz.

Cuando se multiplican dos números, normalmente la operación se escribe así:

```
  14
 ×12
 ---
  28
+14      (a menudo se suele poner
 ---      un 0 final para ayudar a colocar
 168      los dígitos en la columna correcta)
```

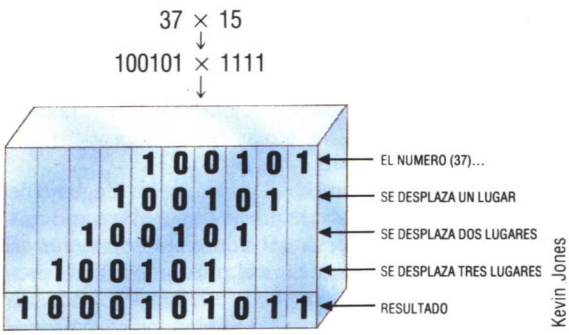
El proceso funciona exactamente de la misma manera con cualquier base de números. Tomemos un ejemplo binario o de base dos:

```
  101
 ×11
 ---
  101
+101
 ----
 1111
```

Aun con números más elevados el procedimiento es exactamente el mismo que hemos expuesto, de modo que volvamos al ejemplo de 14×12 y resolvámoslo en binario:

```
   1110      (14)
  ×1100      (12)
  -----
   0000
  0000
 1110
1110
--------
10101000    (168)
```

En el sistema binario la multiplicación es todavía más sencilla que en el decimal, porque nunca puede llevarse un dígito. Cuando se multiplica un número por 1, dicho número no se modifica: $14 \times 1 = 14$; y cuando se multiplica un número por 0, el resultado es sencillamente 0: $14 \times 0 = 0$. Y esto ocurre así tanto en binario como en decimal y en cualquier otro de los sistemas numéricos.

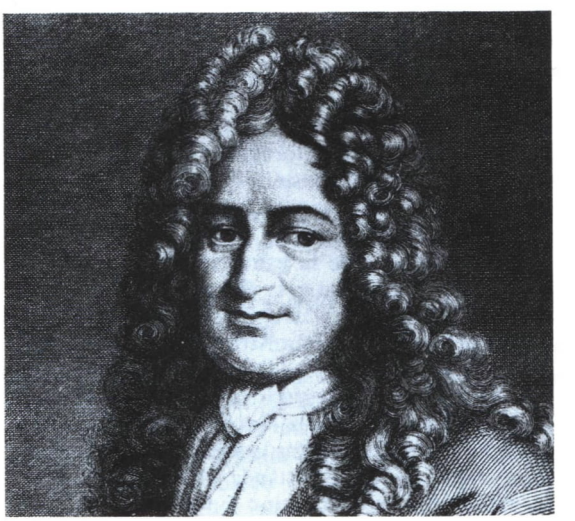


El desplazamiento en la multiplicación
Multiplicar en binario es mucho más fácil que multiplicar en decimal. Se sigue el mismo proceso de multiplicación larga que se aplica en el sistema decimal; pero como sólo hay dos números implicados en la multiplicación (0 y 1), la operación es muy sencilla. Cuando un número se multiplica por 1 el resultado es, obviamente, el mismo número. En la ilustración, donde se multiplica 100101 (37) por 1111 (15), aparecen cuatro copias de 100101. Cada copia se desplaza hacia la izquierda de acuerdo a la posición del 1 por el cual se multiplique. Finalmente, se suman todas las copias y se obtiene el resultado de 1000101011 (555)

Cuando los matemáticos estudiaron cálculos similares al que mostramos arriba, vieron que se producía un patrón sencillo: la multiplicación en binario consiste en sólo dos operaciones: desplazamiento y suma. Y así es exactamente como multiplica el ordenador. Primero desplaza "copias" de la línea superior a su posición correcta (determinada por los unos y los ceros de la línea inferior) y luego suma todas las "copias" juntas.

Para poder multiplicar, el ordenador necesita una gran capacidad de dígitos. Cuando el número de 4 dígitos 1110 multiplicó por el número de 4 dígitos 1100, el resultado fue de 8 dígitos (10101000) y, en general, el resultado de una multiplicación puede tener una longitud que llegue a equivaler hasta el doble de la del número mayor.

Puede constituir una sorpresa enterarse de que los ordenadores pueden equivocarse al multiplicar. Prácticamente todos estos errores se producen en función de la cantidad de espacio que el diseñador de la máquina haya previsto para retener el resultado. Si no existe espacio suficiente se producirá un "desbordamiento", se perderán los dígitos menos significativos y, en consecuencia, el resultado será erróneo.



Gottfried Wilhelm Leibniz
Leibniz (1646-1716) fue contemporáneo de Isaac Newton e hizo aportaciones en el campo de las matemáticas, la ciencia y la filosofía. Inventó una máquina que multiplicaba y dividía. También investigó las posibilidades de utilizar la aritmética binaria en aparatos de cálculo, si bien el primer antecedente conocido en cuanto a números binarios se atribuye a Francis Bacon, y data de 1623. En los últimos años de vida entabló una polémica con Newton acerca de la invención del cálculo

Estructuras de control

Todas las versiones de BASIC incorporan estructuras que gobiernan el flujo de un programa. No obstante, algunas máquinas ofrecen una amplia gama de alternativas, con sutiles diferencias

En los diez primeros capítulos del curso de programación BASIC hemos cubierto casi todos los aspectos más importantes del lenguaje BASIC. En esta ocasión le ofreceremos una recopilación de los temas que hemos desarrollado hasta ahora, haremos algunas interesantes digresiones y proporcionaremos al lector de la obra una idea general de los temas que nos proponemos desarrollar en adelante.

Dediquémonos primero al resumen: un lenguaje de alto nivel como el BASIC le proporciona al usuario una serie de instrucciones que se traducen internamente en una forma comprensible para el ordenador. Todo programa para ordenador se puede escribir utilizando tan sólo dos sencillos patrones denominados *construcciones*. Se trata de las "construcciones de secuencia" y de las "estructuras de control", de las cuales sólo dos son esenciales en BASIC: IF...THEN...ELSE y WHILE...DO. La mayor parte de los otros lenguajes para ordenador proporcionan una cantidad de instrucciones considerablemente superior.

La construcción de secuencia permite que la tarea se divida en una serie de subtareas que, al ser ejecutadas en secuencia, efectúan la tarea principal. Las dimensiones de las subtareas dependen del lenguaje; en BASIC las subtareas se representan mediante las sentencias escritas en cada línea, y la secuencia se representa, a su vez, por medio de los números de línea. Por lo tanto, si la tarea consistiera en multiplicar por 10 el valor asignado a una variable, la secuencia que utilizaríamos podría ser la siguiente:

```
110 INPUT N
120 LET N = N * 10
130 PRINT N
```

Además de construcciones de secuencia necesitamos también estructuras de control. Se trata de construcciones que alteran el orden de ejecución de las sentencias de un programa.

La estructura de control más sencilla que proporciona el BASIC es GOTO. Se trata de un salto (o bifurcación) incondicional que redirige la ejecución del programa hasta un número de línea determinado sin que se haya de satisfacer ninguna prueba o condición. GOSUB es también una bifurcación incondicional, pero el programa siempre retornará (RETURN) hasta el punto inmediatamente posterior a la GOSUB y su utilización en la programación estructurada es perfectamente aceptable.

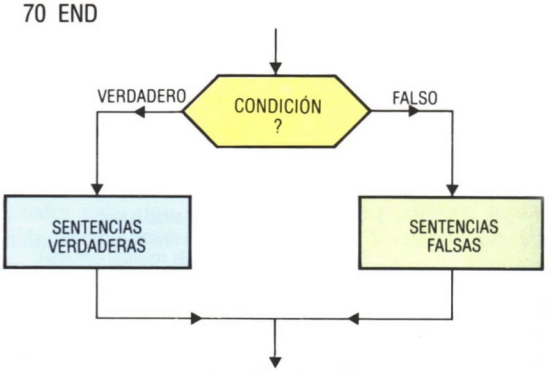
El BASIC dispone de la estructura de control IF... THEN...ELSE. Ésta asume la forma de la sentencia IF...THEN y posee la siguiente sintaxis (téngase presente que, en el lenguaje informático, la palabra "sintaxis" equivale a "forma"):

Si (IF) (la condición especificada) es verdadera *entonces* (THEN) ejecute la sentencia especificada (*si no*) (ELSE), debe ejecutar la sentencia siguiente.

Observe que en el BASIC estándar, la palabra ELSE de IF...THEN...ELSE está implícita. En algunas versiones de BASIC y en algunos otros lenguajes, como el PASCAL, por ejemplo, ELSE forma parte de la sentencia.

IF...THEN...ELSE (IF...THEN en BASIC) realiza una de dos subtareas según si una determinada condición es verdadera o no. Consideremos el siguiente programa, cuyo propósito es hallar la raíz cuadrada de números que se digitan por el teclado. La entrada del número −9999, por ejemplo, nos indicará que queremos terminar el programa:

```
10 PRINT "DE ENTRADA A UN NUMERO"
20 INPUT N
30 IF N = −9999 THEN GOTO 70
40 LET S = SQR(N)
50 PRINT "LA RAIZ CUADRADA DE";N;"ES";S
60 GOTO 10
70 END
```



La estructura de control IF...THEN...ELSE
Si la condición es verdadera, se ejecutarán las sentencias verdaderas. Si la condición es falsa, se ejecutarán las sentencias falsas

Aquí lo que la línea 30 dice realmente es "*Si* (IF) es verdad que N = −9999, *entonces* (THEN) vaya hasta el final del programa, *si no* (ELSE) (si no es verdad que N = −9999) ejecutar la siguiente línea del programa para hallar la raíz cuadrada".

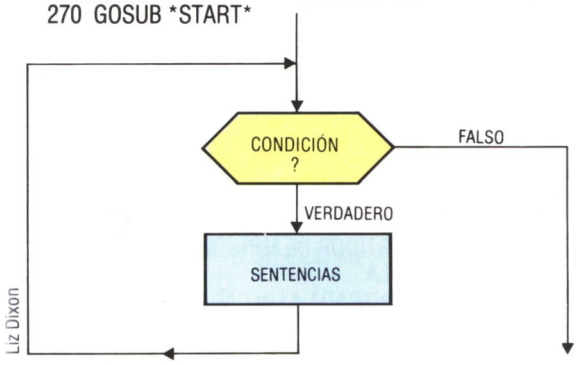
El BASIC no dispone directamente de la otra estructura de control esencial (WHILE...DO), pero se puede conseguir fácilmente. WHILE...DO es una forma de "hacer un bucle" y significa "repita una sentencia o serie de sentencias *mientras* (WHILE) una condición sea verdadera, *haga* (DO) algo".

WHILE...DO siempre comprueba la condición antes de que se ejecuten las sentencias, de manera que si la comprobación fracasa la primera vez que se realiza, las sentencias (denominadas el cuerpo del bucle) no se ejecutan. Como ejemplo, consideremos un programa para juegos que le indica al jugador que 'PULSE LA BARRA ESPACIADORA CUANDO ESTE LISTO'. Esta parte se podría escribir asimismo (usando un seudolenguaje o castellano simplificado) de la manera siguiente:

Mientras (WHILE) no se pulse la barra espaciadora
explore (DO) el teclado
comienza juego

En BASIC esto se escribiría:

```
250 PRINT "PULSE LA BARRA ESPACIADORA
    CUANDO ESTE LISTO"
260 IF INKEY$ <> " " THEN GOTO 260
270 GOSUB *START*
```



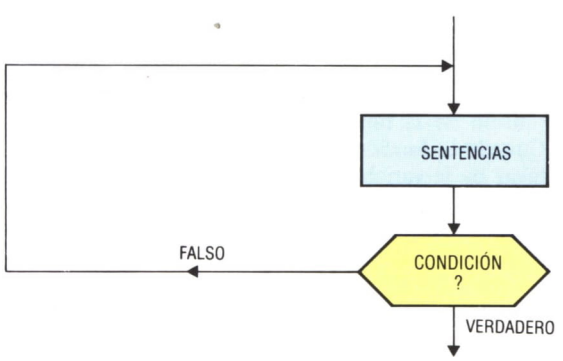
La estructura de control DO...WHILE
El bucle se repite en la medida en que la condición es verdadera. Es posible que las sentencias no lleguen a ejecutarse nunca (si la condición inicial es falsa)

La línea 260 dice que *si* (IF) INKEY$ no es igual a (<>) un espacio (" "), *entonces* (THEN), retroceda y verifique nuevamente el teclado. Una forma algo más elegante de escribir esto sería la siguiente:

```
250 PRINT "PULSE LA BARRA ESPACIADORA
    CUANDO ESTE LISTO"
260 FOR X = 0 TO 1 STEP 0
270 IF INKEY$ = " " THEN LET X = 2
280 NEXT X
290 GOSUB * STAR *
```

En este fragmento del programa el bucle (para explorar el teclado) sólo se ejecuta si no se ha pulsado la barra espaciadora. Si se ha pulsado (es decir, INKEY$ = " "), entonces el programa abandona el bucle FOR...NEXT y va a la línea 290, que llama a la subrutina START. (Obsérvese que estamos utilizando "etiquetas" o nombres para las subrutinas. Muchas versiones de BASIC no pueden llamar a las subrutinas por un nombre; en tal caso, el usuario tendrá que emplear, en lugar de las etiquetas, números de línea.)

Hasta ahora nunca nos habíamos encontrado con STEP y quizá ésta sea una forma algo inusual de aplicarla. Cuando se utiliza un bucle FOR...NEXT, STEP permite que el "índice" se incremente en unidades distintas a uno. FOR I = 1 TO 10 STEP 2 hará que I tenga el valor 1 la primera vez que se efectúe el bucle, y posteriormente 3, 5, 7 y 9. El incremento siguiente (a 11) superaría el límite de 10, de modo que el bucle se da por concluido. Incluso es posible hacer que el índice cuente hacia atrás. FOR I = 10 TO 1 STEP−1 hará que I cuente hacia atrás desde 10 hasta 1. Utilizar STEP 0 es realmente un truco inteligente que asegura que el bucle nunca terminará a menos que X "se incremente artificialmente", como en el caso de nuestra sentencia IF...THEN.



La estructura de control REPEAT...UNTIL
El bucle se repite hasta que la condición resulta verdadera. Las sentencias siempre se ejecutarán al menos una vez

Otra estructura de control muy útil que tampoco está disponible en BASIC directamente pero que se puede imitar con facilidad, es REPEAT...UNTIL. En este caso la prueba de condición viene después del cuerpo principal del bucle, de manera que la sentencia o las sentencias del cuerpo principal siempre se repetirán al menos una vez. Observemos este "generador de números al azar":

```
10 PRINT "PULSE LA BARRA ESPACIADORA"
20 FOR X = 0 TO 1 STEP 0
30 LET R = R + 1
40 IF R > 9 THEN LET R = 1
50 IF INKEY$ = " " THEN LET X = 2
60 NEXT X
70 PRINT "EL VALOR DE R ES  ";R
```

Aquí, el cuerpo principal (que incrementa el valor de R) siempre se ejecuta al menos una vez, dado que la prueba para la bifurcación del bucle (IF INKEY$ = " ") llega sólo después de la sentencia de incremento (LET R = R + 1).

Otra estructura de control muy útil pero, sin embargo, no esencial, es la que generalmente se denomina CASE. En BASIC, la estructura CASE se aplica empleando ON...GOTO o bien ON...GOSUB. Funciona de esta manera. ON...GOTO es una sentencia de bifurcación múltiple que incorpora varias pruebas condicionadas IF...THEN en una sola sentencia. Consideremos, por ejemplo, un fragmento de programa que convierta los números del 1 al 7 en los nombres de los siete días de la semana:

```
1050 IF D = 1 THEN GOTO 2020
1060 IF D = 2 THEN GOTO 2040
1070 IF D = 3 THEN GOTO 2060
1080 IF D = 4 THEN GOTO 2080
1090 IF D = 5 THEN GOTO 3000
2000 IF D = 6 THEN GOTO 3020
2010 IF D = 7 THEN GOTO 3040
2020 PRINT "LUNES"
2030 GOTO*END*
2040 PRINT "MARTES"
2050 GOTO*END*
2060 PRINT "MIERCOLES"
2070 GOTO*END*
2080 PRINT "JUEVES"
2090 GOTO*END*
3000 PRINT "VIERNES"
3010 GOTO*END*
3020 PRINT "SABADO"
3030 GOTO*END*
3040 PRINT "DOMINGO"
3050 GOTO*END*
```

En BASIC, una forma más concisa de conseguir los mismos resultados sería la de utilizar ON...GOTO de la siguiente manera:

```
1050 ON D GOTO 2020, 2040, 2060, 2080,
     3000, 3020, 3040
```

ON...GOSUB funciona igual excepto que el valor de la variable determina hacia qué subrutina se bifurca. A continuación le ofrecemos una ligera modificación del programa de los dados utilizando ON...GOSUB para seleccionar los gráficos adecuados para los dados seleccionados por la función RND:

DECIMAL	BINARIO	CARACTER
32	0 0 1 0 0 0 0 0	= espacio
33	0 0 1 0 0 0 0 1	= !
34	0 0 1 0 0 0 1 0	= "
35	0 0 1 0 0 0 1 1	= #
36	0 0 1 0 0 1 0 0	= $
37	0 0 1 0 0 1 0 1	= %
38	0 0 1 0 0 1 1 0	= &
39	0 0 1 0 0 1 1 1	= '
40	0 0 1 0 1 0 0 0	= (
41	0 0 1 0 1 0 0 1	=)
42	0 0 1 0 1 0 1 0	= *
43	0 0 1 0 1 0 1 1	= +
44	0 0 1 0 1 1 0 0	= ,
45	0 0 1 0 1 1 0 1	= -
46	0 0 1 0 1 1 1 0	= .
47	0 0 1 0 1 1 1 1	= /
48	0 0 1 1 0 0 0 0	= 0
49	0 0 1 1 0 0 0 1	= 1
50	0 0 1 1 0 0 1 0	= 2
51	0 0 1 1 0 0 1 1	= 3
52	0 0 1 1 0 1 0 0	= 4
53	0 0 1 1 0 1 0 1	= 5
54	0 0 1 1 0 1 1 0	= 6
55	0 0 1 1 0 1 1 1	= 7
56	0 0 1 1 1 0 0 0	= 8
57	0 0 1 1 1 0 0 1	= 9
58	0 0 1 1 1 0 1 0	= :
59	0 0 1 1 1 0 1 1	= ;
60	0 0 1 1 1 1 0 0	= <
61	0 0 1 1 1 1 0 1	= =
62	0 0 1 1 1 1 1 0	= >
63	0 0 1 1 1 1 1 1	= ?
64	0 1 0 0 0 0 0 0	= @
65	0 1 0 0 0 0 0 1	= A
66	0 1 0 0 0 0 1 0	= B
67	0 1 0 0 0 0 1 1	= C
68	0 1 0 0 0 1 0 0	= D
69	0 1 0 0 0 1 0 1	= E
70	0 1 0 0 0 1 1 0	= F
71	0 1 0 0 0 1 1 1	= G
72	0 1 0 0 1 0 0 0	= H
73	0 1 0 0 1 0 0 1	= I
74	0 1 0 0 1 0 1 0	= J
75	0 1 0 0 1 0 1 1	= K
76	0 1 0 0 1 1 0 0	= L
77	0 1 0 0 1 1 0 1	= M
78	0 1 0 0 1 1 1 0	= N
79	0 1 0 0 1 1 1 1	= O
80	0 1 0 1 0 0 0 0	= P
81	0 1 0 1 0 0 0 1	= Q
82	0 1 0 1 0 0 1 0	= R
83	0 1 0 1 0 0 1 1	= S
84	0 1 0 1 0 1 0 0	= T
85	0 1 0 1 0 1 0 1	= U
86	0 1 0 1 0 1 1 0	= V
87	0 1 0 1 0 1 1 1	= W
88	0 1 0 1 1 0 0 0	= X
89	0 1 0 1 1 0 0 1	= Y
90	0 1 0 1 1 0 1 0	= Z
91	0 1 0 1 1 0 1 1	= [
92	0 1 0 1 1 1 0 0	= \
93	0 1 0 1 1 1 0 1	=]
94	0 1 0 1 1 1 1 0	= ^
95	0 1 0 1 1 1 1 1	= _
96	0 1 1 0 0 0 0 0	= `
97	0 1 1 0 0 0 0 1	= a
98	0 1 1 0 0 0 1 0	= b
99	0 1 1 0 0 0 1 1	= c
100	0 1 1 0 0 1 0 0	= d
101	0 1 1 0 0 1 0 1	= e
102	0 1 1 0 0 1 1 0	= f
103	0 1 1 0 0 1 1 1	= g
104	0 1 1 0 1 0 0 0	= h
105	0 1 1 0 1 0 0 1	= i
106	0 1 1 0 1 0 1 0	= j
107	0 1 1 0 1 0 1 1	= k
108	0 1 1 0 1 1 0 0	= l
109	0 1 1 0 1 1 0 1	= m
110	0 1 1 0 1 1 1 0	= n
111	0 1 1 0 1 1 1 1	= o
112	0 1 1 1 0 0 0 0	= p
113	0 1 1 1 0 0 0 1	= q
114	0 1 1 1 0 0 1 0	= r
115	0 1 1 1 0 0 1 1	= s
116	0 1 1 1 0 1 0 0	= t
117	0 1 1 1 0 1 0 1	= u
118	0 1 1 1 0 1 1 0	= v
119	0 1 1 1 0 1 1 1	= w
120	0 1 1 1 1 0 0 0	= x
121	0 1 1 1 1 0 0 1	= y
122	0 1 1 1 1 0 1 0	= z
123	0 1 1 1 1 0 1 1	= {
124	0 1 1 1 1 1 0 0	= |
125	0 1 1 1 1 1 0 1	= }
126	0 1 1 1 1 1 1 0	= ~

ASCII
He aquí una lista completa de los valores de ASCII entre 32 y 126, sus equivalentes en binario y los caracteres que representan. El significado atribuido a los valores fuera de esta escala varía considerablemente de una máquina a otra

```
390 REM SUBRUTINA DE SELECCION
400 REM UTILIZANDO ON...GOSUB
410 ON D GOSUB 530, 600, 670, 740, 810, 880
470 RETURN
```

Es posible que su versión de BASIC contenga muchas sentencias y funciones que no hayamos explicado; la mayoría de ellas corresponden a ampliaciones del BASIC "básico", concebidas para sacar más partido a determinadas configuraciones de su máquina. Muchas de ellas estarán relacionadas con las configuraciones para gráficos incorporadas en el hardware: instrucciones como PAINT, PAPER, INK, BEEP y CIRCLE. Éstas tienden a ser "específicas de la máquina" y por ello no las hemos incluido en nuestro curso, si bien en futuros capítulos le proporcionaremos más detalles.

