EVEREST

EL MUNDO DE LOS ORDENADORES

BASIC

Programar es fácil

Ilustraciones: Gerhard Utecht

Basic

Programar es fácil

Rolf Lohberg · Theo Lutz

EVEREST

EL MUNDO DE LOS ORDENADORES

Rolf Lohberg · Theo Lutz

BASIC

Programar es fácil

Ilustraciones: Gerhard Utecht

EDITORIAL EVEREST, S. A.

MADRID • LEON • BARCELONA • SEVILLA • GRANADA • VALENCIA
ZARAGOZA • BILBAO • LAS PALMAS DE GRAN CANARIA • LA CORUÑA
PALMA DE MALLORCA • ALICANTE — MEXICO • BUENOS AIRES

Título original: BASIC, programmieren, ganz einfach
Traducción: Tradutex

EDITORIAL EVEREST, S.A.
Ctra. León-La Coruña, km 5 - LEON
ISBN: 84-241-5324-3
Depósito legal: LE. 560 — 1986

Printed in Spain - Impreso en España

EDITORIAL EVERGRAFICAS, S.A.
Carretera León-La Coruña, km 5
LEON (España)

Prólogo

Un ordenador que tenga que trabajar para nosotros, necesita unas indicaciones precisas y correctas. Estas indicaciones reciben el nombre de «instrucciones». Una serie de instrucciones, con las cuales se puede resolver cualquier problema concreto, se llama «programa».
Las instrucciones de un programa no se escriben tal cual, pues con ello el ordenador no podría ni empezar. Entiende tan sólo un escueto lenguaje de programación, a menudo de aspecto muy curioso, que está formado por una combinación de letras o cifras y signos simbólicos, que para los no iniciados es como una lengua secreta. Del gran número de lenguajes de programación que existe en la actualidad, hay uno que se caracteriza en primer lugar por ser muy sencillo de aprender, en segundo lugar porque lo entienden muchos ordenadores y, en tercer lugar, porque ya hay en el mercado calculadoras de bolsillo que pueden programarse en este lenguaje.
Se llama «BASIC». El nombre es significativo y revela la afición de los norteamericanos (que crearon el BASIC en 1964) por los juegos de palabras: BASIC es la abreviatura de «Beginner's All-purpose Symbolic Instruction Code» o sea, «código universal de instrucciones simbólicas para principiantes». Pero «basic» significa igualmente básico.
Convendría tener a mano el manual del ordenador al leer este libro. Cada vez que previsiblemente haya divergencias, lo indicaremos de inmediato. Lo que queremos aquí no es más que mostrar lo esencial del BASIC y concretamente para los principiantes. Se darán cuenta que no es difícil programar un ordenador. Aunque, eso sí, necesitarán perseverancia y paciencia. La fama de que tan solo un niño prodigio o un pequeño Einstein podrían poner en marcha un ordenador, se remonta a la época inicial del proceso electrónico de datos. Pero ya se ha superado esa época. Les deseamos que muy pronto también ustedes puedan corroborar esta afirmación.

Cómo está organizado un ordenador

Al hablar en las páginas siguientes sobre un ordenador nos referiremos siempre a los conocidos como microordenadores u ordenadores personales, «PC». Estas últimas letras significan «Personal Computer», que es como los norteamericanos designan a un ordenador muy personal, casi privado. Por consiguiente, trataremos los aparatos pequeños hasta el nivel de calculadora de bolsillo.

El ordenador que debe recoger y llevar a cabo un programa, necesita un dispositivo con el que poder dar entrada a este último. Por lo general es un teclado, que es el aparato de entrada. Los resultados que se obtengan aparecerán sobre la pantalla. Esta es el aparato de salida, al igual que una impresora, pero ésta lo realiza registrándolo sobre papel.

Todo lo que el ordenador debe tener en cuenta está dentro de la memoria, que está dividida en células individuales provistas de numeración correlativa. Cuando se introduce un programa, se comunica al ordenador en qué lugar de la memoria se encuentra. Así pues, trabaja instrucción por instrucción.