No obstante, antes de dar por terminada la parte de nuestro curso de BASIC, ataremos algunos cabos sueltos: analizaremos el juego de caracteres ASCII, estudiaremos un par de funciones que ayudan a manejar los caracteres y veremos una manera de definir nuevas funciones (o funciones no incluidas en su versión de BASIC).

Con el correr de los años se han ido desarrollando diversos métodos para representar las letras del alfabeto y otros caracteres, como los números y los signos de puntuación, en forma digital. Uno de los primeros fue el código Morse, que utiliza puntos y rayas para representar los caracteres. Desde el punto de vista del ordenador, el código Morse tiene el inconveniente de que emplea distintas cantidades de bits para diferentes letras: entre uno y seis puntos y rayas para cada carácter. Otros intentos tendentes a elaborar un código de caracteres más regular y sistemático (por ejemplo, el código Baudot, que utiliza cinco bits para representar hasta 32 caracteres) han fracasado en algún punto y, en la actualidad, el sistema que se ha adoptado con carácter casi universal es el código ASCII (*American Standard Code for Information Interchange*: código norteamericano estándar para intercambio de información).

El código ASCII emplea sólo un byte para representar los 94 caracteres imprimibles, el "espacio" y un número de "caracteres" de control. Ocho bits podrían dar 256 (2^8) combinaciones únicas, pero esto es mucho más de lo que se necesita para representar los caracteres del teclado de una máquina de escribir o un ordenador corrientes, de modo que sólo se utilizan siete, que permiten 128 combinaciones exclusivas. (El octavo bit normalmente no se usa, pero en algunas ocasiones se emplea para especificar un juego alternativo en un idioma diferente o un juego de caracteres para gráficos.) En la tabla que incluimos en la p. 147 se proporcionan los códigos ASCII binario y decimal para la gama estándar de caracteres.

Como puede observar a partir de la tabla, el código ASCII para la letra A es 65 y para la B, 66. Los códigos para las letras minúsculas a y b son 97 y 98. Cada letra minúscula tiene en el código ASCII un valor igual al valor de su equivalente en mayúsculas más 32. Esta equivalencia constante hace que resulte muy fácil convertir las series de caracteres de letras minúsculas en letras mayúsculas, y viceversa. Para realizar esta conversión necesitaríamos otras dos funciones que hasta ahora no habíamos utilizado nunca en nuestro curso de programación BASIC: ASC y CHR$.

La función ASC toma un carácter imprimible y lo devuelve con su equivalente en código ASCII, de modo que PRINT ASC("A") imprimiría en pantalla el número 65; PRINT ASC("b") imprimiría el 98.

La función CHR$ hace lo contrario: toma un número, da por sentado que está en código ASCII y devuelve el carácter que representa. De manera, entonces, que PRINT CHR$(65) imprimiría A, mientras que PRINT CHR$ (98) imprimiría b. Las funciones CHR$ y ASC se emplean mucho, junto con LEFT$, RIGHT$ y MID$, en los programas que se valen en gran medida de las series de caracteres. A continuación le ofrecemos un breve programa que acepta un carácter del teclado, verifica si es una mayúscula, y, si no lo es, lo convierte en mayúscula:

```
10 REM CONVERTIDOR DE MINUSCULA
   A MAYUSCULA
20 PRINT "DE ENTRADA A UN CARACTER"
30 INPUT C$
40 LET C = ASC(C$)
50 IF C > 90 THEN LET C = C -32
60 PRINT CHR$(C)
```

En futuros capítulos de nuestro curso de programación veremos más a fondo este tipo de manipulación de variables.

Por último, en este resumen, echemos una mirada a funciones de las que quizá carezca la versión de BASIC del usuario. Casi todas las versiones de este lenguaje permiten que el programador cree nuevas funciones, y éstas son casi tan fáciles de utilizar como las incorporadas. La sentencia DEF le señala al BASIC que se está definiendo una nueva función. A continuación le mostramos cómo definir una función para calcular el volumen de una esfera (la fórmula es $V = \frac{4}{3}\pi r^3$, donde r es el radio de la esfera y π (pi) es la constante aproximadamente igual a 3,14159):

```
10 REM FUNCION PARA CALCULAR EL VOLUMEN
   DE UNA ESFERA
20 DEF FNV(X) = 4 * 3.14159 * X * X * X/3
30 PRINT "DE ENTRADA AL RADIO DE LA ESFERA"
40 INPUT R
50 PRINT "EL VOLUMEN DE UNA ESFERA DE
   RADIO";R;"ES"
60 PRINT FNV(R)
70 END
```

Esta forma de definir una función es bastante directa, pero observemos la línea con todo detalle:

```
DEFine   identificador de función
   ↓     ↓
20 DEF FNV(X) = 4 * 3.14159 * X * X * X/3
       ↑  ↑
   FuNción  variable ficticia
```

Cuando se define la función, a las letras FN les sigue una letra identificadora (V, en el caso de la función de arriba) y luego a ésta le sigue una "variable ficticia". Esta variable ficticia también se debe utilizar en la definición de la función a la derecha del signo igual. Cuando la función se emplea en un programa, en el lugar de la variable ficticia de la definición se puede utilizar cualquier variable numérica.

En un punto posterior del programa anterior sería igualmente posible hacer uso de la función "volumen de una esfera" del siguiente modo:

```
 999 LET A = 66
1000 LET B = FNV(A)
1010 PRINT B
1020 LET C = 5
```

```
1030 LET D = B + FNV(C)
1040 PRINT D
1050 LET G = FNV(16)
1060 PRINT G
```

Algunas versiones de BASIC permiten utilizar variables múltiples en la función definida. En consecuencia, una función para hallar el promedio de dos números se podría escribir:

```
100 DEF FNP(B,C) = (B + C)/2
110 INPUT "DE ENTRADA A DOS NUMEROS";B,C
120 LET P = FNP(B,C): REM LA FUNCION
      'PROMEDIO'
130 PRINT "EL PROMEDIO DE";B;"Y";C;"ES";P
```

Observe que la línea 110 combina en una sentencia el equivalente de dos separadas. La mayoría de los BASIC imprimirán automáticamente las palabras encerradas entre comillas dobles a continuación de la sentencia INPUT, de modo que a tenor de dicha característica, esta línea equivale a:

```
110 PRINT "DE ENTRADA A DOS NUMEROS"
115 INPUT B,C
```

La línea 120 también consigue incluir el equivalente de dos sentencias en una línea empleando los dos puntos (:) como separador. Las sentencias que normalmente pertenecerían a líneas separadas se pueden escribir en una sola línea siempre y cuando cada sentencia "independiente" esté separada de la anterior por los dos puntos. En los programas largos esto puede contribuir a ahorrar espacio, pero no es conveniente abusar de su utilización, por una parte, porque dificulta bastante la lectura de los programas y, por otra, porque aumenta las posibilidades de error.

Ahora sí hemos tratado todos los puntos principales del lenguaje BASIC. En próximos capítulos del curso de programación BASIC estudiaremos el desarrollo y el diseño del programa, en lugar de detalles relativos al BASIC.

Respuestas a los "Ejercicios" de la página 137

Bugs

Se producirá un "ERROR FUERA DE DATOS", porque en la sentencia DATA de la línea 130 debería haber un total de 12 valores. En segundo lugar surgirá un error en la línea 100, cuando intenta direccionar un elemento A(4,1). La línea 100 debería decir:

```
100 PRINT A(X,Y)
```

Asignación de valores

Ahora exponemos una versión que llevaría a cabo las funciones requeridas. Es posible que su programa sea distinto

```
10  DIM A(8,13)
20  FOR F = 1 TO 7
30  FOR C = 1 TO 12
40  READ A(F,C)
50  NEXT C
60  NEXT F
70  REM SUMAR TOTALES
80  GOSUB 300
90  REM IMPRESION DATOS REQUERIDOS
100 GOSUB 200
110 PRINT "¿MAS DATOS?"
120 PRINT "SI(S) O NO(N)"
130 INPUT A$
140 IF A$ = "N" THEN GOTO 160
150 GOTO 100
160 END
200 PRINT "¿QUE MES?"
210 PRINT "1-PARA ENERO,"
220 PRINT "13 PARA TOTAL, ETC"
230 INPUT M
240 PRINT "¿QUE GASTO?"
250 PRINT "1-PARA GASOLINA"
260 PRINT "8-PARA TOTAL, ETC"
270 INPUT X
280 PRINT "EL VALOR ES ";A(X,M)
290 RETURN
300 FOR F = 1 TO 7
310 LET T = 0
320 FOR C = 1 TO 12
330 LET T = T + A (F,C)
340 NEXT C
350 LET A(F,13) = T
360 NEXT F
370 FOR C = 1 TO 13
380 LET T = 0
390 FOR F = 1 TO 7
400 LET T = T + A (F,C)
410 NEXT F
420 LET A (8,C) = T
430 NEXT C
440 RETURN
500 REM SUS DATOS SIGUEN AQUI
510 REM OCHENTA Y CUATRO VALORES
520 REM DATOS 3100, 2600, ETC
```

Complementos al BASIC

ASCII

El Dragon-32 posee una versión de ASCII no estandarizada, que no admite caracteres en minúscula, de modo que el programa convertidor de minúsculas a mayúsculas no podrá realizar su función; inténtelo, a pesar de ello, y lea el manual del ordenador para recabar más información acerca del juego de caracteres del modelo.

DEF FN(A,B)

En el Oric-1, el Vic-20, el Dragon-32 y el Commodore 64 sólo se puede incluir una variable en el interior de los paréntesis

ASC( )

En el Spectrum y el ZX81, reemplace:

ASC(A$) por CODE A$

y sustituya:

CHR$ (65) por CHR$ 65

CHR$( )

Si el argumento es una expresión, se debe colocar entre paréntesis. Sin embargo, los argumentos simples (como A$ y 65) no necesitan paréntesis.

ON... GOSUB

El ZX81, el Spectrum y el Lynx no disponen de estas sentencias.

ON... GOTO

INPUT" "

En el BBC Micro sustituya

```
INPUT "CUALQUIER MENSAJE";MS$
```

por

```
INPUT "CUALQUIER MENSAJE",MS$
```

DEF FN

En el BBC Micro el usuario debe definir las funciones al final del programa, después de la palabra END o STOP, y no al comienzo del mismo, como en nuestro ejemplo. En este caso el BASIC del BBC es más eficaz que el BASIC estándar, de modo que para obtener mayor información consulte su manual.

STEP 0

En el Oric-1, en los dos fragmentos de programa que demuestran una buena estructura de bucle, reemplace:

```
IF INKEY$ = " " THEN LET X = 2
```

por

```
IF KEY$ = " " THEN LET X = 1
```

En el Dragon-32 sustitúyalo por:

```
IF INKEY$ = " " THEN LET X = 1
```

En el Vic-20 y en el Commodore 64 reemplace:

```
IF INKEY$ = " " THEN LET X = 2
```

por

```
GET A$: IF A$ = " " THEN LET X = 1
```

El acoplador acústico

Este dispositivo convierte datos digitales en tonos audibles y viceversa. Acoplado a un teléfono, permite comunicarse con otros ordenadores situados en cualquier lugar del planeta

Conectar su ordenador personal a una impresora o a un conjunto de unidades de disco es relativamente sencillo, porque generalmente éstos se hallarán en el mismo cuarto e, incluso, sobre la misma mesa. Pero conectar un ordenador con otros aparatos más grandes en su oficina o en cualquier otro lugar del mundo es ya una pretensión muy distinta. Por suerte, nosotros ya disponemos de un medio para la comunicación a través de todo el globo terráqueo: la red telefónica. Todo lo que se necesita es conectar el ordenador a este sistema.

Como la red telefónica es utilizada tan ampliamente por los circuitos de grandes ordenadores (los sistemas de venta de billetes para aviones de líneas comerciales, por ejemplo), la tecnología ya está bien establecida. Todo lo que necesita el usuario de un ordenador personal es una versión más barata y más sencilla de esta tecnología. El método convencional para conectar los ordenadores al sistema telefónico es un dispositivo denominado *modem*. Este extraño término no es más que un sencillo acrónimo formado a partir de las palabras modulador-demodulador.

El dispositivo funciona de forma muy parecida a la interface para cassette de un micro. Los patrones de unos y ceros binarios se convierten en señales eléctricas en dos frecuencias audibles diferentes y luego se envían a través de las líneas telefónicas (éste es el proceso de modulación). Al otro extremo, las frecuencias audibles se "demodulan" otra vez en unos y ceros. El modem envía una frecuencia constante (denominada *tono portador*), tanto si está enviando realmente información como si no, que hace que el ordenador receptor sepa que la línea aún está conectada.

El principal inconveniente de un modem es que ha de estar permanentemente acoplado por cable a la red telefónica, monopolizando su utilización, lo cual, en cierto sentido, representa una incomodidad para el usuario personal.

Un método de comunicación alternativo es el acoplador acústico. Dado que el sistema emplea tonos que se pueden oír, nada impide que éstos se puedan generar acústicamente, utilizando un altavoz. Éste se puede conectar luego al teléfono para que transmita a través del mismo. Al otro extremo, un micrófono colocado en contacto con el auricular del teléfono recogería la señal transmitida. Para esto está diseñado el acoplador acústico. A diferencia del modem, no necesita estar conectado permanentemente al teléfono.

Existen en el mercado diversos tipos de acoplador acústico, que van desde el dispositivo que muestra la ilustración, que es lo suficientemente compacto como para poderse conectar a un ordenador portátil, hasta unidades más grandes para escritorio. Las unidades sofisticadas se pueden utilizar para responder automáticamente a las llamadas que se reciben, sin necesidad de que esté presente el operador del ordenador, a través de la escucha constante del tono portador. Así como las interfaces para cassette varían en cuanto a la velocidad a la cual pueden almacenar y recuperar la información, la misma variación existe para los acopladores acústicos. No obstante, la gama de velocidades es estrictamente limitada. Las características de transmisión de un cable telefónico impiden que cualquier señal que supere los 1 200 caracteres por segundo (c.p.s.) pueda transmitirse con un nivel de fiabilidad razonable.

Las unidades económicas pueden funcionar a velocidades tan bajas como 30 c.p.s., mientras que los modelos más caros incorporan un interruptor para seleccionar una variedad de velocidades. Sin embargo, lo que hay que tener presente en todos los casos es que los dispositivos situados a ambos extremos de la línea telefónica siempre deben operar a la misma velocidad, ya que de lo contrario no se produciría la transmisión.

La utilización cada vez mayor de ordenadores personales en las gestiones económicas ha dado como resultado el desarrollo de gran cantidad de productos nuevos. Dispositivos como el Sendata y similares permiten que ordenadores portátiles como el Tandy 100 y el Epson HX-20 puedan ser utilizados como terminales de ordenador a distancia por cualquier persona, desde un periodista hasta un agente de ventas. Es posible darles entrada en la memoria del ordenador a los

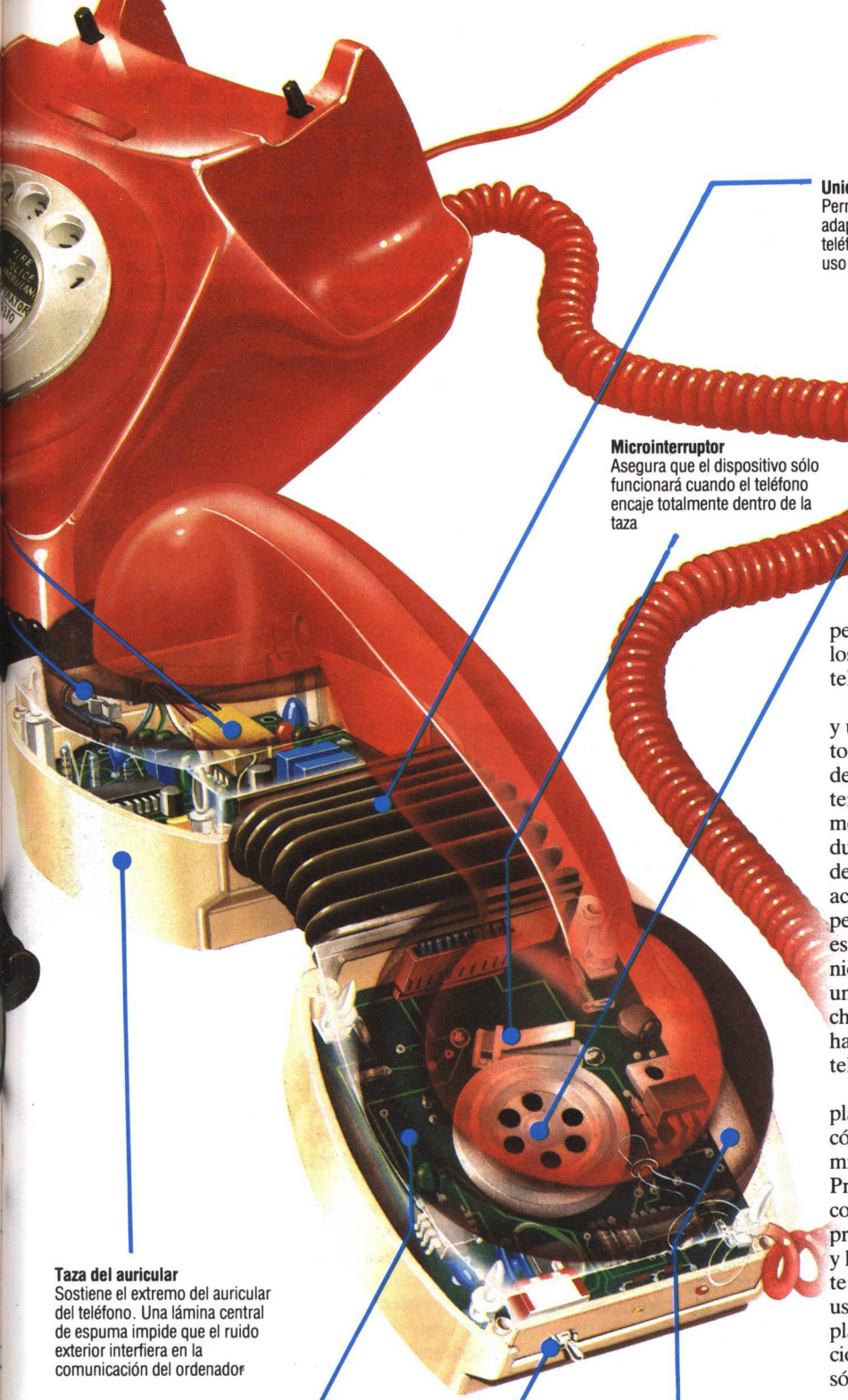
Marcus Wilson-Smith

En una cabina telefónica
Los acopladores acústicos de poco peso permiten que el usuario de ordenador, estando de viaje, tenga acceso a la información de otros ordenadores, situados en cualquier lugar del mundo, a través de la red de teléfonos públicos

Micrófono
Recoge la señal proveniente del altavoz del teléfono y alimenta con ella el tablero del circuito

Enchufe red
Le proporciona la alimentación eléctrica al acoplador desde un transformador adecuado. También se utiliza para recargar las pilas internas de niquel-cadmio

Cable para interface
Conecta con el enchufe para interface (en serie) RS232 del ordenador



Unión flexible
Permite que el acoplador se adapte a la mayoría de los teléfonos más comúnmente en uso

Microinterruptor
Asegura que el dispositivo sólo funcionará cuando el teléfono encaje totalmente dentro de la taza

Altavoz
Un acoplador acústico es, en realidad, lo contrario de un teléfono. Este altavoz lee los datos en forma de una tonalidad audible

Trevor Hill

Taza del auricular
Sostiene el extremo del auricular del teléfono. Una lámina central de espuma impide que el ruido exterior interfiera en la comunicación del ordenador

Circuito impreso
Estos componentes electrónicos no sólo regulan los interfaces con el microordenador, sino que convierten los unos y ceros en dos frecuencias distintas

Selector de modalidad
Este interruptor determina si el acoplador originará la llamada o si, por el contrario, responderá a ella

Taza del micrófono
El extremo del micrófono del teléfono encaja dentro de esta taza

pedidos, los artículos y la correspondencia, enviándolos luego a la oficina principal a través de las líneas telefónicas.

En el campo de los negocios, un acoplador acústico y una terminal de ordenador permiten el acceso directo a una amplia gama de servicios de información y departamentos de informática. La forma de pedir existencias de una cadena de farmacias está ahora totalmente informatizada: el personal da entrada a los productos y sus cantidades y luego los transmite al sistema de ordenadores del almacén central. En el hogar, el acoplador acústico ofrece la importante ventaja, respecto a un modem convencional, de que no necesita estar permanentemente conectado a las líneas telefónicas. Un ejecutivo que está trabajando en su casa con un ordenador personal puede contactar con su despacho para enviar o recibir información sin que para ello haya de tener ocupada de manera continua la línea telefónica.