El teclado

Constituye, para el usuario, el instrumento más importante del ordenador, con el que hace que el aparato le entienda. Dispone de teclas para todos los signos que se necesitan: cifras, letras (mayúsculas y minúsculas) y también los signos especiales requeridos con el BASIC.

Muchas de ellas son teclas dobles o triples.

El cambio de una a otra función se hace por medio de una tecla que en inglés recibe el nombre de SHIFT (=desplazar).

Una tecla muy importante es la de ENTER. Al pulsarla, todo lo que se ha tecleado entra en el ordenador, o sea, en el procesador. Entonces ya está realmente «en» el ordenador, y es entonces cuando puede reaccionar.

La memoria

Almacena los datos y las instrucciones del programa, y constituye un elemento de la moderna microelectrónica: es pequeña, barata y con una enorme capacidad. Puede guardar como mínimo 64 000 signos de teclado. Sin embargo, el número de posiciones de memoria puede alcanzar hasta el millón en los microordenadores y los ordenadores domésticos.

La memoria es direccionable. Esto quiere decir que cada una de sus células tiene un número de identificación, la dirección. Si damos ésta al sistema electrónico, saldrá el contenido de la memoria, o bien será archivado, según sea la orden.

El procesador

Representa algo así como el motor de un automóvil. Hace funcionar al aparato. Para ello puede recoger datos representados electrónicamente y transformarlos en otros datos según un programa, cuyas instrucciones toma una a una de la memoria. El procesador entiende las instrucciones y para cada una de ellas conecta la correspondiente vía o recorrido de datos. Enlaza así las distintas partes del ordenador y las activa. Los procesadores se construyen en la actualidad con elementos microelectrónicos.

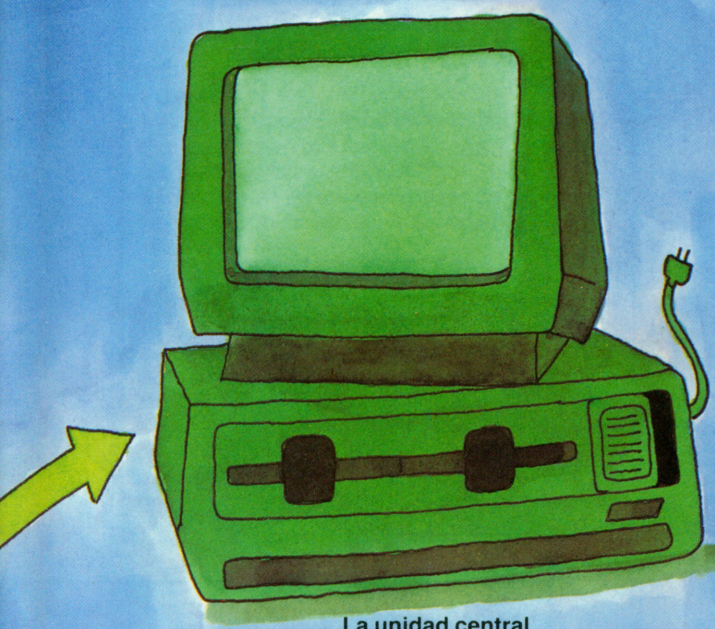
Por esta razón se habla también de microprocesadores.

La pantalla

Es la cara del ordenador. En ella puede verse lo que se ha pulsado en el teclado y lo que se quiere introducir. Pero presenta también lo que el ordenador tiene decir, ya sea un cálculo que ha realizado o cuando se ha cometido una equivocación.

La superficie de la pantalla está dividida en líneas. Cada una de ellas puede admitir una determinada cantidad de caracteres, aproximadamente unos 80. Si hay 25 líneas, hay espacio en la pantalla para 2 000 caracteres. Esto es lo que cabe en una hoja de formato DIN A 4.

La posición que en cada instante preciso se puede describir se señala mediante una raya luminosa intermitente u otro signo similar. Es el cursor. Se le puede desplazar en cualquier dirección sobre la pantalla por medio de cuatro teclas que llevan flechas, y de esta manera se localiza cualquier posición. Si se pulsa entonces una tecla, el signo correspondiente quedará entonces encima del cursor.



Distintos tipos

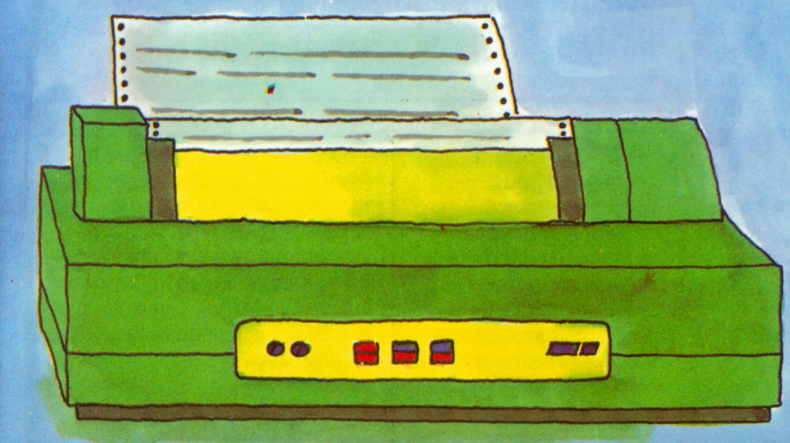
La ilustración muestra un microordenador en el que el teclado, la pantalla y la unidad central son aparatos distintos. Hay otros modelos en los que todas estas funciones se encuentran agrupadas dentro de un mismo aparato. Para el usuario no tiene demasiada importancia, pues no afecta para nada al modo de funcionamiento del ordenador. Lo máximo que varía es el precio y la comodidad de manejo de los aparatos.

Existen asimismo ordenadores domésticos en formato de calculadora de bolsillo y que utilizan el BASIC. El teclado en tal caso es simplemente el de una calculadora. La pantalla es la de una línea como la que tienen estas últimas, y para trabajos pequeños es suficiente. Lo que nos importa es que dominen también el BASIC, aunque se trate tan sólo de un dialecto de pocas palabras de este lenguaje.

La unidad central

Puesto que en la memoria están todos los programas válidos y por ella pasan todos los datos, y como el procesador es también el elemento central de nuestro ordenador, es por esta razón por lo que se montan juntos la memoria y el procesador.

Al conjunto se le denomina «unidad central». El término ya es expresivo por sí mismo. Dado que estas unidades son muy pequeñas en los actuales microordenadores, el espacio sobrante en la carcasa se utiliza para colocar otros elementos. Por ejemplo, los mecanismos de desplazamiento de los disquetes, sobre los que se pueden almacenar grandes cantidades de datos e incluso programas. Pero dentro de la carcasa o caja están alojados también los elementos electrónicos de otros aparatos que pueden conectarse. Tal es el caso de la palanca de mando para los juegos de ordenador.



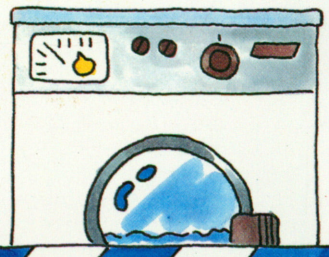
La impresora

Sirve para registrar en blanco y negro todo lo que sale del ordenador. Se pueden incluir en el programa instrucciones que ponen en marcha la impresora, pero por lo general hay una tecla que al ser pulsada hace que pase a ella todo lo que hay sobre la pantalla. Así de sencillo es en el caso de los microordenadores.

La impresora suele tener un rollo de papel continuo y no hojas separadas. Pero todo el que ha trabajado con impresoras sabe que siempre en el momento más inoportuno es cuando se acaba el rollo.