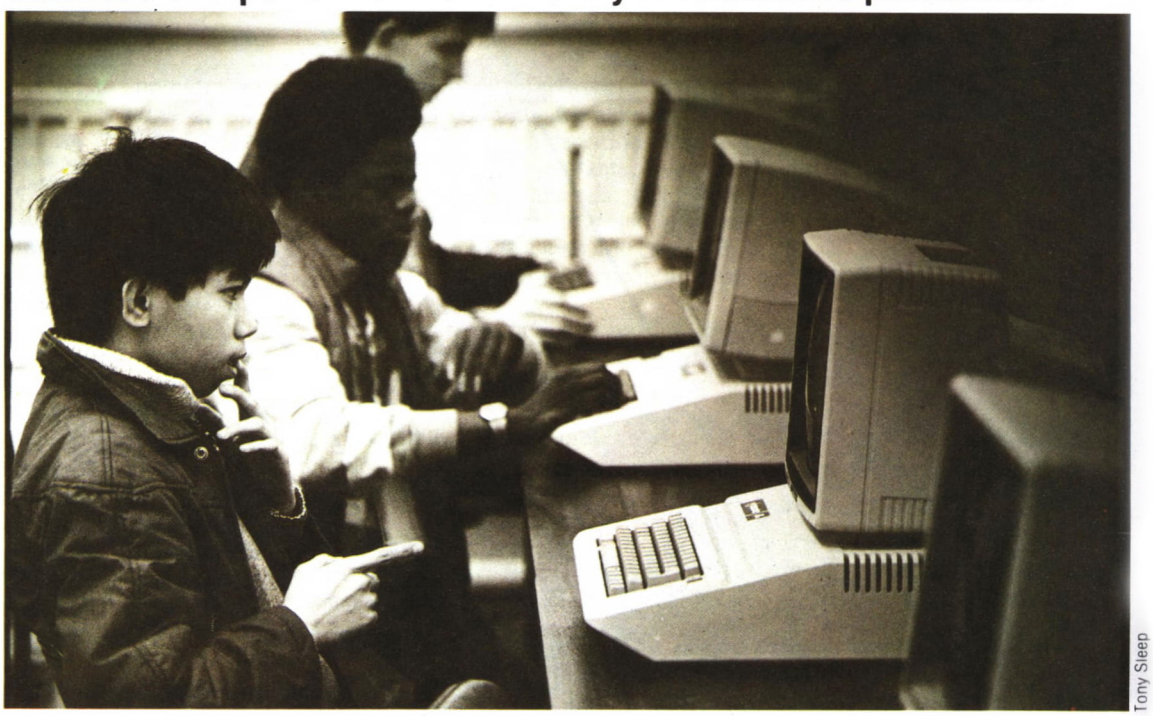
En el mercado de ordenadores personales, el acoplador acústico está proporcionando una alternativa cómoda y económica al modem convencional al permitir el acceso a las bases de datos públicas como la Prestel y la Micronet 800. También este dispositivo constituye una forma mucho más fiable de enviarles programas a los amigos en vez de grabarlos en cassette y hacerlos llegar a través del servicio de Correos. Existe también el correo electrónico, que le permite al usuario de ordenadores personales el acceso a un acoplador acústico capaz de proporcionar una comunicación instantánea, procedimiento que anteriormente sólo estaba al alcance de las empresas más grandes.

No obstante, un acoplador acústico europeo no se puede comunicar con uno norteamericano. La industria de ordenadores de Estados Unidos emplea el sistema denominado Bell 103 y Europa utiliza el CCITT V21. Ambos son incompatibles.

Una complicación adicional es la que plantea la nueva generación de teléfonos que se está introduciendo actualmente, y que parecen no adaptarse bien a la estructura (véase ilustración) de muchos acopladores acústicos. Dado que el sistema trabaja transmitiendo sonidos, es importante evitar que ruidos exteriores interfieran en la comunicación. Si el teléfono no se adapta correctamente o si hay muchos ruidos exteriores, existen muchas posibilidades de que los datos enviados se confundan.

Redes de información

Los centros de trabajo por ordenador de un mismo edificio se pueden enlazar entre sí con una red de área local. Así, varios usuarios comparten la información y los costosos periféricos



Tony Steep

El uso de las redes se está generalizando en los centros de enseñanza, como el que muestra la fotografía, donde los jóvenes aprenden a utilizar los ordenadores y la microelectrónica

Ahora que se está haciendo familiar la presencia de los micros en las escuelas y oficinas, es cada vez más factible la posibilidad de que existan en un mismo edificio, e incluso en la misma dependencia, máquinas compatibles. En la medida en que este hecho se hace realidad, inevitablemente se tiende a pensar en los métodos que podrían utilizarse para vincular estas máquinas, aunque sólo fuera para compartir dispositivos periféricos como unidades de disco o impresoras. Los accesorios de esta clase son caros y compartirlos entre varios micros es una forma económica de emplearlos.

Pero la unión de periféricos no es, de ninguna manera, el único beneficio que se obtiene a partir del enlace de máquinas entre sí. Los micros se pueden montar a fin de que se comuniquen entre ellos formando una "red". Cuando todas las máquinas o centros de trabajo están situados en un solo edificio, se aplica el término *red de área local* (*Local Area Network*: LAN).

Se ha hablado mucho acerca de la posibilidad de abandonar el uso del papel en la oficina. Un primer paso hacia ello sería el de reemplazar el memorándum escrito por un mensaje que aparezca, desde una fuente lejana, en la pantalla de un monitor. La mayoría de las redes de área local, y también el Prestel, disponen de esta facilidad de "buzón". Para no distraer el trabajo que esté realizando en ese momento quien lo recibe, el mensaje recién llegado se anuncia en la línea inferior de la pantalla.

La aplicación de una red de área local puede ser muy útil en las escuelas. El texto se puede "reflejar" como en un espejo desde el micro del maestro a cualquiera o a todos los centros de trabajo de los estudiantes, o el maestro puede aprovechar esta instalación electrónica para revisar el trabajo de cada alumno, haciéndole comentarios y sugerencias.

La misma red de área local posibilita el empleo de una misma fuente común de información por usuarios diferentes. Tal vez la aplicación más frecuente de tal utilización se realice en las bases de datos que contienen información tanto pública como privada; el Prestel y los otros sistemas de videotexto constituyen ejemplo de ello.

Cuando un número determinado de personas está empleando el mismo archivo de información, es esencial asegurar que la información que éste contiene no se puede modificar sin que se les advierta a todos los usuarios. Por ejemplo, una fábrica puede utilizar una red de ordenadores para retener la información relativa a la disponibilidad de componentes y materias primas. Si a cada uno de los usuarios no se le presentara idéntica información, en el mejor de los casos no habría más que confusión y, en el peor, se le asignaría mercadería inexistente a quizá más de un departamento a la vez.

Al utilizar microordenadores con 64 Kbytes de RAM o más, resulta tentador "cargar", digamos, una sección de una base de datos y luego no volver a remitirse al archivo maestro hasta que se necesite examinar un segmento diferente. A menos que el software de control sea lo suficientemente amplio como para detectar y reflejar los cambios producidos en la información que contiene esa sección cargada, el usuario se podría estar refiriendo a cifras desfasadas.

El precio de gran parte del hardware de los ordenadores tiende a bajar, pero a menudo los periféricos

más avanzados técnicamente suelen costar más que el microordenador que los controla. Un buen ejemplo de esto es el Winchester, un disco rígido que funciona en un ambiente herméticamente cerrado (al vacío), y que ofrece parte de la velocidad y capacidad de almacenamiento de las unidades de disco principales. Los discos Winchester (así llamados algo arbitrariamente aludiendo al rifle de repetición Winchester 30/30, porque originalmente se construyeron como discos gemelos, cada uno de ellos con 30 megabytes de capacidad) son necesarios por el volumen de transacciones que maneja el software comercial. Sin embargo, tienen suficiente capacidad libre, aun en los emplazamientos de intensa actividad, como para hacer frente a más de un usuario de forma simultánea. Del mismo modo, mientras que una impresora matricial será adecuada para muchos trabajos, el tratamiento de textos suele requerir una impresora margarita, que suele costar más dinero. A menos que un solo usuario la pueda mantener en funcionamiento constante, para compensar su precio es conveniente que la impresora sea compartida por varios centros de trabajo.

Las redes ofrecen, asimismo, otras posibilidades. Los documentos que exigen la atención de varias personas se pueden traspasar de una a otra sin necesidad de imprimirlos en papel. Esto sucede, por ejemplo, en las redacciones de los periódicos y revistas técnicamente más adelantadas. El autor crea el material escrito a máquina, que luego entrega al redactor para que le dé su opinión crítica. Luego interviene el corrector, quien se encarga de revisar la ortografía, la gramática y el estilo del texto, marca las especificaciones técnicas y lo entrega al impresor, quien comienza el proceso de impresión. Antes de que existieran las redes de área local, las etapas por las que tenía que pasar el texto mecanografiado antes de llegar al impresor eran más diferenciadas y menos expeditas.

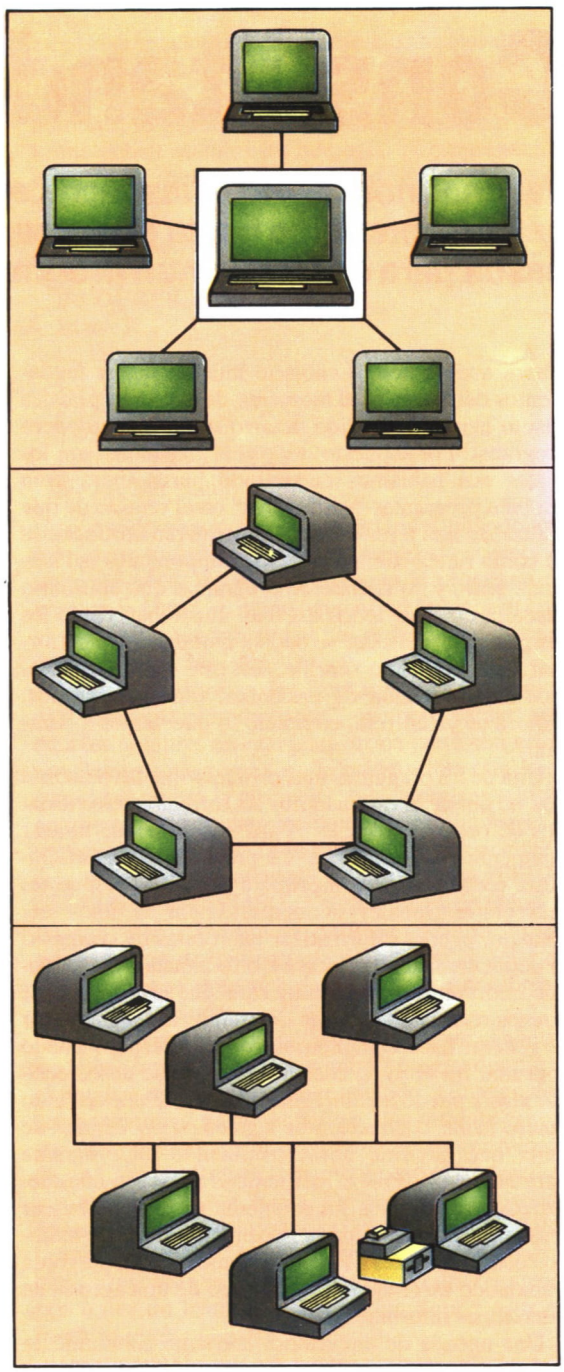
Los sistemas de red están a disposición de la mayoría de los ordenadores personales. Uno de los más utilizados en Gran Bretaña es el Econet de Acorn, concebido para el microordenador BBC. El Econet designa a una de las máquinas de la red para que actúe como "mensajero del archivo", que cuida de la unidad de disco central y se ocupa de las diversas solicitudes de información. Esta máquina puede estar dedicada exclusivamente a esa finalidad, o bien puede estar disponible para un usuario cualquiera siempre que no se la necesite para proporcionar un servicio a los otros miembros de la red. Si ésta comparte una impresora, se deberá disponer asimismo de una máquina destinada a controlarla.

El Econet puede abarcar hasta 254 centros de trabajo más los dos centros de "servicio", pero una limitación mucho más realista en cuanto a las dimensiones de la red viene dada por la mayor distancia a la cual puede estar situada la central más alejada respecto a la "unidad de reloj": un máximo de 500 metros. La unidad de reloj es una caja separada, también incluida en la red, que controla la velocidad a que se envía la información a través del sistema. El Econet utiliza dos pares de cables, como el sistema telefónico, y, al igual que éste, su instalación es relativamente sencilla. Uno de los pares transporta datos, y el otro los impulsos del reloj necesarios para asegurar la sincronización.

El software de comunicaciones del Econet, bastante sofisticado, reside en una EPROM (*Erasable Programmable Read-Only Memory:* memoria programable de lectura solamente, que puede borrarse) de ocho Kbytes situada en cada uno de los centros de trabajo. La tarea más complicada del sistema es la de evitar las "colisiones", es decir, asegurar que sólo un miembro de la red esté transmitiendo en un momento dado. Existen otras redes similares basadas en ordenadores personales, si bien por lo general no ofrecen en igual medida estas amplias configuraciones.

Las redes basadas en miniordenadores y ordenadores de unidad principal han alcanzado un nivel de utilización masivo en el curso de los diez últimos años, y no están restringidas a un solo país. Muchas líneas aéreas utilizan sistemas de reserva y venta de billetes que abarcan todo el globo, transmitiendo sus datos por las líneas telefónicas o por satélite.

Con la actual generalización de la televisión por cable, resulta razonable esperar que aumente la utilización de las redes, centrada probablemente en un concepto similar al del Micronet 800, un sistema basado en el Prestel, que permite que los programas se carguen al microordenador BBC a través de las líneas telefónicas.



Kevin Jones

Tipos de redes

Star
Una red star (en estrella) conecta a los usuarios de cada máquina con un controlador central, que también dirigirá a los periféricos que se utilicen en común

Ring
Algunas redes exigen que los usuarios de los ordenadores estén unidos entre sí en un bucle continuo. Estas redes están menos popularizadas porque los datos han de pasar a través de la mitad de las máquinas del anillo antes de que lleguen a su destino

Bus
El diseño de redes más refinadas, como por ejemplo la Econet, es muy similar, en cuanto a su concepto, a la arquitectura de un microordenador moderno: los datos y los mensajes de control pasan directamente de un usuario a otro

Consultando la agenda

Valiéndonos de todas las técnicas de la programación en BASIC que hemos aprendido hasta el momento, damos ahora los primeros pasos para desarrollar un programa de base de datos

Ahora que ya hemos cubierto muchos de los fundamentos del BASIC, es el momento de poner en práctica cuanto hemos aprendido desarrollando un verdadero programa. Por supuesto, todos los programas con los cuales nos habíamos encontrado hasta ahora eran también programas "verdaderos" en el sentido de que realizaban una tarea específica, pero eran ilustraciones de cómo funcionan los diversos componentes del lenguaje BASIC y no la clase de programas que el usuario desearía emplear todos los días. Ilustraban cómo los "engranajes" del BASIC se podían unir entre sí para formar un mecanismo sencillo. En este momento estamos en disposición de ensamblar esos mecanismos. ¡Ahora es ya un reloj completo lo que vamos a construir!

Una de las preguntas más comunes que las personas que no emplean ordenadores les formulan a los usuarios de ordenadores es: "¿Para qué se puede utilizar realmente un ordenador?" La pregunta no es tan simplista como parecería a primera vista. Las respuestas más convencionales tienden a seguir la línea de: "Bueno, puedes informatizar tus recetas de cocina" o "Puedes crear una agenda de direcciones o de teléfonos informatizada". Es muy raro que esta táctica dé buenos resultados, porque quien hizo la pregunta por lo general hace comentarios como: "Pero yo puedo consultar mi libro de cocina cada vez que deseo cocinar algo y puedo leer mi agenda de direcciones cuando quiero hallar la dirección de alguien, sin necesidad de tener que pasarme horas escribiendo un programa para hacerlo". ¿Qué circunstancias se deben considerar para decidir si a un problema se le debe buscar respuesta por medio de la informática y no por métodos convencionales? Responderemos a esta pregunta trabajando en el ejemplo específico de una agenda de direcciones informatizada.

Una agenda de direcciones corriente comúnmente consta de un índice alfabético que se puede manipular con los dedos, diseñado para que el usuario pueda localizar, muy aproximadamente, un nombre determinado. Por lo general, y cuando es necesario, se suelen ir agregando otros nombres, y no por orden estrictamente alfabético. Su primera entrada en la página de la P podría ser Daniel Puig. Después se podría agregar Alfonso Prado o Camilo Pérez. Aunque estos nombres no está ordenados alfabéticamente, están agrupados juntos bajo la P, de modo que la tarea de hallar un apellido determinado que comience con P no será demasiado complicada. Por otra parte, si no se utilizara ningún tipo de índice, localizar un nombre sería un verdadero problema.

Las otras entradas corrientes en una agenda de direcciones son las señas y el número de teléfono de la persona e, incluso, alguna información personal. Sin embargo, una agenda de direcciones convencional no le puede dar una lista separada de todas las personas que viven en Barcelona, o a quién le corresponde un determinado número de teléfono. Puede que esto no represente un inconveniente grave, pero si usted fuera propietario de una pequeña empresa de ventas por correo sería una ayuda muy valiosa que pudiera obtener información específica acerca de las personas de su cartera de clientes. Por ejemplo, si quisiera distribuir una nueva línea de pijamas para niños podría anticipar nuevos pedidos informando a sus clientes, pero para ahorrarse algún dinero en sellos de Correo probablemente no valdría la pena enviarles los folletos a los clientes que no tuvieran hijos. Éste es el tipo de consideraciones que se deben evaluar antes de decidir si, frente a un problema dado, es más conveniente una solución informática o una solución convencional.

Si la solución informática es apropiada, la siguiente consideración ha de ser, entonces, si comprar o no un software existente a nivel comercial. Una mirada a los anuncios de las revistas de informática sugeriría que los programadores de ordenadores ya han pensado en todas las eventualidades posibles. Sin embargo, un examen más detenido podría dejar en evidencia el hecho de que un programa disponible comercialmente podría no ajustarse con toda exactitud a lo que se desea, o que no exista para su modelo de ordenador, o que sea demasiado caro. El costo de un programa refleja generalmente los costos de desarrollo. Un paquete de tratamiento de textos puede ser caro, pero si el usuario decidiera escribirlo él mismo y tuviera que destinar seis meses de dedicación exclusiva a su realización, sin duda le resultaría mucho más caro.

Desde el punto de vista de lo positivo, el software que se escriba usted mismo podría hacer exactamente lo que deseara que hiciera. El otro factor es la incomparable satisfacción de que, solo y sin ayuda de nadie, escriba usted un programa con el que obtenga un éxito rotundo.

El diseño de un programa consta de varias fases, la primera de las cuales es la comprensión profunda del problema, premisa que implica una descripción clara de dicho problema.

La segunda consiste en hallar un enfoque para solucionar el problema. Esto implica una descripción de la forma esperada de la entrada y la salida como un "primer nivel de descripción" del problema. Los problemas y las soluciones se deben plantear en los términos más amplios y éstos se han de concretar gradualmente hasta llegar a la etapa en la cual los podemos codificar en un lenguaje determinado.

La tercera etapa es la codificación en sí misma. Utilizaremos el BASIC como nuestro lenguaje de alto nivel, pero en lugar de éste podríamos emplear igualmente cualquier otro lenguaje. Hasta la etapa final de codificación en BASIC utilizaremos un seudolenguaje intermedio entre la libertad y flexibilidad del lenguaje corriente y las estructuras rígidas de un verdadero lenguaje para ordenador como el BASIC.

El enfoque a la programación que acabamos de des-

cribir se denomina generalmente programación *top-down* (de arriba/abajo). Se trabaja desde el nivel superior (una enunciación general de los objetivos globales), a través de varios niveles de refinamiento, hacia abajo, hasta los detalles más precisos del programa necesarios para empezar a codificar en el lenguaje de alto nivel escogido. También intentaremos ceñirnos a los principios de la denominada "programación estructurada". Estos principios se irán clarificando a través del desarrollo de este proyecto.

Los pasos que seguiremos en el desarrollo del programa se pueden resumir de la siguiente manera:

1. Una clara enunciación del problema
2. La forma de la entrada y la salida (primer nivel de descripción)
   2.1 Refinamiento (segundo nivel de descripción)
   2.2 Sucesivo refinamiento (del tercer al enésimo nivel de descripción)
3. Codificación en lenguaje de alto nivel

Antes de comprometernos en un proyecto importante de software, es esencial plantear el problema con toda claridad. Se trata de un ejercicio que no es en absoluto trivial. Probemos con algunas ideas para nuestra agenda de direcciones informatizada.