El programa son instrucciones

Un programa es una orden al ordenador: debe hacer lo que se quiere de él. Establece con precisión en qué orden deben usarse las distintas partes del aparato y cómo deben trabajar conjuntamente para resolver el problema planteado. Un programa de este tipo, aunque bien sencillo, es el que tienen las lavadoras. Determina cuándo debe abrirse la válvula para que entre el agua, conecta después el calefactor para que se caliente y por último hace girar el tambor. Un programa dice al ordenador que hay que leer datos, cómo deben procesarse y qué tiene que imprimirse. Está formado por varios pasos, que son instrucciones u órdenes. Se dan en BASIC mediante palabras procedentes del inglés americano. Resulta un poco molesto al principio, pero se aprende enseguida. No es necesario saber inglés, y las palabras son sencillas. En el BASIC aparecen también signos tales como + para sumar y × para multiplicar. Esto facilita mucho las cosas.



Un pequeño ejemplo

El dinero que está en el banco produce intereses. Su tipo es un porcentaje. El 5 por ciento de 100 son 5 y el de 500 cinco veces más, es decir, 25. Aquí tenemos un programa que a partir de un CAPITAL calcula los INTERESES cuando el tipo de interés se llama PORCENTAJE:

1 LEER CAPITAL, PORCENTAJE
2 Calcular INTERES = CAPITAL × PORCENTAJE/100
3 Imprimir INTERESES

Las instrucciones en BASIC

Aquí veremos el programa «intereses» con las palabras clave en el lenguaje BASIC. Cada una de las instrucciones comienza con un número de por lo menos dos cifras y una de esas palabras clave. Es decir, que para un tipo de instrucciones sería:

```
10 INPUT CAPITAL, PORCENTAJE
20 LET INTERESES = CAPITAL ×
     PORCENTAJE / 100
30 PRINT INTERESES
40 STOP
```

INPUT, LET, PRINT y STOP son nuestras primeras palabras clave en BASIC. INPUT exige la entrada de datos, con LET se calcula, PRINT indica al ordenador que hay que dar salida. El BASIC tiene alrededor de 50 palabras clave de este tipo.

BASIC es un lenguaje de programación

El lenguaje que el ordenador entiende directamente se llama «lenguaje de máquina». El ordenador traduce automáticamente a éste los lenguajes de programación de alto nivel, que son con los que las personas se entienden mejor. El BASIC es uno de estos lenguajes. Resulta fácil de aprender y es muy adecuado para los ordenadores pequeños. Por esa razón, la mayoría de los ordenadores domésticos entienden el BASIC.

El BASIC consiste en palabras fijas procedentes del inglés americano. Son palabras de instrucción o clave. Hay además nombres para datos que hay que tratar, y caracteres tales como + o × que tienen determinados significados. Incluye asimismo reglas que determinan el modo como se escriben las instrucciones.

Los datos tienen nombres

Para la cantidad de dinero que debe rendir unos intereses hemos escrito la palabra CAPITAL y de esta manera hemos dado un nombre a un número. Esto es lo habitual en los ordenadores y puesto que el nombre significa unas veces un valor y otras uno distinto, se llama «variable».

En BASIC a las variables numéricas se las designa siempre por una letra

En la memoria están los datos y las instrucciones

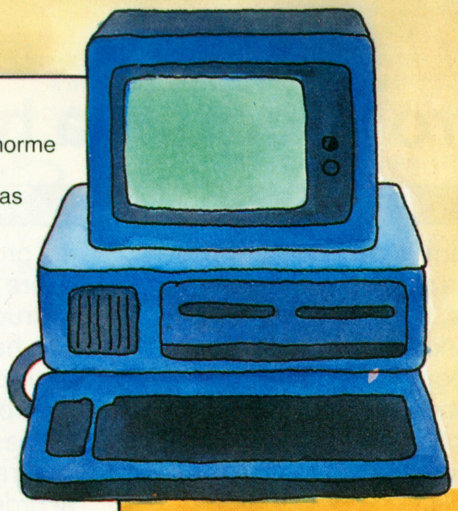
La memoria de un ordenador es un enorme almacén para datos e instrucciones. Está dividido en celdas, cada una de las cuales registra un carácter y tiene un número que es su dirección. En el lenguaje de máquina hay que tratar con estas direcciones. Cuando se trabaja con BASIC no existe este problema, pues el lenguaje de programación lo hace por uno.