Empezaremos primero por una lista de las características deseables; luego podemos decidir cuáles de ellas se pueden emplear con un esfuerzo de programación razonable. Deseamos que se pueda:

1. Localizar una dirección, un número de teléfono y notas dando entrada a un nombre desde el teclado
2. Obtener una lista de nombres, direcciones y números de teléfono dando entrada a sólo parte de un nombre (sólo el primer apellido o el nombre)
3. Obtener una lista de nombres, direcciones y números de teléfono de una ciudad o unas zonas determinadas
4. Obtener un listado de todos los nombres que comienzan por una letra determinada
5. Obtener un listado completo de todos los nombres de la agenda de direcciones, ordenados alfabéticamente
6. Agregar todas las entradas nuevas que deseemos
7. Modificar la información ya existente
8. Suprimir los datos que deseemos

Supongamos que ya hemos escrito el programa para la agenda de direcciones. ¿Qué forma deberían asumir la entrada y la salida? ¿Cómo le gustaría que funcionara el programa desde el punto de vista del usuario? En términos generales, los programas pueden ser "activados por menú", "activados por órdenes" o por una combinación de ambos. En un programa activado por menú, cada vez que se ha de tomar una decisión se le ofrece al usuario una lista (menú) de opciones. La selección se suele efectuar pulsando sólo una tecla. En los programas activados por órdenes, el usuario digita las palabras-órdenes o frases-órdenes específicas, por lo general sin que se le requiera en ese sentido. Algunos programas combinan ambas técnicas. La ventaja de un programa activado por menú es que a un recién iniciado le debería resultar sencillo de utilizar, logrando que el programa fuera más "amable con el usuario". Un programa activado por órdenes debería resultarle más expedito al usuario experimentado. Nosotros optaremos por un enfoque activado por menú, si bien usted está en libertad de decidirse por ejecutar este programa utilizando, en cambio, rutinas de órdenes: la decisión es suya.

Dado que el programa se centrará en una lista de nombres, lo primero que debemos considerar es qué forma deben asumir esos nombres. ¿Comprenderá el ordenador, por ejemplo, todos los formatos siguientes?:

A. J. P. Soler
Leonardo da Vinci
Bo DEREK
Juana R.
L. Prado
j. j. García
F López
Twiggy
GROUCHO MARX
Sir Freddie Smith

Esto puede sonar como que estamos hilando muy fino, pero consideremos lo que sucedería si diera entrada a F López y después le pidiera al programa que buscara F. López. A menos que usted hubiera previsto el problema, probablemente el ordenador le respondería NOMBRE NO HALLADO.

Existen dos formas de afrontar el problema: podemos tener una entrada "libre", que permita dar entrada a los nombres en cualquier forma junto con rutinas inteligentes que tomen en consideración este hecho cuando se realiza la búsqueda; o bien podemos insistir en que se dé entrada a los nombres en una forma estrictamente definida. Todo nombre que no se ciñera a ésta produciría un mensaje de error como FORMATO DE NOMBRE INACEPTABLE. La elección es de índole arbitraria, pero tenemos la posibilidad de inclinarnos por la opción de una entrada muy "libre" y dejar que el programa se preocupe de convertir los nombres en una forma estándar.

Desde el punto de vista de una búsqueda alfabética, podemos imaginar que los nombres poseen dos partes: el nombre de pila y el primer apellido. Un apellido es relativamente sencillo de definir: una serie de caracteres alfabéticos en mayúsculas o minúsculas terminados en un Carriage Return (retorno de carro) y precedidos por un espacio (ASCII 32). De inmediato se presenta un problema: ¿qué sucedería si se diera entrada sólo al nombre "Twiggy" sin estar precedido por un espacio? Presumiblemente el programa lo rechazaría por tener un formato inaceptable. Será mejor que modifiquemos nuestra definición.

Consideremos que un nombre consta de una o dos partes: el apellido o el apellido y el nombre de pila. El nombre puede incluir tanto caracteres en mayúsculas o en minúsculas, como puntos, apóstrofos y guiones. Siempre comenzará con un carácter alfabético y terminará con un "Carriage Return" (no se admitirán puntos al terminar). De haber un espacio, el último grupo de caracteres (incluyendo apóstrofos y guiones) se contará como el apellido y otras partes, incluyendo el espacio, se contarán como el nombre de pila. De no haber espacio, el nombre completo se considerará como el apellido.

El apellido requiere una consideración especial porque, en el ordenamiento alfabético, precede siempre a los nombres de pila. De manera que Ana Pérez vendría después de Victoria Paredes. Si un nombre se compone de solo un grupo de caracteres, como Trevanian, Twiggy o un apodo como "Gordo", a los efectos de nuestro programa se lo puede considerar como un apellido.

En una búsqueda alfabética, ¿qué nombre iría primero: A.J.P. Soler o Alfredo Soler? La decisión es arbitraria, pero la solución más sencilla sería ignorar los caracteres no alfabéticos comprendidos antes del último espacio y hacer que los nombres equivalgan a AJP Soler y ALFREDO SOLER. Al proceder de esta manera, tanto ALFREDO como AJP se considerarían como nombres de pila, y AJP iría primero.

Parte de nuestro programa aceptaría como entrada un nombre y produciría como salida un nombre, una dirección y un número de teléfono (observe que aún no hemos ni siquiera comenzado a considerar los significados de "dirección" y "número de teléfono"). Si aceptáramos como entrada nombres de formato "libre", con conversión interna a un formato estandarizado, ¿esperaríamos que la salida estuviera en la forma "estandarizada" o en la misma forma que la entrada original? La salida más cómoda para el usuario sería aquella en la que el nombre estuviera en la forma original, pero, como veremos, esto complicará la programación.

Como tarea inicial de programación, supongamos que se le ha asignado un nombre a la variable alfanumérica NOMBRE$ (nombre completo) y que tenemos otras dos variables, NOMB$ (nombre de pila) y APELL$ (apellido). ¿Cómo le asignaremos a NOMB$ y a APELL$ las partes apropiadas de NOMBRES? Ignorando, de momento, el problema de llevar un registro de la forma original en que se ha dado entrada al nombre (de modo que se pueda recuperar cuando la necesitemos más adelante), una sentencia simple del programa podría ser:

Convertir en mayúsculas todos los caracteres
Eliminar todos los caracteres no alfabéticos excepto el espacio final
Asignar a APELL$ todos los caracteres que siguen al último espacio
Asignar a NOMB$ todos los caracteres que preceden al último espacio

Antes de considerar cómo se podría codificar este problema en BASIC, veremos cómo el proceso de "programación de arriba abajo" nos puede llevar desde una enunciación muy amplia de nuestro objetivo hasta el punto en el que se hace posible la codificación en un lenguaje de programación determinado. Observará que estamos utilizando no sólo nombres de variables largos como NOMBRES, sino palabras-órdenes como BEGIN, LOOP y ENDLOOP. Éstas son construcciones que hemos inventado para que nos ayuden a describir nuestro programa. En la etapa final de desarrollo serán reemplazadas por órdenes equivalentes en BASIC. Explicaremos mejor estas órdenes, así como por qué hemos sangrado algunas de las líneas, en el próximo capítulo de nuestra obra.

1.ª ENUNCIACION DE OBJETIVOS

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

1.er REFINAMIENTO

1. Leer NOMBRE$
2. Convertir en mayúsculas todas las letras
3. Hallar último espacio
4. Leer APELL$
5. Leer NOMB$
6. Eliminar caracteres no alfabéticos de NOMB$

2.º REFINAMIENTO

```
1. Leer NOMBRE$
2. (Convertir en mayúsculas todas las letras)
   BEGIN
   LOOP mientras los caracteres no explorados
   permanecen en NOMBRE$
      Leer los caracteres de NOMBRE$ sucesivamente
      IF el carácter está en minúscula
         THEN convertir en mayúscula
         ELSE no hacer nada
      ENDIF
      Asignar el carácter a la variable alfanumérica
      transitoria
   ENDLOOP
   LET NOMBRE$ = variable alfanumérica transitoria
   END
3. (Hallar último espacio)
   BEGIN
   LOOP mientras los caracteres no explorados
   permanecen en NOMBRE$
      IF carácter = " "
         THEN anotar posición en una variable
         ELSE no hacer nada
      ENDIF
   ENDLOOP
   END
4. (Leer APELL$)
   BEGIN
   Asignar a APELL$ los caracteres a la derecha del último
   espacio de NOMBRE$
   END
5. (Leer NOMB$)
   BEGIN
   LOOP mientras caracteres no explorados permanecen en
   NOMBRE$ hasta último espacio
      SCAN caracteres
      IF el carácter no es una letra del alfabeto
         THEN no hacer nada
         ELSE asignar el carácter a NOMB$
      ENDIF
   ENDLOOP
   END
6. (Eliminar caracteres no alfabéticos de NOMB$)
   (Esto se ha manipulado arriba, en 5)
```

Este segundo nivel de refinamiento está ahora muy cerca de la etapa donde será codificado en un lenguaje de programación. Desarrollemos ahora 2 (es decir, convertir en mayúsculas todas las letras) en un tercer nivel de refinamiento y codifiquémoslo a continuación en lenguaje BASIC. Anteriormente ya habíamos encontrado un algoritmo que nos permitía realizar este proceso.

3.er REFINAMIENTO

```
2. (Convertir en mayúsculas todas las letras)
   BEGIN
   READ NOMBRE$
   LOOP
   FOR L = 1 TO longitud de la variable
      READ carácter L
      IF carácter está en minúscula
         THEN restarle 32 al valor del
            carácter en ASCII
         ELSE no hacer nada
      ENDIF
      LET VARTRAN$ = VARTRAN$ + carácter
   ENDLOOP
   LET NOMBRE$ = VARTRAN$
   END
```

Este fragmento de programa en seudolenguaje ahora ya se parece bastante a un lenguaje de programación como para ser codificado. Nuestra versión de BASIC Microsoft no permite que los nombres de variables alfanuméricas sean palabras completas, de modo que en su lugar usaremos letras. Por lo tanto, NOMBRE$ se convierte en N$.

```
1000 REM SUBRUTINA DE CONVERSION A MAYUSCULAS
1010 INPUT "DE ENTRADA A NOMBRE";N$: REM SOLO
     PARA PROBAR
1020 LET P$ = " ": REM ASEGURAR QUE LA VARIABLE
     ESTA VACIA
1030 FOR L = 1 TO LEN(N$): REM INDICE DE BUCLE
1040 LET T$ = MID$(N$,L,1): REM EXTRAER
     CARACTER
1050 LET T = ASC(T$): REM HALLAR VALOR ASCII DE
     CARACTER
1060 IF T > = 97 THEN LET T = T - 32: REM
     CONVERSION A MAYUSCULAS
1070 LET T$ = CHR$(T)
1080 LET P$ = P$ + T$: REM P$ ES LA VARIABLE
     TRANSITORIA
1090 NEXT L: REM FIN BUCLE
1100 LET N$ = P$: REM AHORA N$ ESTA TODA EN
     MAYUSCULAS
2000 PRINT N$: REM SOLO PARA PROBAR
2010 END: REM SOLO PARA PROBAR. REEMPLAZAR
2020 REM 'RETURN' EN PROGRAMA VERDADERO
```

Este fragmento de programa se podría usar como una subrutina dentro de un programa principal. Lo hemos escrito para probar con una sentencia INPUT, otra PRINT, y varias REM y un END. No obstante, estas dos últimas habrían de eliminarse antes de que la subrutina se incorporara a un programa completo.

Complementos al BASIC

LYNX

El Lynx posee una función, UPC$(A$), que convierte en mayúsculas las letras de A$, de manera que el programa se reduce así:

```
1010 INPUT "dé entrada a un nombre";N$
1020 LET N$ = UPC$(N$)
2000 PRINT N$
2010 REM "RETURN" AQUI EN EL
     PROGRAMA VERDADERO
```

SPECTRUM

El programa completo para el Spectrum es:

```
1010 INPUT "DE ENTRADA A UN
     NOMBRE";N$
1020 LET P$ = " "
1030 FOR L = 1 TO LEN N$
1040 LET T$ = N$(L)
1050 LET T = CODE T$
1060 IF T > = 97 THEN LET T =
     = T - 32
1070 LET T$ = CHR$ T
1080 LET P$ = P$ + T$
1090 NEXT L
2000 LET N$ = P$
2010 PRINT N$
2020 REM "RETURN" AQUI EN EL
     PROGRAMA VERDADERO
```

DRAGON 32

Este programa se puede ejecutar en el Dragon, pero los caracteres en minúsculas están reservados para la impresora, de modo que el problema no se plantea realmente

BBC Micro

Reemplazar la línea 1010 por:

```
1010 INPUT "dé entrada a un nombre",
     N$
```

Ejercicios

- Refinar todas las etapas anteriores hasta el punto en que se pudieran convertir en un programa en BASIC. El "seudolenguaje" que emplee no tiene por qué ser igual al nuestro, pero es una medida atinada continuar utilizando letras mayúsculas para los términos clave que probablemente corresponderán a palabras-sentencia en el lenguaje final (por ejemplo, LOOP, IF, LET, etc.). Emplee letras pequeñas para las operaciones que habrán de enunciarse más explícitamente cuando se codifiquen finalmente. Éstas las puede escribir en lenguaje corriente.
- Luego de haber desarrollado los programas hasta un nivel de refinamiento satisfactorio, conviértalos en módulos de programa (subrutinas) en BASIC. Verifíquelos uno a uno empleando entradas ficticias e imprima sentencias que, si funcionan adecuadamente, se puedan eliminar luego.

Ejercicios de revisión

En estos ejercicios se aplican la mayor parte de las sentencias y funciones en BASIC que se utilizan más comúnmente. No hay ninguna pregunta capciosa ni se introduce ningún concepto nuevo. Si puede realizar todos estos ejercicios o la mayoría de ellos sin ninguna dificultad, puede usted considerar que está ya en camino de convertirse en un programador en BASIC hecho y derecho.

- Escriba un programa para aceptar una entrada de dos números desde el teclado, para sumarlos e imprimir el resultado.

- Asigne dos palabras (series de caracteres) a dos variables alfanuméricas y luego cree una tercera variable alfanumérica que concatene (es decir, que una entre sí) las dos palabras originales. Imprima a continuación la tercera variable.

- Escriba un programa que le permita digitar cualquier palabra en el teclado y que después imprima la longitud de la variable en el mensaje LA PALABRA QUE HA DIGITADO TIENE * CARACTERES (* representa un número).

- Escriba un programa que acepte un único carácter digitado en el teclado y que le diga luego cuál es el valor ASCII de ese carácter (en decimal).

- Escriba un programa que le ofrezca el mensaje DIGITE UNA PALABRA y que después le responda con el mensaje LA ULTIMA LETRA DE LA PALABRA ERA * (* representa una letra).

- Escriba un programa que le solicite que DIGITE EL NOMBRE Y PRIMER APELLIDO DE UNA PERSONA y que después le responda EL ESPACIO ERA EL *º CARACTER.

- ¿Cómo modificaría el programa anterior para que imprimiera 3er en vez de 3º si el espacio estuviera en la tercera posición?

- Escriba un programa que le solicite que DIGITE UNA ORACION y que después le responda con el mensaje LA ORACIÓN QUE DIGITO TENIA * PALABRAS (dando por sentado que en la oración habrá una palabra más que el número de espacios).

- Controle su ordenador para ver si se le han asignado caracteres (o gráficos especiales) a los valores de ASCII desde 128 hasta 255 (utilice un bucle y la función CHR$(X)).

Términos clave

La sala de espera

Los ordenadores transfieren la información a mayor velocidad que la que pueden manipular los dispositivos mecánicos. Ello se obvia con una memoria intermedia, el "buffer" o tampón

Los *buffers* (topes) que utilizan los trenes están diseñados para amortiguar el impacto, absorbiendo energía en muelles o pistones amortiguadores. Los ordenadores también poseen buffers y de alguna forma funcionan como aquellos topes, pues ayudan a "permanecer juntos" a los componentes del sistema del ordenador.

En el mundo de la informática este término se aplica con cierta vaguedad y se lo emplea en dos sentidos bastante diferentes. Para el programador, buffer significa una utilización especializada de memoria del ordenador, mientras que para el diseñador de circuitos significa una clase de amplificador de señales eléctricas. Los del segundo tipo, que denominaremos *buffers de señal*, son los de las tablas de conexiones.

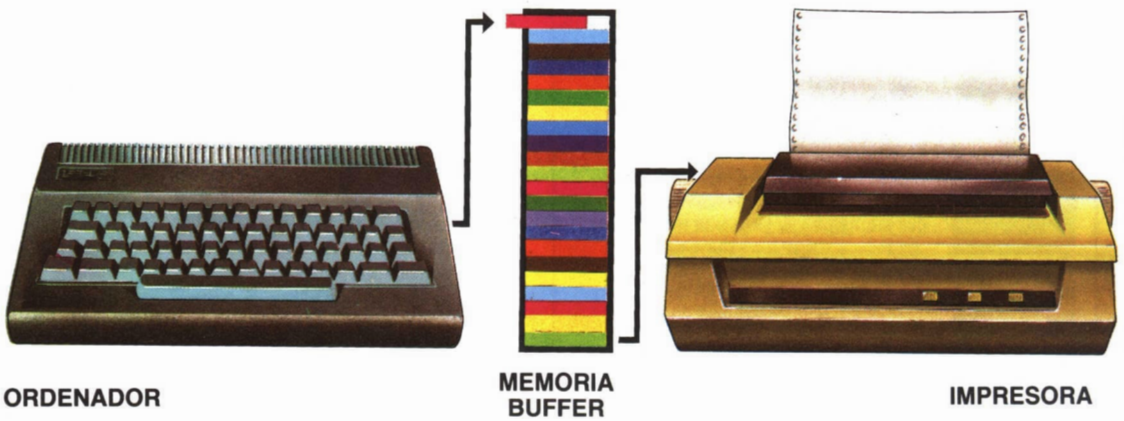
Buffers de memoria

Pensemos en un programa para tratamiento de textos que, entre otras cosas, puede desplazar un bloque de texto desde una parte de un "documento" en la memoria del ordenador hasta otra. El texto se compone de caracteres imprimibles y espacios, y de ciertos caracteres "no imprimibles", como el Carriage Return (retorno de carro). En la memoria del ordenador todos ellos están representados como códigos ASCII en binario. Para cada carácter se requiere un byte de memoria. Llevar los caracteres del bloque desde su antigua posición en la memoria hasta una nueva implica que se debe apartar otra parte de la memoria del ordenador para que sirva de zona de almacenamiento temporal de texto. Dicha zona de memoria apartada para una tarea específica se denomina *buffer*, o tampón.

Como segundo ejemplo, consideremos el problema de imprimir un documento creado con un procesador de textos. El documento podría constar de 15 000 caracteres separados, pero es evidente que no se le podrían enviar todos simultáneamente a la impresora para que los imprimiera: la velocidad de impresión de la mayoría de estos dispositivos no suele superar la cifra aproximada de 80 caracteres por segundo. Para orillar este inconveniente, se apartará parte de la memoria del ordenador, que cumplirá la función de *buffer de impresión*, bajo el control del software de tratamiento de textos. Este programa llenará primero este buffer con los caracteres a imprimir, y luego los irá enviando para su impresión a una velocidad adecuada para la impresora.

El buffer de impresión puede no ser muy grande, tal vez de una capacidad de sólo 128 a 256 bytes, pero, independientemente de sus dimensiones, los principios que lo rigen son los mismos. Primero se escribe en él un "bloque" de caracteres ASCII y después éstos se vuelven a enviar de uno en uno. El primer byte que se escriba en el buffer será, asimismo, el primer byte que se lea de él (como es lógico, deseamos que los caracteres se impriman en el mismo orden en que fueron digitados). Esta clase de buffer se conoce como buffer FIFO (*First In First Out*: primero en entrar, primero en salir). Una vez leídos todos los caracteres del buffer, el software lo llena con el siguiente bloque de caracteres destinados a la impresora.

Los buffers FIFO son muy empleados en la mayor parte de este tipo de software. Se utilizan cuando existe incompatibilidad de velocidades, no sólo entre ordenadores e impresoras, sino también entre ordenadores y unidades de disco flexible y entre ordenadores y teclados de ordenador. Aunque la asombrosa velocidad de procesamiento de los ordenadores significa que por lo general éstos pueden identificar las teclas digitadas en un tiempo menor que el empleado en pulsarlas, es posible que en algunas ocasiones el ordenador no pueda identificar las teclas y visualizar los caracteres correspondientes con la suficiente rapidez. Esto puede suceder si el ordenador está momentáneamente ocupado cumpliendo otra función (accediendo a un disco, por ejemplo). Cuando esto ocurre, lo común es incorporar en el sistema operativo del ordenador un *buffer de digitación adelantada*. Este buffer "recuerda" cuáles teclas se han pulsado y el ordenador las visualiza de inmediato. Lo más probable es que esta acción pase inadvertida al usuario, pero con ciertos sistemas ope-



Parada temporal
Uno de los usos más comunes del buffer es aquél entre el ordenador y una impresora, dado que ésta no puede dar salida a los caracteres a la misma velocidad en que el ordenador los envía. En consecuencia, los caracteres se almacenan en la memoria temporal hasta que este buffer esté completo, y entonces se le envía al ordenador una señal de "ocupado" para que éste deje de transmitir. Los contenidos de la memoria buffer se envían luego a la impresora por el mismo orden en que se recibieron, pero a una velocidad muy inferior. Cuando esta tarea ha terminado, el proceso vuelve a comenzar hasta que se haya impreso el texto completo

Controlando el flujo

Si echamos agua con un cubo dentro de un tanque, del grifo de salida saldrá un chorro uniforme. Del mismo modo, el ordenador "vierte" datos en el buffer, el cual los libera a una velocidad muy inferior a la que funciona, por ejemplo, la impresora

rativos de disco, o con ciertos tipos de software aplicativo (que implican mucho procesamiento de información), podría producirse un ligero intervalo entre la pulsación de la tecla y su aparición en pantalla. Unos pocos sistemas operativos permiten encender o apagar el buffer de digitación adelantada, o incluso que el usuario altere las dimensiones del buffer.

La forma exacta en que está organizado el software para manipular los buffers varía según la función que hayan de cumplir los mismos, pero por lo general será necesario apartar unos pocos bytes para utilizarlos como contadores y banderas. Es indispensable saber cuántos bytes se han leído de un buffer lleno antes de escribir otros; de lo contrario, se podrían destruir datos importantes antes de que se los utilizara.

Buffers de memoria en hardware

Los lectores que posean impresoras pueden haber notado lo lentas que éstas parecen ser, especialmente al imprimir un listado de programa o un documento largo. La mayoría de los sistemas operativos de ordenador no pueden realizar ninguna otra labor mientras se está utilizando la impresora; por lo tanto, si la impresión requiere mucho tiempo, el usuario no tendrá otra opción que permanecer sentado frente a la pantalla a la espera de que la impresora finalice su cometido. Ahora muchos fabricantes ofrecen buffers de impresión accesorios, por lo general en forma de una caja que se conecta entre el ordenador y la impresora. En efecto, desde el punto de vista del ordenador, estas cajas consiguen que la impresora trabaje más aceleradamente. En realidad ésta no imprime con mayor rapidez, sino que esta mayor "velocidad" se debe a la memoria extra exclusiva (a veces hasta de 16 Kbytes), con su propio software incorporado, que poseen estas cajas. Cuando el ordenador ha de imprimir un archivo, le traspasa bytes a la impresora hasta que recibe una señal de "ocupado", que significa que ésta ya no puede aceptar ningún byte más. El ordenador, entonces, ha de esperar hasta que la línea "ocupada" se convierta en "falsa", lo que indicará que la impresora está nuevamente en condiciones de aceptar datos. Si bien por lo general las impresoras cuentan con un pequeño buffer de memoria incorporado, éste no suele poseer más de dos Kbytes y no permite que el ordenador envíe más datos hasta encontrarse vacío. Los accesorios de buffers de memoria en hardware contienen más memoria, de modo que pueden aceptar muchos más datos antes de enviarle al ordenador la señal de "ocupado". Si el buffer es lo suficientemente grande, puede tener capacidad para retener de una sola vez todos los datos a imprimir, con lo cual el ordenador podrá continuar con otras tareas mientras el buffer le envía los datos a la impresora a menor velocidad.

Con frecuencia la memoria se utiliza más bien como un gran soporte de archivo o bus para almacenar programas y datos, pero en lugar de ello se puede organizar en "pilas" o buffers. Las "pilas" son estructuras LIFO (*Last In First Out*: último en entrar, primero en salir), mientras que los buffers, como acabamos de ver, son estructuras FIFO (*First In First Out*: primero en entrar, primero en salir). Para las pilas se suele emplear la analogía del montón de platos apoyados sobre un resorte que existe a menudo en los mostradores de los self-sèrvices. Los platos están apilados en el rimero y el último que se coloque en él será el primero que se quite. Al igual que los buffers, las "pilas" también son zonas de memoria temporal y sólo se diferencian de los buffers en el orden en que se da entrada y se recupera la información. En los lenguajes de alto nivel (como en los intérpretes de BASIC, por ejemplo), las "pilas" se utilizan "internamente", pues se necesita almacenar temporalmente la información para ser cargada luego. Consideremos este fragmento de programa en BASIC:

```
FOR X = 1 TO 10
PRINT "X = ";X
FOR Y = 1 TO 10
GOSUB SCAN
NEXT Y
PRINT "C$ = ";C$
NEXT X
```

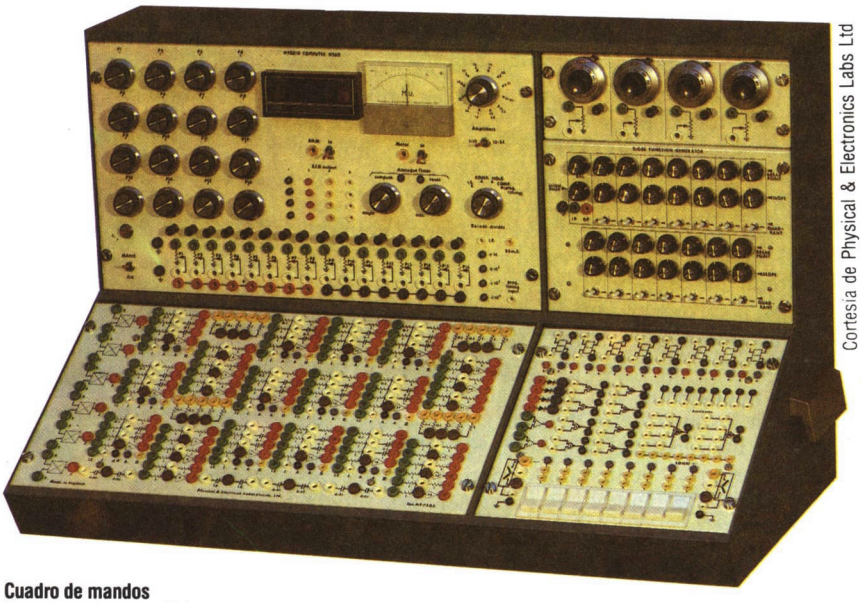
Éste es un ejemplo de bucles FOR...NEXT anidados. Cuando el intérprete de BASIC llega a la segunda sentencia FOR, necesita recordar cuál es la variable utilizada para el FOR anterior (X, en este caso) y por tanto "empuja" la información relativa al primer FOR dentro de una "pila". Cuando se ha completado el bucle interior, hace "saltar" la información desde el extremo superior de la pila y sabe que el FOR actual utiliza la variable X. Dado que los bucles FOR...NEXT se pueden anidar tan profundamente como sea necesario, podría necesitar empujar en la "pila" información para varios FOR. Cuando hace saltar la información de la "pila", obviamente necesita tener la información en orden inverso al orden en que fue empujada.

Por el contrario, los buffers organizan la memoria de modo que la primera información que entra es la primera información que sale. Los buffers se suelen utilizar para rutinas de entrada/salida y se emplean como "interfaces" entre rutinas o dispositivos que trabajan en unidades diferentes o a distintas velocidades. Por ejemplo, una rutina de entrada en BASIC podría trabajar en unidades de líneas, terminadas por un Carriage Return <CR>, pero el intérprete podría funcionar en las líneas por unidades de un carácter. Por lo general los buffers necesitan de un "señalador" que indique en qué lugar del buffer se ha de escribir el siguiente carácter. El señalador serían uno o varios bytes que contuvieran la dirección de aquel carácter. La dirección se incrementaría después de que se hubiera almacenado cada carácter.

Señales poderosas
La lógica interna del ordenador funciona a niveles TTL. Transistor-Transistor-Logic significa un 1 binario con cinco voltios y un 0 con cero voltios. No obstante, aunque dispositivos como la CPU pueden producir estos voltajes, no pueden generar corriente suficiente para activar todos los otros chips que podrían estar conectados a cada patilla. Por tanto, se conectan buffers de señal a las líneas de salida de la CPU para aumentar la cantidad de corriente. Los buffers de señal son pequeños chips, y cada uno de ellos actúa como un buffer para seis señales

Cálculo analógico

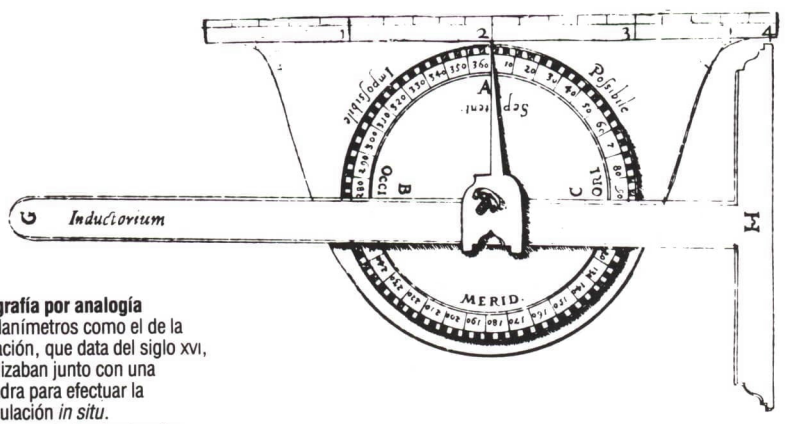
Los ordenadores analógicos, utilizados para controlar máquinas y procesos, reaccionan directamente a los cambios del mundo real, sin que sea preciso traducir la información a forma digital



Cortesía de Physical & Electronics Labs Ltd

Cuadro de mandos
Los ordenadores analógicos no utilizan lenguajes como el BASIC. Se "programan" empalmando diversos componentes eléctricos. Los componentes se fijan a la cara posterior de un tablero de circuitos. En la cara anterior se colocan los enchufes y conectores que permiten conectar entre sí los componentes seleccionados

Existen dos familias de ordenadores bastante diferentes, y hasta ahora sólo nos hemos estado ocupando de una de ellas: el ordenador digital, así llamado porque todas las instrucciones de un programa y todos los datos se representan utilizando "dígitos" binarios. La otra familia, la más antigua de las dos, es la de los ordenadores analógicos, que se programan de forma diferente: uniendo varios componentes electrónicos.



Topografía por analogía
Los planímetros como el de la ilustración, que data del siglo XVI, se utilizaban junto con una escuadra para efectuar la triangulación *in situ*. Anteriormente, los topógrafos tenían que tomar medidas y luego efectuar cálculos matemáticos para determinar la longitud del lado más lejano de su triángulo, pero gracias a este sencillo instrumento analógico su trabajo se volvió mucho más sencillo y exacto. Aún hoy en día se emplean instrumentos de nombre similar para medir la superficie de planos irregulares, como pueden ser las pieles de animales

El velocímetro constituye un ejemplo de este ordenador "analógico", especializado y sencillo; el nombre proviene del comportamiento "análogo" de la velocidad del coche respecto a la posición de la aguja en el dial, siendo aquélla directamente proporcional a ésta. Los ordenadores analógicos modernos pueden realizar muchas tareas y se basan en el tipo de componentes eléctricos que encontramos comúnmente en los aparatos domésticos: transistores, condensadores, resistencias e inductancias magnéticas. Cuando se comenzaba a desarrollar la electrónica, se descubrió que el comportamiento de los componentes eléctricos se parecía al de los dispositivos mecánicos. Por ejemplo, los ingenieros eléctricos descubrieron que las oscilaciones de corriente eléctrica que podían resultar al conectar entre sí una inductancia magnética y un condensador se asemejaban muchísimo a las oscilaciones de una pesa colgada de una cuerda. De hecho, la descripción matemática de ambos sistemas era idéntica: la base para el "cálculo analógico".

Algunos dispositivos analógicos se asemejan mucho a los sistemas que ejemplifican; por ejemplo, el modelo de un aeroplano que se emplea para un experimento de túnel de viento es una copia exacta, a escala reducida, de la armazón del avión. Otros modelos resultan muy distintos. Por ejemplo, un "modelo" de una situación de la vida real puede consistir tan sólo en una lista de fórmulas matemáticas, o puede ser un circuito eléctrico que reproduce el flujo de agua a través de una presa.

Las máquinas de calcular que utilizan principios analógicos demostraron por primera vez su importancia en 1630, cuando William Oughtred inventó la regla de cálculo. Los números estaban dispuestos sobre dos reglas de modo tal (estaban espaciados logarítmicamente) que el movimiento de una a lo largo de la otra equivalía a la multiplicación, y la respuesta simplemente se podía leer de la escala.

A fines del siglo XIX lord Kelvin diseñó un ingenioso dispositivo mecánico a manivela que se podía emplear para calcular las mareas altas y bajas de un puerto durante todo el año.

Luego, en 1930, se construyó en Estados Unidos una máquina electromecánica que podía resolver ecuaciones diferenciales generales (el tipo de ecuaciones que se descubre prácticamente en cualquier representación matemática del mundo real), en vez de la serie específica de ecuaciones diferenciales que Kelvin había conseguido resolver. La máquina la inventó Vannevar Bush, quien le dio la denominación de analizador diferencial.

Los primeros ordenadores analógicos

Los primeros ordenadores analógicos totalmente electrónicos entraron en funcionamiento en 1947, justo después de que nacieran los primeros ordenadores digitales. Hemos visto hasta ahora cómo los ordenadores digitales realizan cálculos aritméticos utilizando una combinación de puertas lógicas. El ordenador analógico puede efectuar cálculos matemáticos simplemente utilizando la naturaleza de la electricidad. Si hay, por ejemplo, una corriente eléctrica de cinco amperios que fluye por un cable y una corriente de cuatro amperios que fluye por otro, y los dos cables están

unidos entre sí para converger en un único cable, la corriente del nuevo cable será de nueve amperios: la suma de las dos corrientes. De manera, por lo tanto, que el conjunto del comportamiento y las propiedades del flujo de corriente constituyen automáticamente un "sumador".

Hasta para las funciones matemáticas muy complicadas a menudo se hallan soluciones muy sencillas en circuitos elementales (un ejemplo es la "integración": hallar la superficie bajo una curva). Los ordenadores analógicos no se "programan" como las máquinas digitales; en cambio, se ha de construir un circuito que ejemplifique el programa a resolver. Todos los componentes disponibles se montan en la parte posterior de un "tablero", en cuya cara anterior están los enchufes y conectores para unir los componentes entre sí mediante cables. El aspecto de este "cuadro de mandos" recuerda mucho a un anticuado cuadro conmutador de teléfonos.

En un ordenador analógico se utilizan los diversos voltajes o corrientes para representar cantidades físicas como fuerza o velocidad, y los "valores" de los componentes eléctricos representan cosas como la masa de un coche o la fuerza de sus amortiguadores. Pero en un ordenador digital todos los datos se representan mediante series de impulsos, cinco voltios para el 1 binario y cero voltios para el 0 binario. Aquí hallamos una distinción más amplia entre los ordenadores analógicos y los digitales: en unos la información se puede almacenar de forma que satisfaga cantidades que cambian continuamente, pero en los otros los datos se almacenan en unidades "discretas" o individuales.

Algunos ordenadores personales (el BBC Micro, por ejemplo) poseen un conector para entrada analógica. En general, cuando los ordenadores digitales manipulan información analógica, como una temperatura o una fuerza, primero han de convertir los datos en forma digital.

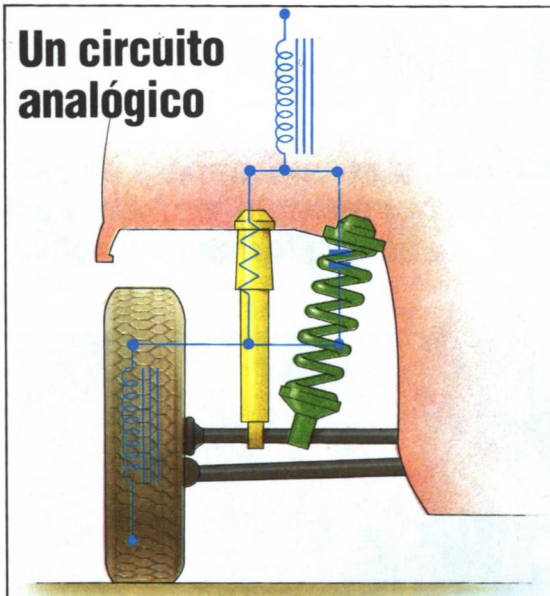
La gran ventaja de los sistemas digitales es que la información se puede procesar o transmitir sin pérdida de calidad. Si un impulso de cinco voltios se hace pasar a través de un circuito eléctrico, podría muy bien ser afectado por la distorsión inherente a todo circuito y salir, quizá, como 4,9 voltios. En un sistema analógico, en el cual las fluctuaciones de voltaje representan cambios en la información, esto podría significar, en la voz de un cantante, la diferencia que existe entre un la y un la bemol. Pero en un sistema digital, en el cual sólo hay dos señales posibles, cinco voltios o cero, cualquier señal próxima a cinco (por ejemplo, 4,9) se reconoce y se reproduce automáticamente como cinco voltios. De modo que los errores se corrigen y no se acumulan. Por el contrario, la acumulación de errores a medida que la señal pasa a través de circuitos sucesivos representa uno de los inconvenientes del ordenador analógico.

No obstante, este tipo de ordenadores saca provecho de representar las cantidades como valores variables de un voltaje o una corriente. Esto significa que una condición de entrada se puede cambiar rápidamente y que el sistema reflejará de inmediato las modificaciones consiguientes. No se necesita tiempo para codificar los datos en impulsos binarios, ni para procesarlos ni para, finalmente, volver a decodificarlos para producir la salida.

Esta característica es muy importante para las aplicaciones en las que resulta esencial producir respuestas rápidas. Por ejemplo, un piloto automático debe responder a una repentina corriente de viento durante un aterrizaje, cuando ya no hay tiempo para efectuar cálculos largos, ni siquiera a la velocidad a que funcionan los ordenadores digitales modernos. Los sensores detectan la corriente repentina, generando un voltaje de salida relativamente pequeño. El circuito del piloto automático responde de manera instantánea con un cambio del voltaje de salida relativamente grande, que de forma automática activa los alerones de las alas, de tal manera que el aparato consiga mantener la estabilidad sin mayores percances.

Los ordenadores analógicos se utilizan en muchas áreas del control industrial, en las cuales se han de manejar con sumo cuidado complicados sistemas mediante ajustes sutiles y continuos, como en una planta industrial química. Pero son menos conocidos que sus equivalentes digitales. Aunque algunas veces se emplean ordenadores analógicos sencillos en las escuelas, con fines pedagógicos, el tipo de aplicaciones para las cuales resultan idóneos revela que es muy poco probable que lleguemos alguna vez a ver un ordenador personal analógico. Los ordenadores analógicos se utilizarán siempre, pero serán los ordenadores digitales los que irán dominando el mercado a medida que vayan siendo más rápidos y más potentes.

Un circuito analógico



Los sistemas de suspensión de coches se componen de un muelle y un amortiguador. El primero absorbe el impacto súbito producido al pasar un bache, y luego el amortiguador elimina las oscilaciones del muelle. Los ingenieros deben seleccionar las dimensiones óptimas del muelle y el amortiguador para lograr la marcha más cómoda posible sobre diferentes superficies de carreteras y a distintas velocidades. Para representar la disposición del muelle, el amortiguador, el chasis y la rueda se puede utilizar un circuito eléctrico. Se aplica un voltaje fluctuante a uno de los extremos para representar los baches de la calzada y se varían las dimensiones de los componentes eléctricos hasta que el voltaje de salida sea lo más uniforme posible

Kevin Jones

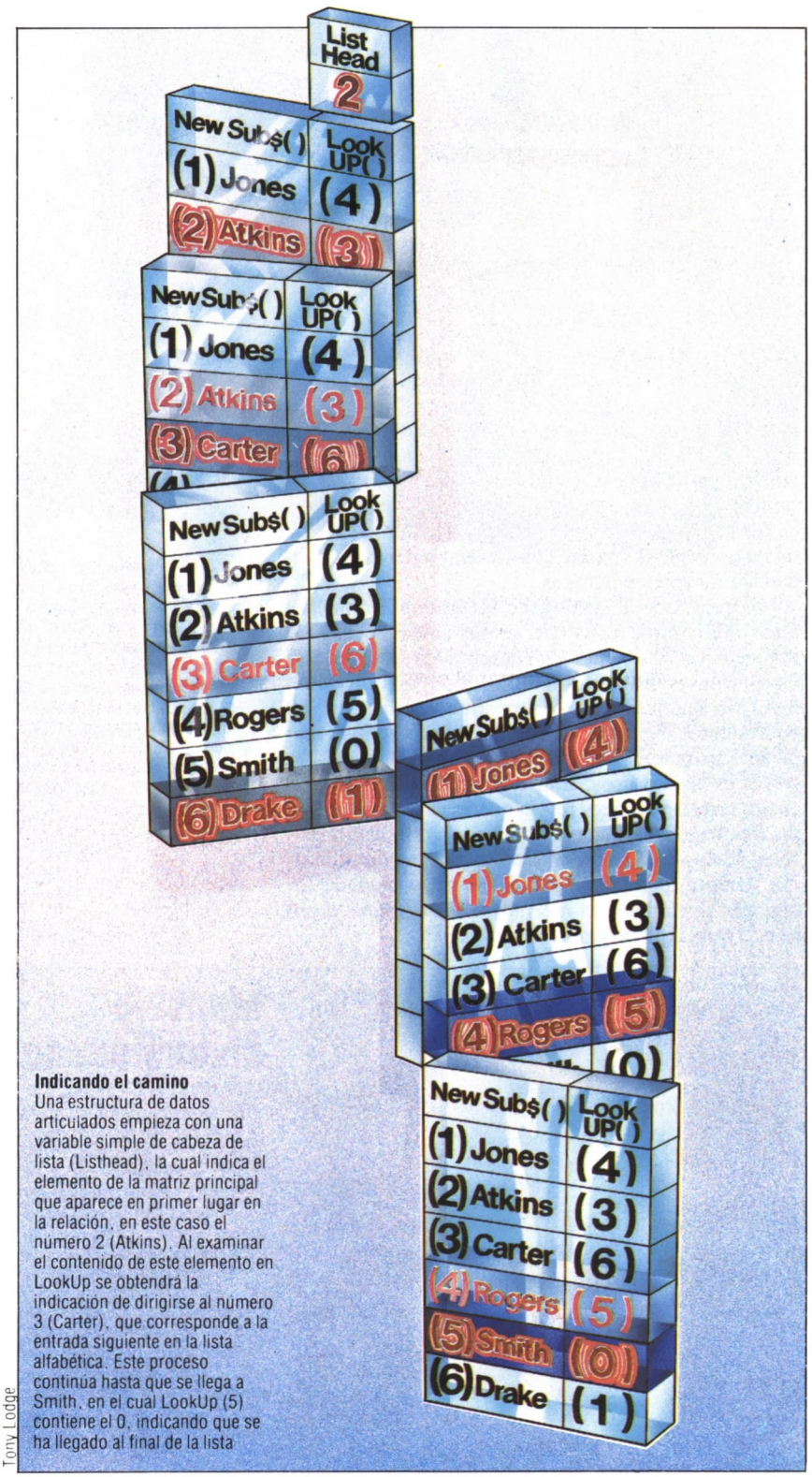
Mensaje distorsionado

La manera en que se va alterando un mensaje cuando se transmite de oído a oído tiene gran similitud con los errores acumulativos de un circuito analógico