Las instrucciones vienen caracterizadas por números tras los cuales vienen dos puntos. El orden natural en el que están determina el orden en el que fueron almacenadas las instrucciones, y casi siempre también en el que tienen que ser tratadas.

Los datos se almacenan en BASIC —y análogamente en otros lenguajes de programación de alto nivel— mediante nombres.

Una letra (o una letra más un número) designan valores numéricos. Las letras que van seguidas del símbolo del dólar $ contienen literales, o sea, información alfabética.

La administración de las direcciones de memoria corre a cargo del propio BASIC, o mejor dicho, del microprocesador, que está organizado en BASIC. Establece en cuadros qué nombres corresponden a cada dirección, de manera análoga a como hace una guía de teléfonos.



La memoria en el microprocesador

El procesador sólo puede procesar lo que tiene en la memoria. Y los datos, y sobre todo las instrucciones, tienen sentido en la memoria únicamente si alguien los solicita. Este alguien es el procesador. Una memoria de trabajo puede almacenar en la actualidad varios miles de caracteres. En términos técnicos mil se dice «kilo». Si una memoria tiene una capacidad de 64 kilos o 64K, puede guardar 64 000 caracteres. 64K son suficientes para muchas tareas. Algunos microordenadores pueden almacenar un millón de caracteres. Se dice que tiene 1 mega o 1M. Una memoria de trabajo consta de varias memorias parciales. Dos de ellas se llaman RAM o ROM, y se las trata en uno de los campos de esta página. Existen también los registros, que son las memorias de datos más pequeñas para pocos símbolos.

Los intereses con BASIC

Ahora ya podemos escribir en BASIC puro nuestro ejemplo de los intereses, dando a los datos las primeras letras del nombre que teníamos hasta ahora. De intereses será I, de capital una C y de porcentaje una P. Detrás del número de la instrucción hay que poner dos puntos. El programa completo tiene entonces el siguiente aspecto:

```
10 : INPUT C, P
20 : LET I = C × P / 100
30 : PRINT I
40 : STOP
50 : END
```

Con STOP el programa se detiene y con END se llega al final en la memoria.

¿Qué es un registro?

Un ordenador necesita registros para los fines más diversos. Son pequeñas memorias muy rápidas que registran datos en volumen de un número. Si se quiere sumar algo, se cargan dos registros cuyos contenidos se cuentan entonces conjuntamente. Un usuario de BASIC no necesita saber nada acerca de ellos, puesto que su utilización es automática.

y cuando esto ya no es suficiente, se les añade un número de una cifra. Si en un programa de BASIC se escribe

```
135 LET K8 = 1000
```

la instrucción 135 tiene en la variable K8 el valor 1000.

Muchas veces, bajo un nombre no se almacenan números sino letras y caracteres. A estos datos de letras se les llama «literales».

¿Qué es ROM?

Es la abreviatura de «**R**ead-**O**nly-**M**emory», memoria de sólo leer. Es una memoria a la que sólo se puede leer ya que el fabricante ha hecho en ella registros definitivos. Suministra en ROM programas fijos, precisamente aquellos que el BASIC traduce al lenguaje de máquina. ROM suele albergar menos información que su pariente RAM.

¿Qué es RAM?

Es «**R**andom-**A**ccess-**M**emory», una memoria en la que uno mismo puede escribir. Registra lo que procede de fuera o de dentro y lo transmite hacia fuera, al usuario, o hacia dentro, al procesador.

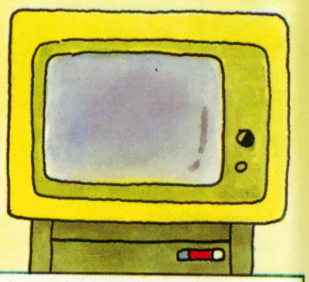
Si se lee en RAM, el contenido se conserva y se obtiene una copia.