En un circuito digital sólo se puede pasar un número limitado de mensajes (concretamente 1 o 0). De modo que si se produjera cualquier distorsión, ésta se podría identificar fácilmente y eliminar en la siguiente etapa

Cota de malla

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



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

Tony Lodge

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

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

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

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

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

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

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

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

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

ListHead (2)

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

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

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

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

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

ListHead (2)

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

La matriz LookUp ( ) sólo cambia en dos lugares:

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

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

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

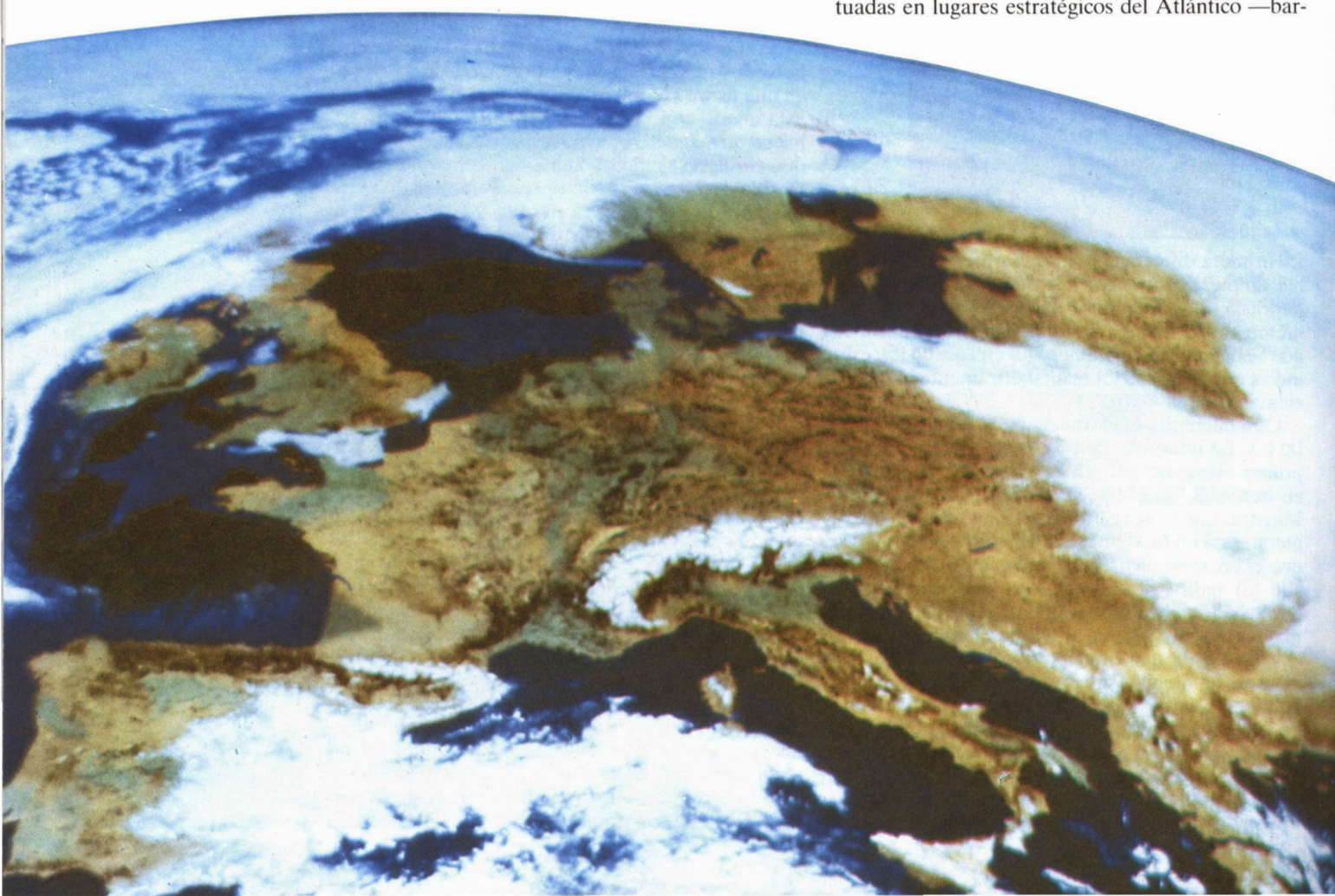
Pronósticos más precisos

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

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

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

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



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

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

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

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

Procesamiento de números

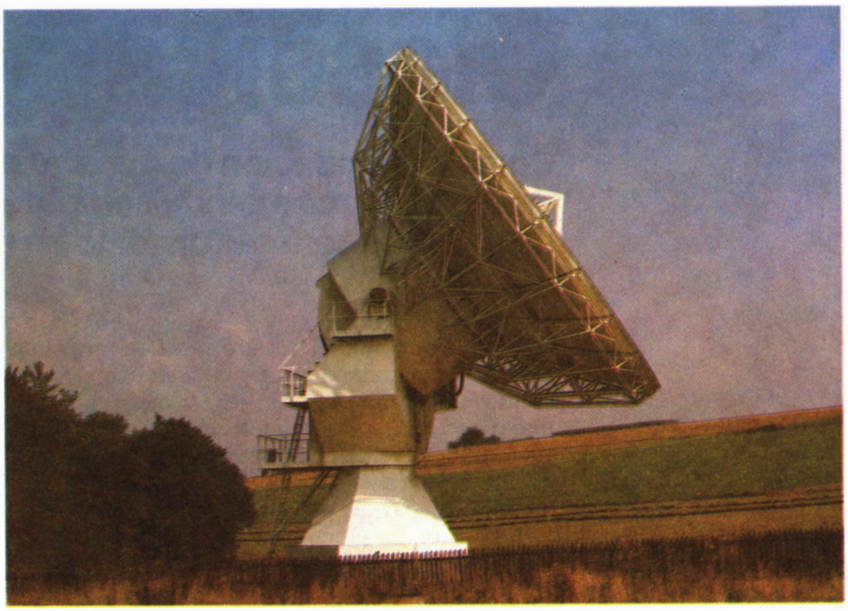
Uno de los principales usos de los grandes ordenadores en la investigación científica consiste en el procesamiento de información exclusivamente numérica en forma de ecuaciones muy extensas y complejas. Las aplicaciones puramente científicas, como por ejemplo de física nuclear, o de ciencia aplicada,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

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

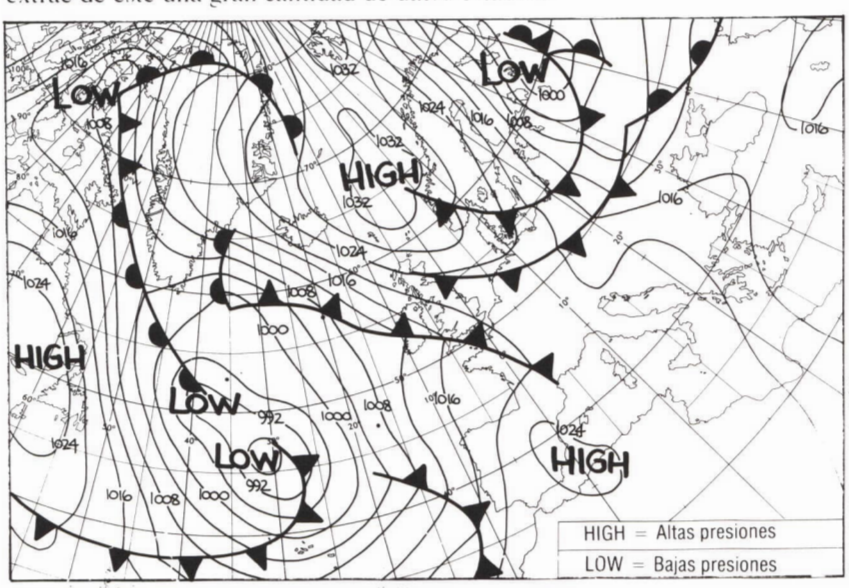
El Meteosat 2, que reemplazó al satélite anterior en junio de 1981, está situado en una órbita geoestacionaria a unos 35 880 km sobre el ecuador, en el meridiano cero. Reúne información que le proporcionan un gran número de estaciones terrestres esparcidas por toda la superficie del globo, que es transmitida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

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

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



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



Cortesía Meteorological Office

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

Clasificar y archivar

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

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

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

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

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

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

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

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

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

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

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

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

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

Y ahora el mismo proceso para la etapa cuarta:

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

3.er REFINAMIENTO

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

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

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

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

2.o REFINAMIENTO

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

3.er REFINAMIENTO

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

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

CODIFICACION

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

En BASIC corriente esto tendría la forma siguiente:

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

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

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

Parte de subrutina X

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

Parte de subrutina Y

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

Ejercicio

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

campo NOMBRE
campo CALLE
campo CIUDAD
campo PROVINCIA
campo NUMERO TELEFONO

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

5. CREAR UNA NUEVA ENTRADA

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

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

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

Complementos al BASIC

SPECTRUM

Paso 3

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

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

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

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

Paso 4

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

Paso 5

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

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

Parte de subrutina X

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

Parte de subrutina Y

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

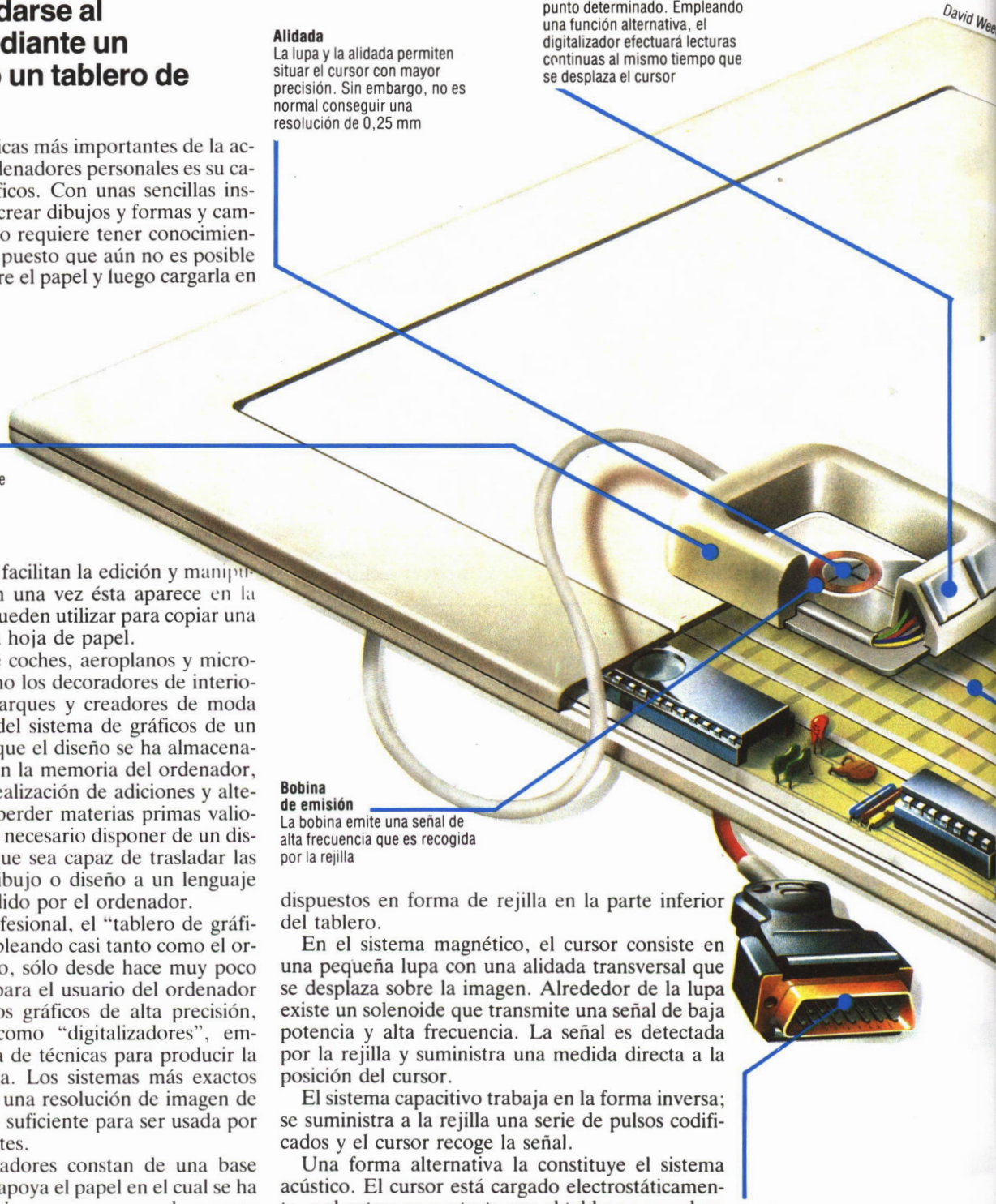
VARIABLES

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

Papel de calco

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

Una de las características más importantes de la actual generación de ordenadores personales es su capacidad de crear gráficos. Con unas sencillas instrucciones se pueden crear dibujos y formas y cambiar colores. Todo ello requiere tener conocimientos de programación, puesto que aún no es posible crear una imagen sobre el papel y luego cargarla en el ordenador.



Los lápices ópticos facilitan la edición y manipulación de una imagen una vez ésta aparece en la pantalla, pero no se pueden utilizar para copiar una figura a partir de una hoja de papel.

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

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

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

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

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

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

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

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

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

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

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

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

Simon Lewis

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

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

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

Términos clave

"Dulces dieciséis"

Los ordenadores trabajan en notación binaria, dado que pueden hacer frente a sólo dos situaciones. ¿Por qué, entonces, algunas veces los programadores utilizan el sistema hexadecimal, que es de base 16?

Resulta sencillo comprender por qué los ordenadores emplean internamente el sistema binario: la representación de los números utilizando sólo ceros y unos se presta bien para las señales eléctricas encedido/apagado del ordenador. Asimismo, resulta sencillo comprender por qué la sociedad humana utiliza casi universalmente el sistema decimal: como tenemos diez dedos, este número se ha convertido en nuestro número de base natural.

Pero ¿para qué habría de servir el sistema hexadecimal (de base 16)? La razón es que convertir números hexadecimales en números binarios y viceversa es mucho más fácil que convertir decimales en binarios y binarios en decimales. Puesto que los ordenadores "comprenden" los números binarios, ¿por qué los informáticos no programan con dígitos binarios? Por dos razones. La primera de ellas es que los números binarios son mucho más largos que sus equivalentes de base 10 o de base 16. Por ejemplo, el número 356 en decimal es 0000000101100100 en binario. La segunda razón es que utilizando números que sólo consten de ceros y unos resulta mucho más fácil equivocarse.

El sistema hexadecimal utiliza 16 dígitos diferentes, incluyendo el cero, de manera que el número mayor que se puede representar en la columna de las "unidades" es el equivalente al decimal 15. Los matemáticos podrían haber inventado seis nuevos símbolos para los números entre 10 y 15, pero la convención exige emplear las seis primeras letras del alfabeto para los seis dígitos extra. Cuando contamos mediante la adición de unos, usamos los dígitos numéricos estándar junto con los dígitos alfabéticos hasta llegar a la F, el equivalente hexadecimal de 15. Después de ello, se nos terminan los símbolos y hemos de comenzar una nueva columna, la de los "dieciséis". Aquí también hemos de utilizar los símbolos disponibles hasta la F. Cuando tanto las columnas de los "dieciséis" como la de las "unidades" estén completas, sumar uno implicará que tengamos que agregar una tercera columna, la de los "doscientos cincuenta y seis". En el panel vemos algunos números decimales con sus equivalentes en binario y en hexadecimal.

En hexadecimal, el número 65 535 se representa FFFF, con lo cual resulta evidente una de las ventajas que ofrece este sistema. Un número cuya representación en decimal exige 5 dígitos y 16 en binario, en notación hexadecimal sólo requiere 4 dígitos. Aparte de esta compactibilidad, ofrece otra ventaja más importante. Cuatro dígitos binarios se pueden representar exactamente con un dígito hexadecimal. Esto hace que la conversión de binario a hexadecimal y viceversa sea relativamente directa.

Para transformar un número binario en su equivalente hexadecimal, divida el número binario en grupos de cuatro dígitos, empezando por la derecha, y convierta a hexadecimal un grupo cada vez. A continuación le ofrecemos algunos ejemplos:

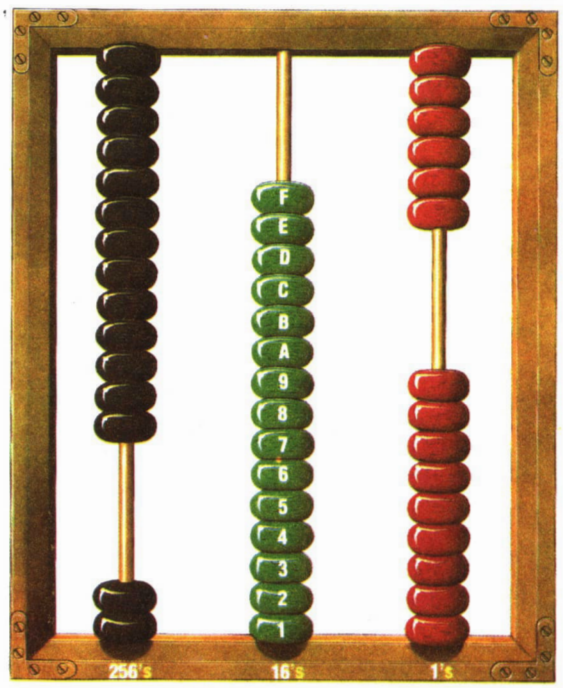
		1110	1001	binario
		=E	9	hex.
		=233		decimal
	1111	1000	1100	binario
	=F	8	C	hex.
	=3980			decimal
0111	1110	0010	1101	binario
=7	E	2	D	hex.
=32 301				decimal

¿Cuándo deseará realmente emplear la notación hexadecimal? A pesar de que los lenguajes de alto nivel, como el BASIC, no le exigen la utilización de hexadecimales (ni, para el caso, de binarios), sí lo exigen lenguajes de bajo nivel como el ensamblador y el código de lenguaje máquina.

Los números escritos en hexadecimal se suelen escribir seguidos de una H para diferenciarlos de los números decimales. Por lo tanto, 256 (en decimal) se escribiría 100H y se leería "uno cero cero hex.".

En esta obra utilizaremos muy poco los números hexadecimales. No obstante, puede que el usuario se encuentre con este término en los apéndices del manual de instrucciones del ordenador. Por eso conviene saber calcular su equivalente en decimal o en binario.