Nuestro programa hace cálculos

Continuamos ocupándonos del programa que calcula en BASIC los intereses. Primero hay que introducirlo en el ordenador, algo que se logra con pocas operaciones. También es muy sencillo corregir un programa al que se ha dado entrada. Para ello es importante tener el número de instrucción (o de programa) bajo el cual se encuentra ésta.
Igual de fácil es poner entonces el programa en marcha, suponiendo que todo esté en orden. Si no es así, si se han saltado las reglas del BASIC, el procesador lo señalará. Por ejemplo, cuando se utilizan nombres que no están autorizados. El procesador detecta también cuando al teclear se han cometido errores. Según el tamaño y el tipo del ordenador, lo señala además sobre la pantalla. En el manual de instrucciones se dice lo que hay que hacer en ese caso. Las equivocaciones principales se perciben enseguida, pero lo que el procesador no ve es cuando se han programado mal, como puede ser al ordenar «sumar» cuando lo correcto habría sido «multiplicar».

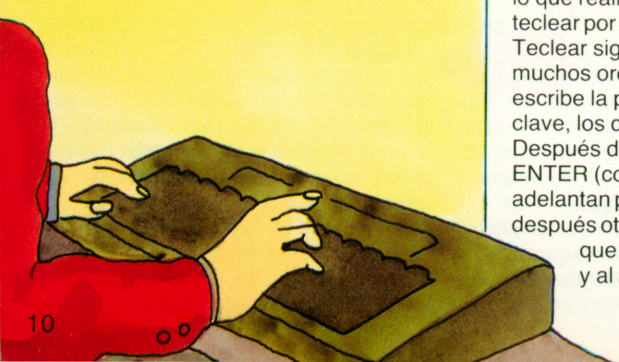
Un buen comienzo

Cuando se inicia de nuevo un programa y se sabe exactamente lo que se quiere programar, lo primero que hay que hacer es poner en orden la memoria. Se teclean las entradas NEW y ENTER.
El ordenador borra entonces la memoria de programas. Se añade CLEAR y se acaba con ENTER y entonces se borrarán también todas las memorias de datos. Con NEW y CLEAR se tiene en cierta medida una hoja en blanco.
En la mayoría de los microordenadores hay también una tecla con la que se puede borrar todo lo que hay en la pantalla. El cursor queda entonces en el margen superior izquierdo. Esa tecla puede llamarse por ejemplo CLS, «**Cl**ear-**S**creen» (¡borra la pantalla!).



Tecleamos

Antes de entrar un programa, hay que preparar sobre una hoja de papel lo que realmente se quiere. Es aconsejable que los principiantes al teclear por primera vez un programa lo hagan con uno que sea correcto. Teclear significa que primero se indica un número de programa (con muchos ordenadores es necesario colocar detrás dos puntos) y se escribe la primera orden. Dejando espacios que separen las palabras clave, los datos y los nombres de éstos, todo queda más legible. Después de la instrucción se vuelve a teclear su número y por último ENTER (con los números de programas o de instrucciones se adelantan por conveniencia de diez en diez, y así se pueden intercalar después otras instrucciones). Si en la pantalla no aparece nin guna nota que indique lo contrario, la instrucción está en la memoria y al mismo tiempo desaparecerá de pantalla.



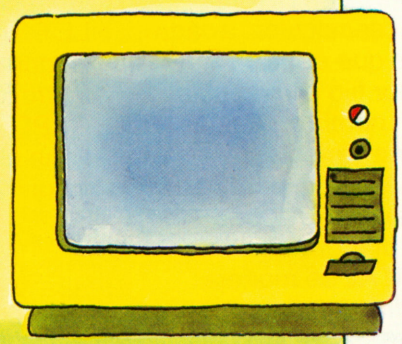
Ordenes y mandatos

Hay que distinguir entre las órdenes, que van fijas en el programa, y los mandatos, con los cuales el usuario dice al aparato lo que debe hacer. Mientras que la orden PRINT pasa datos del programa a la pantalla o a la impresora, el mandato LIST que da el usuario, lleva a la pantalla el programa que conviene solamente una de sus instrucciones). Los mandatos son para el manejo, y no han de buscar nada en el programa. Y a la inversa, con las órdenes —tales como IMPUT o PRINT— no se puede manejar el aparato.