Mark Watkinson



El ábaco hexadecimal
Un ábaco construido para el sistema hexadecimal tendría 15 bolitas en cada varilla. En la varilla situada a la derecha, las bolitas se pueden hacer bajar de una en una hasta llegar a la última, la F. El hexadecimal es un sistema numérico de base 16 y para los números mayores que 9 se utilizan las letras del alfabeto. Como muestra la varilla central, A corresponde al decimal 10, B al 11, C al 12, D al 13, E al 14 y F al 15. Después del "número" F se nos terminan los dígitos y hemos de remitirnos a la columna siguiente, la de los "dieciséis". En la ilustración, la columna de los "1" nos muestra una cuenta de 9 bolitas. La columna de los "16" muestra una cuenta de F y la columna de los "256" una cuenta de 2. Esto se lee "2 F 9 hex.". Es el equivalente al número decimal 761 (256 × 2 + 16 × 15 + 1 × 9)

Binario	Decimal	Hex.
00000001	1	1
00000010	2	2
00000011	3	3
00000100	4	4
00000101	5	5
00000110	6	6
00000111	7	7
00001000	8	8
00001001	9	9
00001010	10	A
00001011	11	B
00001100	12	C
00001101	13	D
00001110	14	E
00001111	15	F
00010000	16	10
00010001	17	11
00010010	18	12
00010011	19	13
00010100	20	14
00010101	21	15

Modelos de comportamiento

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

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

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

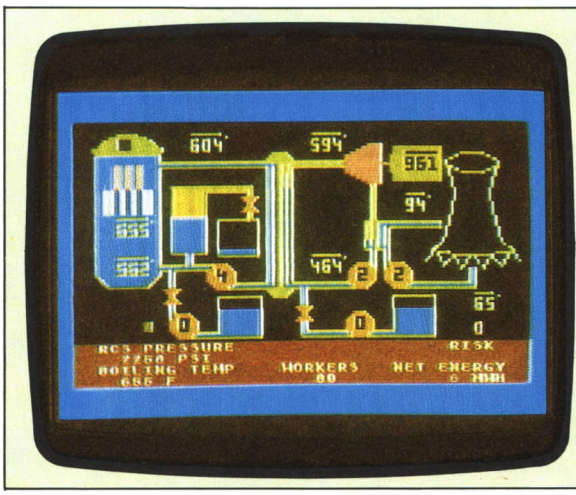
Existen muchas maneras de resolver el problema, pero una de las más sencillas consiste en utilizar un modelo. Por ejemplo, podríamos establecer uno en el que el volumen de vino de cada vaso fuera exactamente de una cucharada. Es fácil comprender que en este caso ambos vasos acabarán con el mismo grado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

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

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

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

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



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

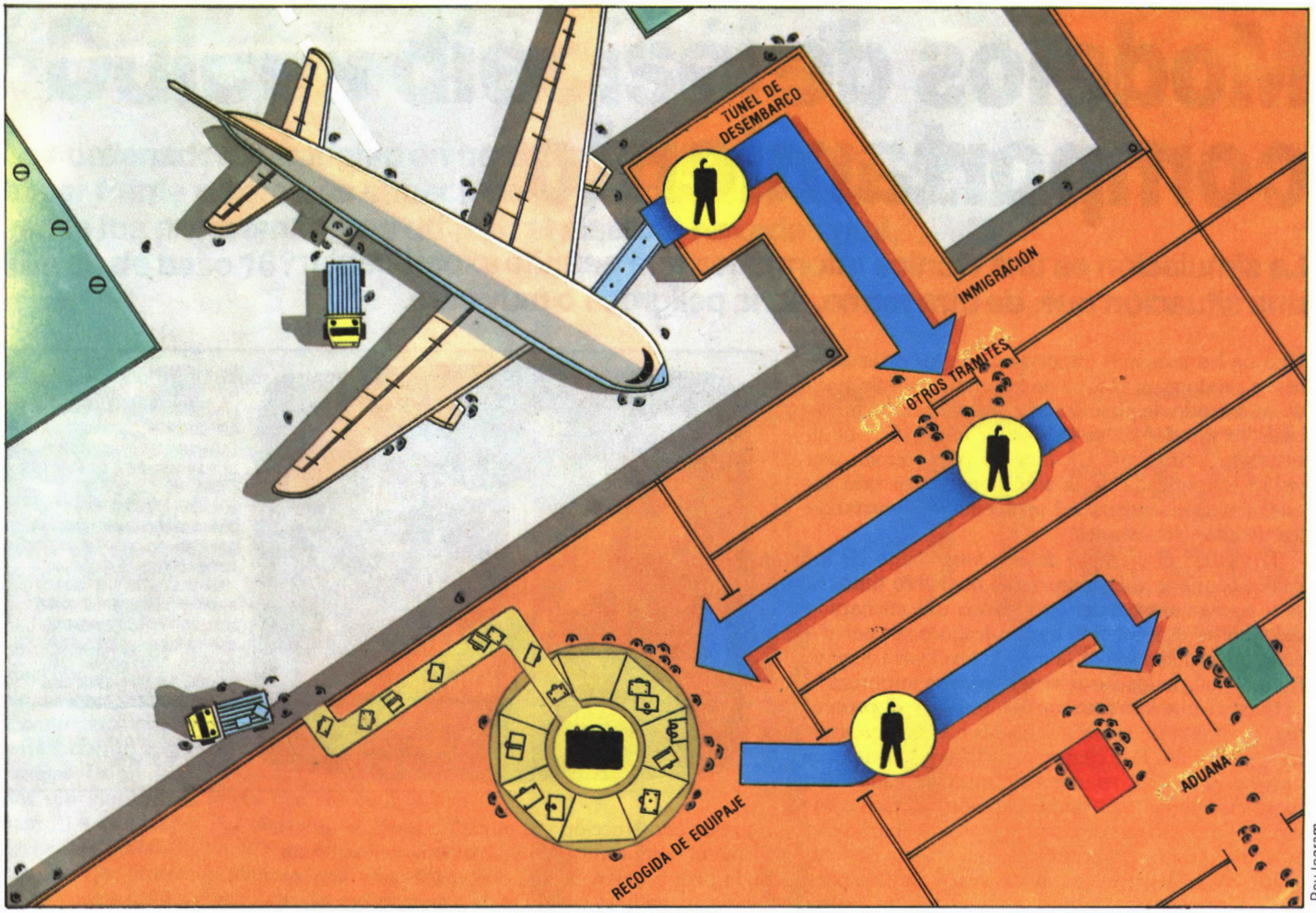
Software, cortesía de Pilot Software
Ian McKinnell

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

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

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

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



Roy Ingram

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

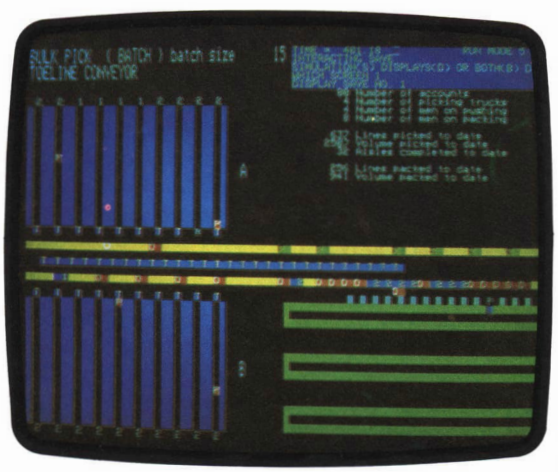
porcionó un modelo matemático mucho más sencillo, con el Sol en el centro. En la actualidad, al observar lejanas galaxias de estrellas, los astrónomos descubren que todas ellas se alejan de nosotros a velocidades que aumentan de continuo, lo que vuelve a sugerir que nosotros nos hallamos en el centro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

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

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

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

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

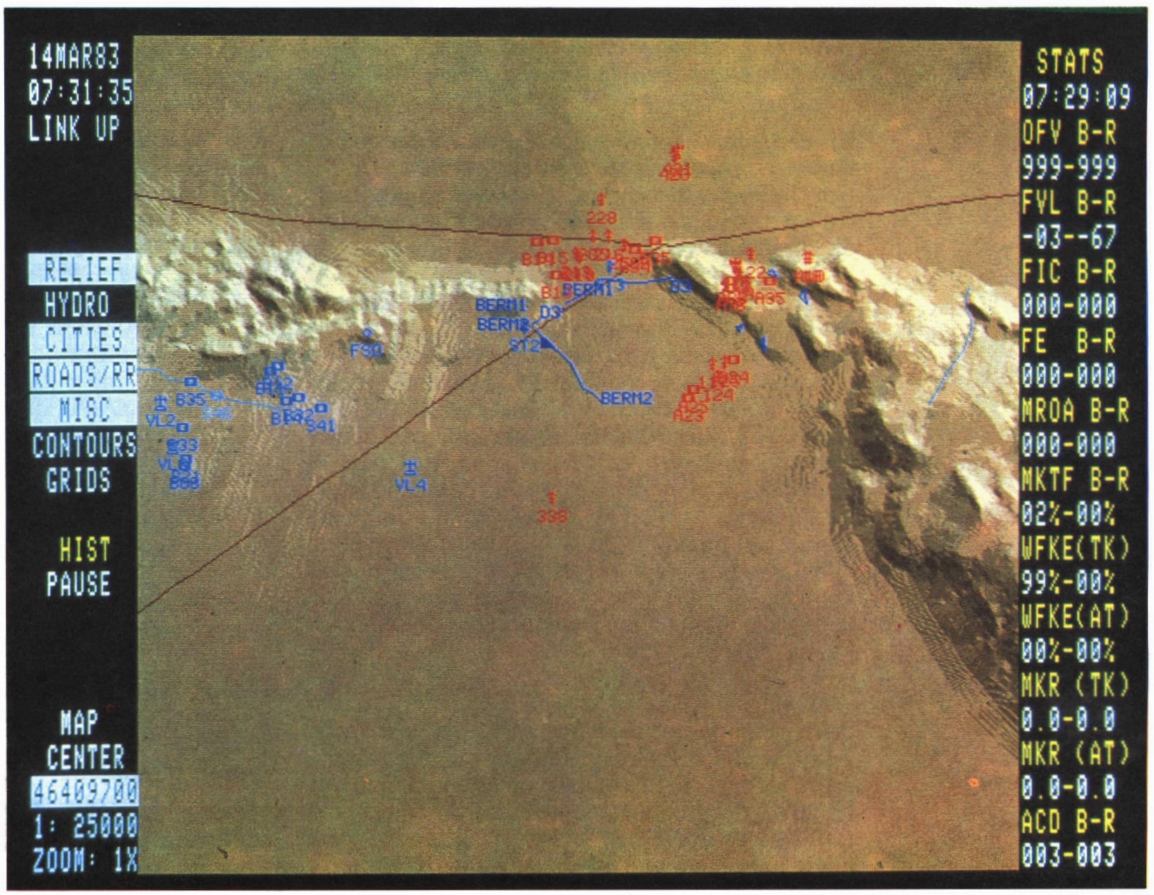


Cortesia de BL Systems

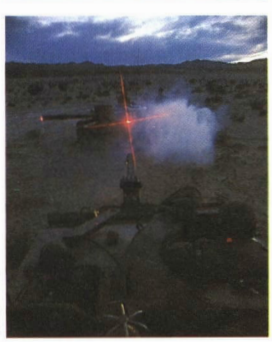
"See why" (Vea por qué)
BL Systems, subsidiaria de la British Leyland, desarrolló un paquete de modelación por microordenador para diseñar sus nuevas factorías y cadenas de producción. El paquete, denominado "See why", desde entonces se ha puesto a la venta en Gran Bretaña. Aunque el mayor énfasis del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

Micros al mando

La aplicación del ordenador en la tecnología militar ha sido un factor decisivo en el desarrollo de la informática



En el campo de batalla
El ejército norteamericano en un simulacro en el desierto de Mojave. Las ametralladoras y otras armas están conectadas con sistemas láser y sus objetivos tienen dispositivos sensores. Cuando el rayo láser golpea un sensor, el ordenador graba la precisión del impacto



Brian Wolff

En la actualidad tal vez las fuerzas armadas sean las "industrias" más computerizadas del mundo. El soldado moderno, además de estar bien entrenado para el combate tradicional y para desarrollar una gran capacidad de resistencia, debe ser asimismo capaz de operar la tecnología que emplean las armas modernas.

La guerra de las Malvinas demostró el poder devastador que posee la nueva generación de armas por microchip. El misil Exocet que hundió al *Sheffield* británico estaba guiado por ordenador y el piloto ni siquiera necesitó mirar el objetivo y apuntarle en la forma convencional.

Los ordenadores están demostrando ser enormemente útiles para los tanques, en los que la precisión y la velocidad son factores esenciales. Los carros de combate Chieftain, del Ejército británico, están equipados con un ordenador que controla factores de importancia primordial, como son la dirección del viento, la posición del tubo del cañón, la clase de munición que se está utilizando y el ángulo de tiro. A partir de esta información calcula la ruta correcta y apunta hacia el blanco con precisión. Un ordenador de este tipo le permite al tanque acertar nueve impactos de nueve disparos, en un lapso de 53 segundos y con un alcance de 1,2 km.

Los ejércitos de la OTAN están utilizando un ordenador similar. Se trata de un sistema belga que emplea un telémetro láser, sensores, un ordenador y un visor óptico. Los sensores miden factores como el viento y la posición del cañón, y el ordenador calcula los ángulos y luego expone en la mira una serie de hilos del retículo. Como los hilos siguen al blanco, lo mismo hace el cañón.

Los misiles Cruise emplean diversos ordenadores para apuntar su cabeza explosiva a una distancia de 18 m del blanco, después de un vuelo de aproximadamente 3 200 km. Cuando se lanza el misil, su vuelo es guiado por un ordenador que almacena toda la información relativa a la ruta seguida. El ordenador realiza de manera continua ligeros ajustes en la trayectoria de vuelo. Cuando el misil se está aproximando al objetivo, el ordenador activa su sistema de guía final e identifica el objetivo a partir de una "imagen" compuesta por millones de números que tiene en su memoria. Una vez los detectores del misil han "visto" el objetivo, le envían al ordenador otra imagen digital. Entonces el ordenador guía al misil hasta que las dos series de números coinciden exactamente, momento en que lo dirige hacia el blanco.

El desarrollo de los ordenadores para su aplicación a la tecnología militar ha sido un factor primordial para acelerar el progreso de la ciencia y el diseño de la informática. Sin las enormes sumas de dinero que se invirtieron en los laboratorios militares, es muy probable que jamás en toda nuestra vida hubiésemos entrado en contacto con un ordenador personal.

Nuevas entradas

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

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

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

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

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

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

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

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

En BASIC, esto tendría un desarrollo sencillo:

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

La búsqueda

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

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

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

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

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

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

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

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

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

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

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

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

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

A esto le podríamos dar la siguiente forma:

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

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

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

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

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

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

Ejercicios

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

Su tarea consiste en:

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

En la página 157 le ofrecíamos una serie de ejercicios de revisión para ayudarlo a valorar su nivel de aprovechamiento en nuestro curso de programación BASIC. Hallará las soluciones en la página 182.

Complementos al BASIC

SPECTRUM

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

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

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

El segundo programa para el Spectrum es:

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

VARIABLES

Para lo referente a las variaciones de los nombres de las variables, véase "Complementos al BASIC" anterior, p. 169.

STEP 0

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

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

LYNX

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

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

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

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

Cristal líquido

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

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

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

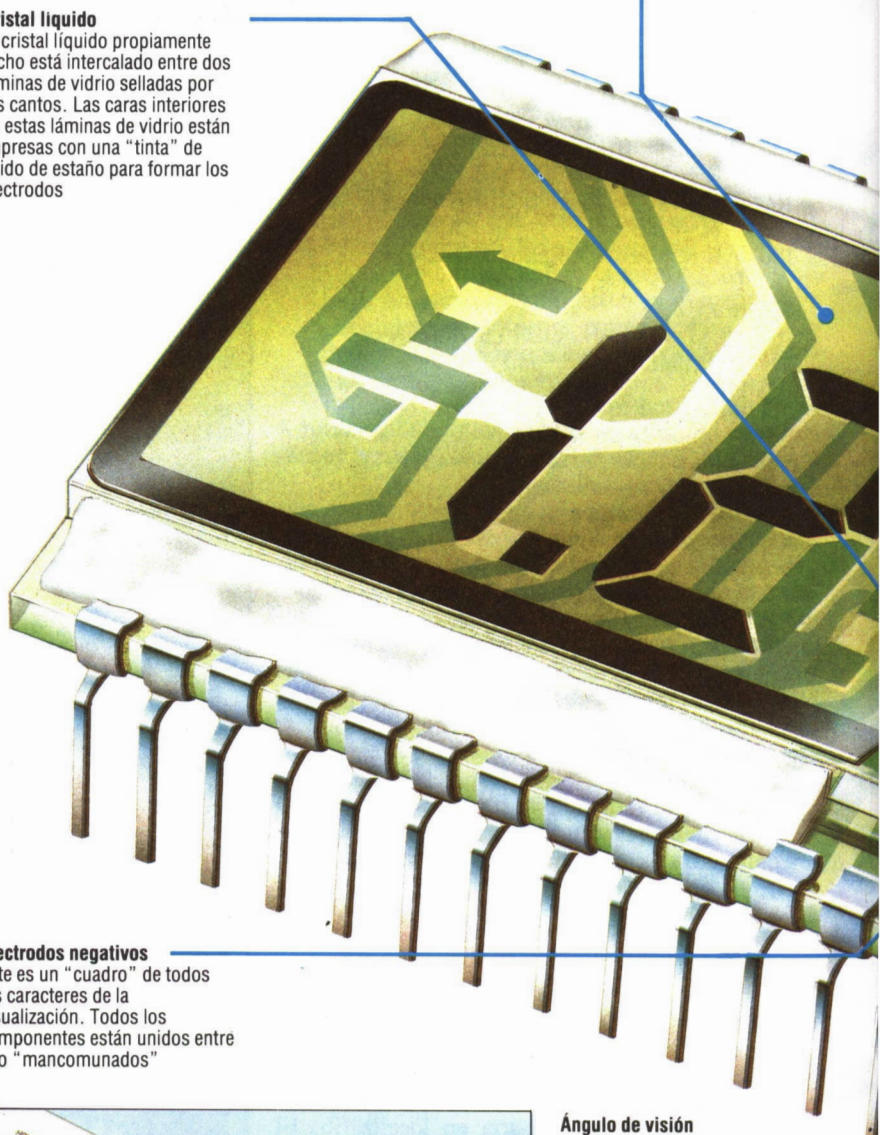
Hasta mediada la década de los setenta, las visualizaciones de las calculadoras y los relojes estaban compuestas por LED (*Light Emitting Diodes*: diodos emisores de luz) en forma de barra, dispuestos de manera que formaban una versión más bien angular de una letra o un número. Pero las visualizaciones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

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

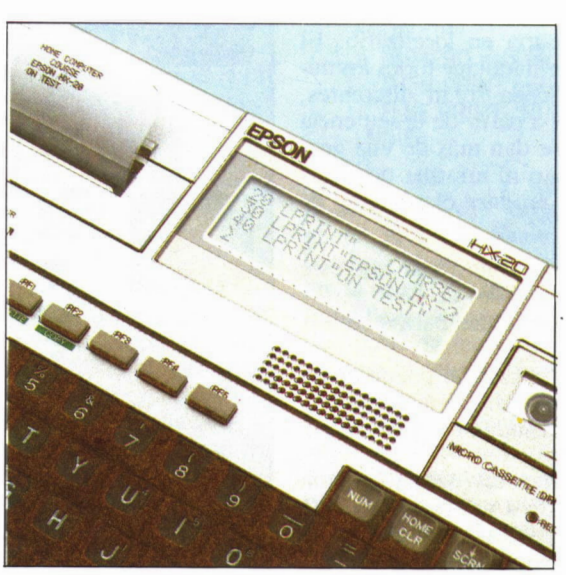
El proceso en virtud del cual se define este carácter requiere que la luz pase a través del primer filtro y que, de este modo, se polarice verticalmente. Luego se desvía 90°, y a causa de este hecho queda concentrada en el filtro posterior. De esta manera, el área del cristal líquido a la cual se le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

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

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

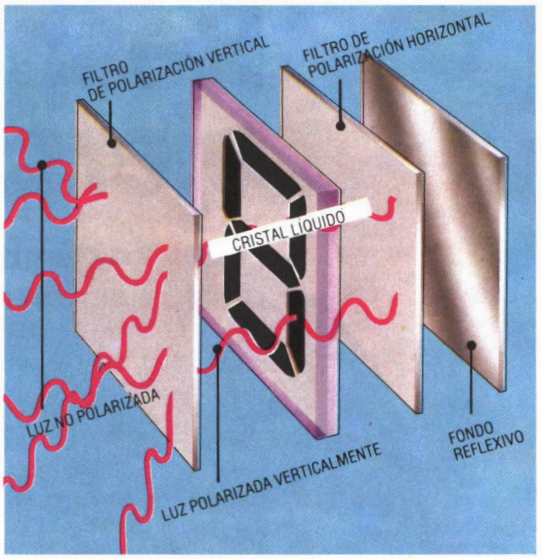


Electrodos negativos
Éste es un "cuadro" de todos los caracteres de la visualización. Todos los componentes están unidos entre sí o "mancomunados"



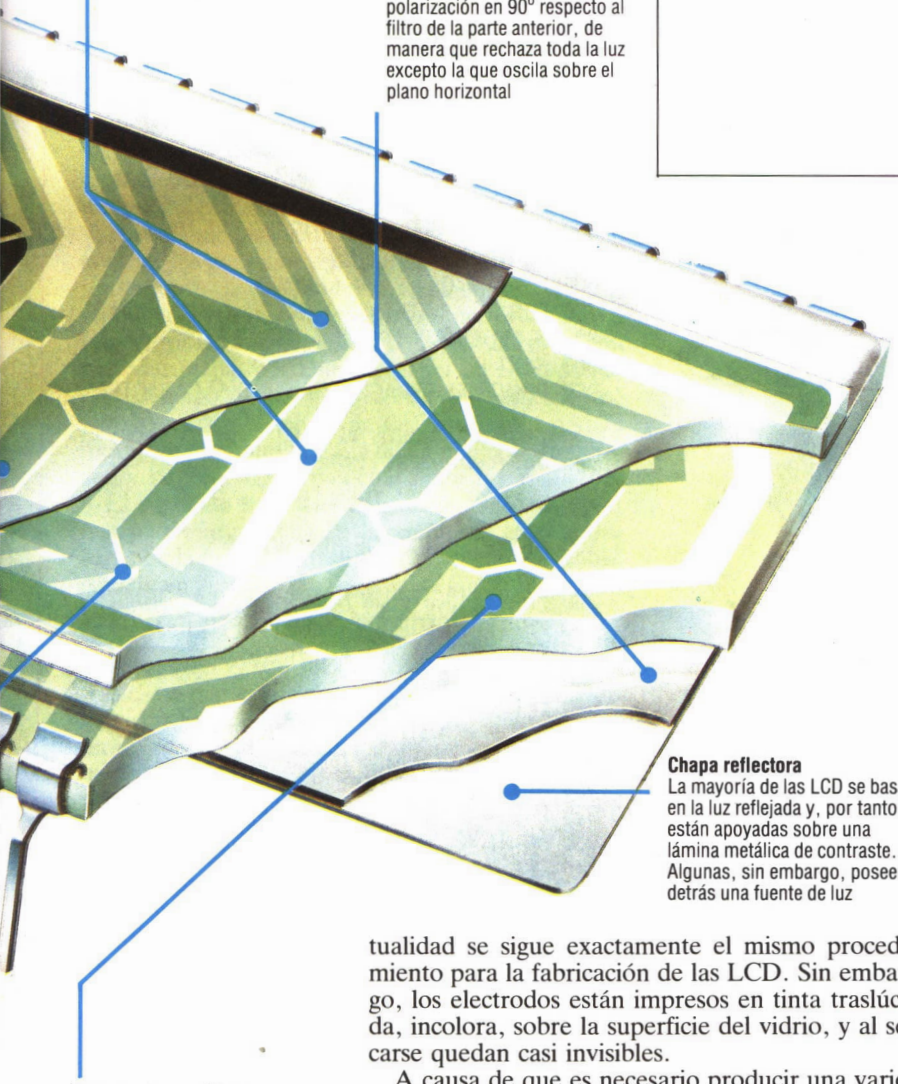
Ángulo de visión
Uno de los refinamientos del Epson HX-20, un ordenador portátil notablemente compacto y sofisticado, es el ajustador del ángulo visual. Los cristales líquidos están compuestos por moléculas largas y delgadas que poseen polos magnéticos situados en el medio de sus lados largos. La aplicación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

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



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

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



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

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

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

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

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

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

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

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

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

Respuestas a los ejercicios

He aquí algunas soluciones a los problemas de la página 157, si bien usted puede haber encontrado otros métodos alternativos

En la página 157 le propusimos nueve problemas, concebidos para comprobar su habilidad en la utilización de las sentencias y funciones más usadas en BASIC. He aquí las soluciones que sugerimos.

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

Si encuentra que las soluciones son tan enigmáticas como las preguntas, vuelva a leer el curso desde la página 119 y vaya estudiando las soluciones a medida que avance. Si ha logrado resolver los ejercicios 2, 4, 6 y 8, ha comprendido la mayoría de las lecciones de nuestro curso.

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

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

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

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

Véase "Complementos al BASIC", de pp. 131 y 149. En el Spectrum, reemplazar la línea 700 por:

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

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

Véase el recuadro "Complementos al BASIC" de p. 119. En el Spectrum este ejercicio se lee:

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

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

Véase "Complementos al BASIC", p. 119. En el Spectrum, reemplazar la línea 800 anterior por:

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

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

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

Véase "Complementos al BASIC", p. 119. En el Spectrum, reemplazar la línea 500 anterior por:

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

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

Su fiel servidor

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

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

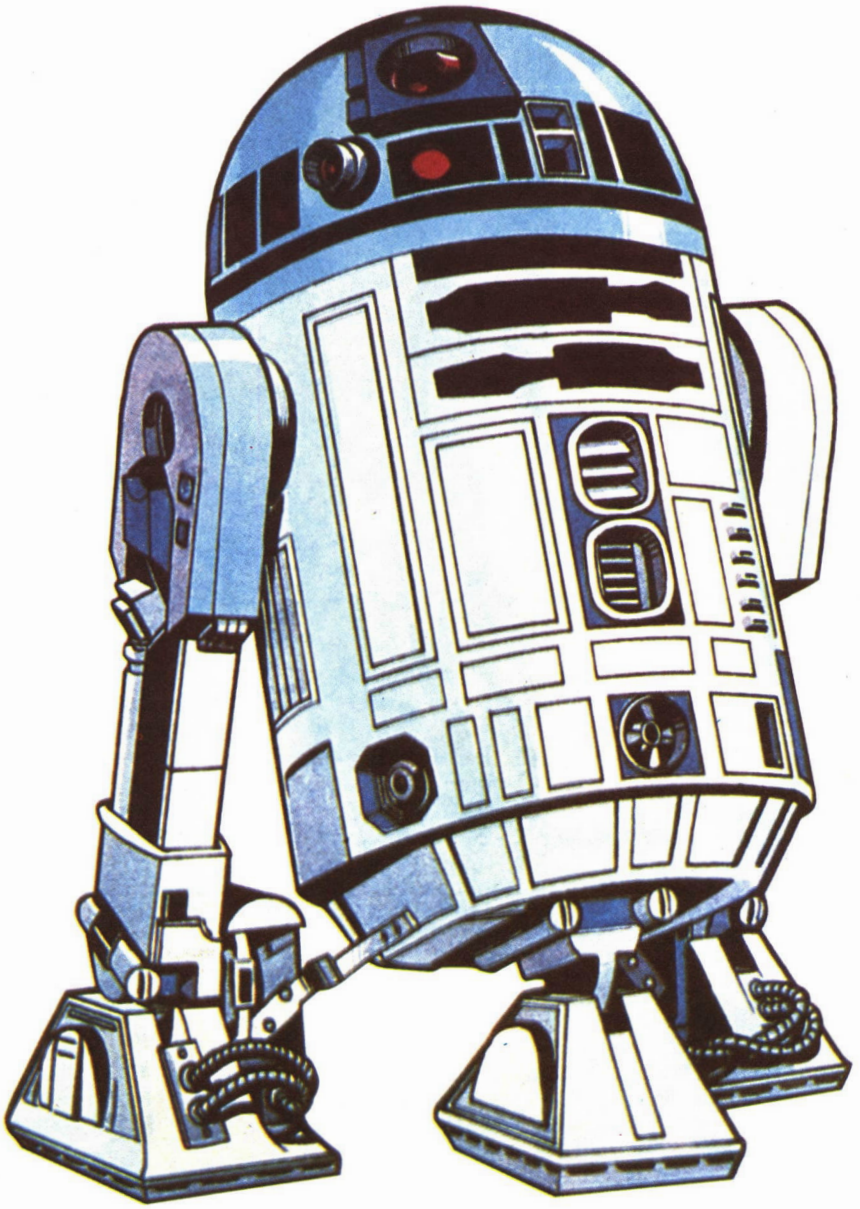
Sus orígenes se remontan a los talleres de maquinaria de los años cincuenta, en los que se aplicó por primera vez la teoría del control numérico a las máquinas-herramienta. Estos primeros esfuerzos fueron, previsiblemente, muy elementales: máquinas controladas por cinta de papel de cinco agujeros (del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

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

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

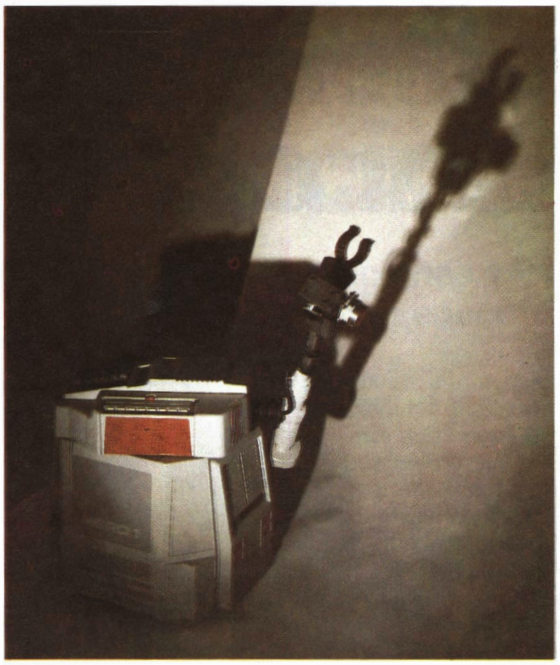
Si deseamos comprender cómo se programan los robots, primero debemos considerarlos en relación al espacio en el cual operan. La mayoría de los robots industriales ocupan una posición fija, de modo que el espacio será una esfera aplanada en su parte inferior, y podemos pensar en la cuestión del control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

No obstante, el problema de programar al robot



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

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



Ian McKinnel

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

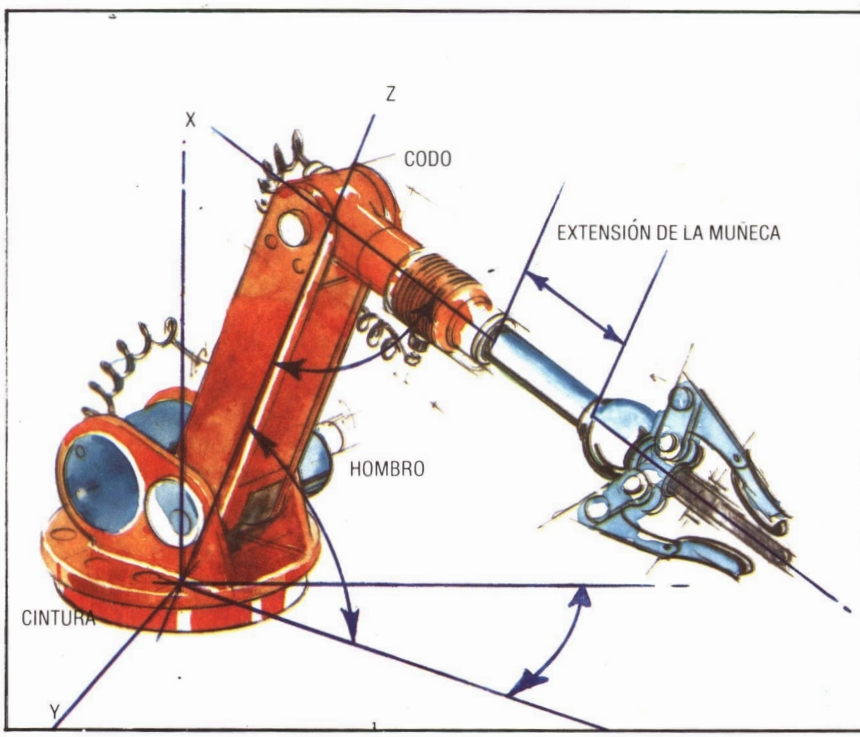
Por lo general, el margen de precisión de los robots industriales es de un milímetro. Incluso los modelos más sencillos (que se pueden utilizar con cualquier ordenador personal que posea una salida en paralelo de ocho bits) tienen un margen de precisión de dos milímetros.

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

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

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

Movimiento angular



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

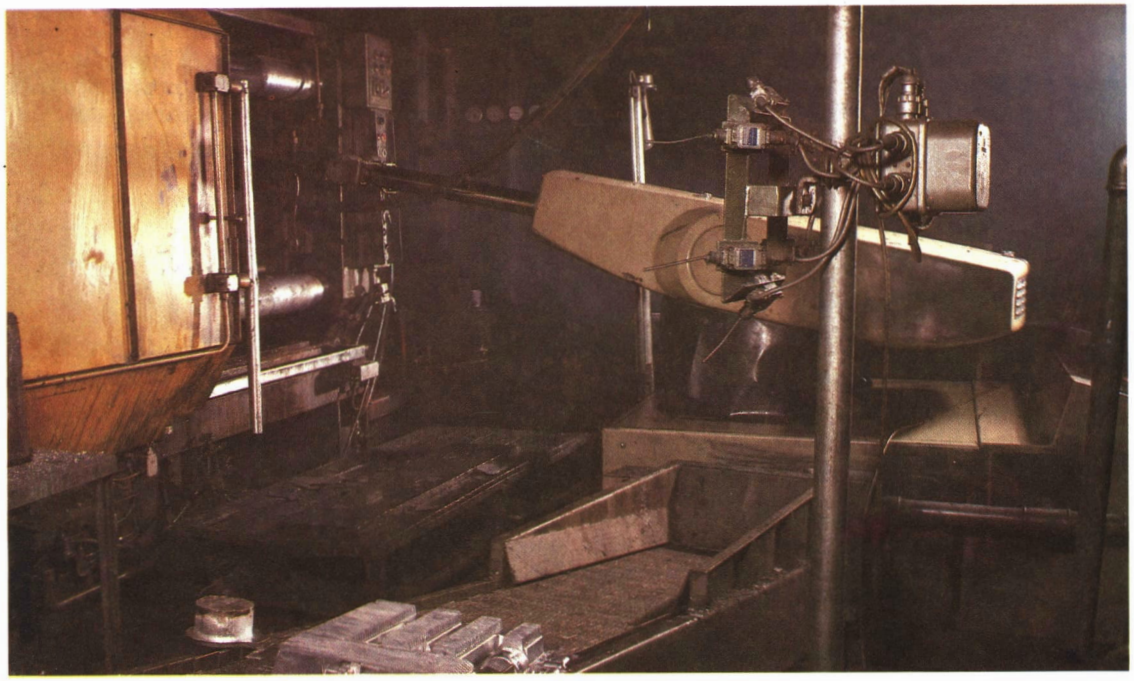
Bob Freeman

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

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

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

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



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

Ordenando la baraja

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

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

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

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

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

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

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

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

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

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

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

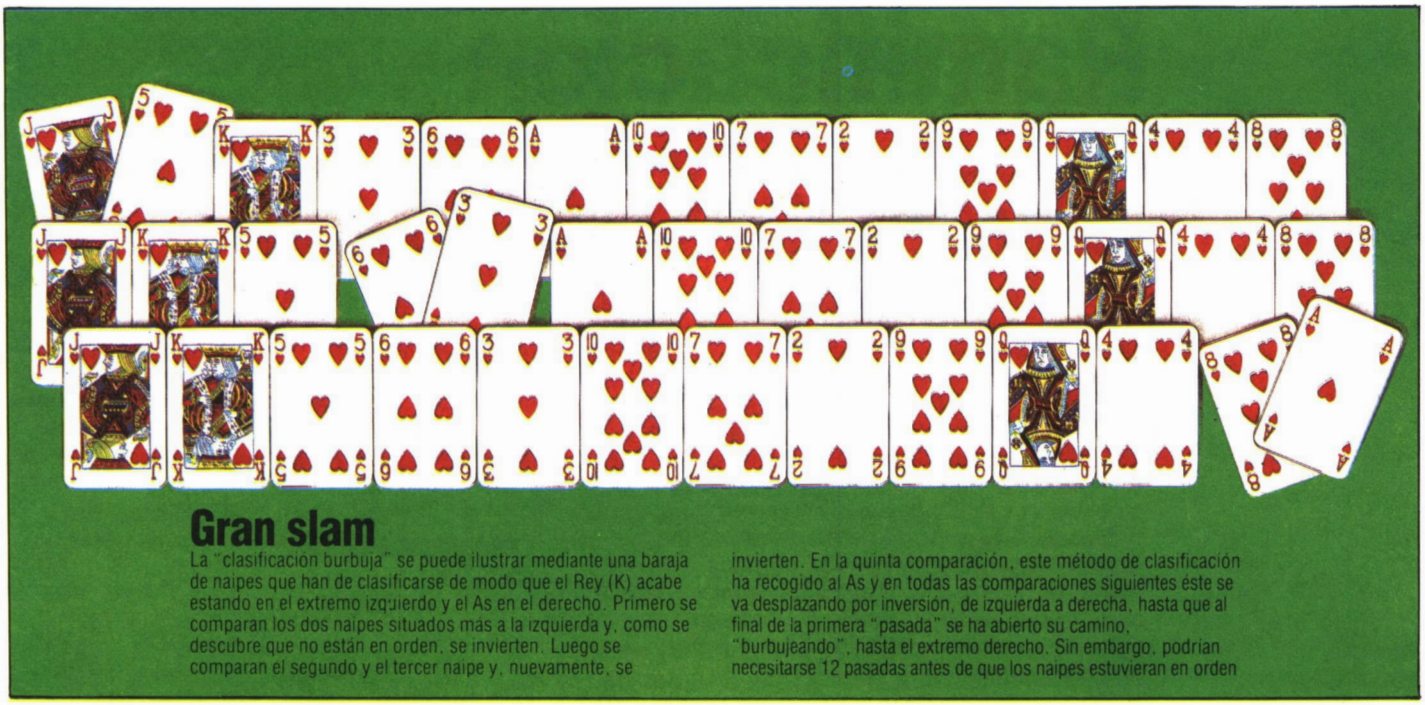
Es posible que haya observado diversos inconvenientes en este método. Es muy tedioso; no económico, ya que la sola acción de intercambiar las posiciones de dos naipes requiere efectuar tres operaciones diferentes; y, sobre todo, muchas de las comparaciones realizadas entre cartas distintas son innecesarias.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

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

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

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

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



Gran slam

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

Tony Lodge

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

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

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

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

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

Clasificación a gran velocidad
Este programa en BASIC muestra la diferencia, en cuanto a eficacia, entre una "clasificación burbuja" y una "clasificación por inserción". El código se ha escrito teniendo siempre presente la velocidad, de modo que no hemos documentado la operación de las rutinas. El listado se debería poder ejecutar en la mayoría de las máquinas, pero lea en p. 149 los "complementos" referentes a ON...GOSUB y en p. 131 los relativos a RND y RANDOMIZE

Ramificación

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

```
GOSUB *ELECCION*
RETURN
```

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

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

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

o

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

Complementos al BASIC

SPECTRUM

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

RUTINA DE VERIFICACION DE ESCALA

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

LISTADO FINAL

```
10 CLS
```

después copiar la lista del texto principal hasta:

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

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

TAB

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

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

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

CHR$(12)

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

ON.. GOSUB

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

VARIABLES

Ver "Complementos al BASIC", p. 169.

INKEY$

Ver "Complementos al BASIC", p. 131. Los usuarios del Commodore sustituirán LET A$ = INKEY$ por GET A$, e IF INKEY$ = "" THEN por: GET A$:IF A$ = "" THEN

Tony Lodge

Términos clave

Peek y Poke

Estas dos órdenes se utilizan cuando se quiere programar algo a lo que no puede hacer frente el lenguaje Basic, pero cada máquina las emplea de modo diferente

POKE
Es necesario utilizar con cuidado la orden POKE, puesto que cambia el contenido de las localizaciones de memoria, y ello podría afectar al funcionamiento del ordenador, y causar la pérdida de un programa. Se puede probar con las siguientes órdenes "seguras" POKE.
En el Atari 400 u 800, si mediante POKE se introduce 1 en la posición 751, el cursor de la pantalla desaparecerá. Pruebe POKE 751,1.
En el Commodore 64, pruebe POKE 1024,1. 1024 es la dirección de la primera localización de pantalla.
En el Spectrum, probar:

```
100 FOR N = 0 TO 6
    STEP 2
110 POKE USR "A" + N,
    BIN01010101
120 POKE USR
    "A" + N + 1,
    BIN10101010
130 NEXT N
140 PRINT "AAAAAAAA"
```

Las A de la línea 140 deben introducirse a través del módulo de gráficos. Al procesar el programa se producirá una línea de símbolos de tableros de ajedrez en miniatura. Sin embargo, también puede producir formas de interferencia en el televisor

PEEK y POKE son dos órdenes del lenguaje BASIC que se emplean en programas avanzados, que necesitan manipular bits y bytes individuales en la memoria. La orden PEEK se usa para examinar el contenido de una dirección específica (localización) de la memoria, y la orden POKE se utiliza para almacenar un número (del 0 al 255) en un punto específico de la memoria.

Mediante ellas el programador puede acceder al funcionamiento interno del ordenador de un modo que no sería posible con otro sistema. Normalmente el BASIC propio del ordenador es el que se ocupa de saber los puntos donde se encuentran almacenados los datos y las variables que definen los caracteres a visualizar en la pantalla. Aunque, por regla general, el operador no debe preocuparse en saber en qué parte de la memoria se hallan localizados estos datos, a veces sí puede ser necesario. La orden PEEK permite hacerlo.

Fácilmente puede escribirse un programa corto para examinar cualquier punto de la memoria:

```
10 REM BUSCANDO LOCALIZACIONES DE MEMORIA
20 PRINT "INTRODUCIR LOCALIZACION DE MEMORIA
   EN DECIMAL"
30 INPUT M
40 P = PEEK(M)
50 PRINT "CONTENIDO DE LOCALIZACION";M;
   "ESTA";P
60 GOTO 20
70 END
```

Esto imprimirá el contenido de la dirección específica, expresada en forma de número decimal. (En realidad, por supuesto, el ordenador lo almacena en binario.) Si se quiere ver a qué es equivalente el contenido en términos de caracteres "imprimibles", el lenguaje BASIC incluye una función que convierte los números decimales en sus caracteres equivalentes. Ésta es la función CHR$, y cambiando la línea 50 se imprimirán los equivalentes de los caracteres:

```
50 PRINT "CONTENIDO DE LOCALIZACION";M;
   "ESTA";CHR$(P)
```

Para examinar la totalidad de la memoria, puede añadirse un bucle FOR...NEXT, suprimiendo la línea 30, cambiando la línea 20 por FOR X = 0 TO 65535 y sustituyendo la línea 60 por NEXT X.

Si se quiere tener tiempo suficiente para ver cuándo se imprime cada carácter, quizá sea necesario añadir un bucle de retraso tras la orden PRINT y antes de NEXT X. Téngase en cuenta también que el límite superior del bucle FOR...NEXT presupone que se tiene una memoria de 64 Kbytes. Este número puede cambiarse para memorias más pequeñas: 16 Kbytes requiere, en notación decimal, 16383; 32 Kbytes necesita 32767, y 48 Kbytes precisa 49151. El desarrollo completo de este programa es:

```
10 REM PEEKING AND PRINTING TODAS LAS
   LOCALIZACIONES
20 FOR X = 0 TO 65535
30 LET Y = PEEK(X)
40 PRINT "LOCALIZACION";X;"=";Y;"=";
50 PRINT CHR$(Y)
60 FOR D = 1 TO 200
70 NEXT D
80 NEXT X
90 END
```

Aunque la función CHR$ convierte los números decimales en sus equivalentes, los caracteres imprimibles se representan con los números 32 a 127. La mayor parte de los ordenadores utilizan los números entre 128 y 255 (el número mayor representable en un solo byte) para caracteres gráficos especiales.

Muchos de los números entre el 0 y 31 desempeñan funciones especiales de control de pantalla. Al pasar el programa, cuando éstos son encontrados en la memoria, serán convertidos, por la CHR$, en efectos de pantalla especiales, por ejemplo, harán que la pantalla se vuelva negra o que el cursor se desplace a la esquina superior izquierda.

La orden POKE es, en esencia, opuesta a PEEK. Permite "escribir" un byte o dato (cualquier número del 0 al 255) en un punto de la memoria. Debe tenerse cuidado al emplear POKE; si se utiliza para introducir un número y se hace en una sección errónea de la memoria, podría destruir o alterar parte del programa esencial. La única forma de remediar este hecho sería arrancar de nuevo el ordenador (desconectándolo y volviendo a conectarlo otra vez, a menos que tuviera un mando de borrado), y esto comporta el riesgo de destruir alguno de los programas. Por consiguiente, antes de utilizar la orden POKE, se debe consultar el manual para hallar en el esquema de la memoria un área designada como *área para el usuario*.

En muchos ordenadores personales, la memoria video (la empleada para almacenar los caracteres que se visualizarán en la pantalla) es accesible al usuario. Normalmente, el ordenador recibe la información sobre la forma de los caracteres a visualizar desde un circuito ROM especial, llamado generador de caracteres, el cual almacena las estructuras de puntos de cada carácter. Pero, por lo general, también es posible emplear una RAM. Si los códigos de las estructuras se almacenan en RAM, se podrán introducir, mediante la orden POKE, nuevas estructuras, expresadas como números decimales, en las direcciones apropiadas de la RAM, y usarlas para definir caracteres visualizables totalmente nuevos.