Comenzamos el programa

Cuando el programa es correcto, es muy fácil iniciarlo. Si se da el mandato RUN ENTER, comienza y lo hace con el número de instrucción más bajo. Si escribimos RUN 45 ENTER, el programa empezará con la instrucción 45. Así de sencillo.
Cuando la instrucción es falsa, aparecerá una indicación.
Con LIST y el número de la instrucción equivocada se la sacará de pantalla y se corregirá el error.

El programa de los intereses en marcha



Cuando hemos introducido el programa de los intereses, nuestra memoria principal presenta el siguiente aspecto:

```
10: IMPUT C,P
20: LET I = C × P / 100
30: PRINT I
40: STOP
50: END
```

Si tecleamos ahora RUN ENTER, se desarrollará primero la instrucción 10. Es la instrucción de entrada. El ordenador queda a la espera de valores para C y P. Lo indica mediante un signo de interrogación que aparece en la siguiente línea libre, como petición para que se dé un valor a C.
Tecleamos 1000 ENTER. De nuevo aparece el signo de interrogación. Para P damos 5 ENTER. A partir de ahora todo discurre más rápidamente de lo que podemos escribir. El ordenador forma I en la instrucción 20 y con la instrucción 30 da el resultado.

Ya conocemos los siguientes mandatos

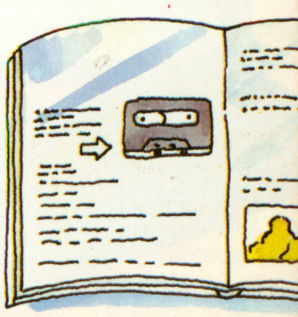
ENTER	¡Lleva lo que se ha tecleado a la memoria de trabajo!
NEW	¡Borra la memoria de programas!
CLEAR	¡Borra la memoria de datos!
INSERT	¡Incluye un carácter! (tecla especial)
DELETE	¡Borra el carácter! (tecla especial)
LIST	¡Muestra el programa!
LIST 50	¡Muestra la instrucción número 50!
RUN	¡Inicia el programa con la instrucción de número más bajo!
RUN 100	¡Inicia el programa con la instrucción 100!
SAVE	¡Guarda el programa en disquete o cassette!
LOAD	¡Lleva el programa guardado a la memoria de trabajo!

Error al teclear

Si nos damos cuenta de haber cometido un error en una instrucción, por ejemplo la 45, se la retiene en pantalla tecleando LIST 45 ENTER. Si lo único que está mal es un carácter, se coloca sobre él el cursor con la tecla de flecha y se escribe encima correctamente. Al teclear ENTER la instrucción corregida pasa al programa. Si hay que incluir un carácter, se lleva el cursor al lugar que corresponda, se pulsa la tecla que diga INSERT (en inglés: insertar), y se escribe lo que falta. El resto corre automáticamente los espacios necesarios hacia la derecha. Con algunos ordenadores hay que pulsar INSERT para cada carácter nuevo.
A la inversa, la tecla DELETE (en inglés: borrar) sirve para quitar un carácter, así que el corregir también es muy fácil.

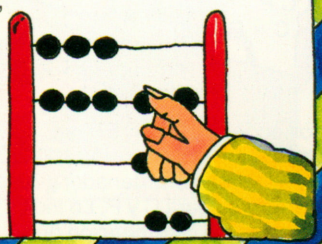
Se guarda el programa

Si nuestro ordenador tiene un disquete o una cassette para almacenar programas, existe entonces también un mandato con el que se puede almacenar allí nuestro programa, o dicho más exactamente, una copia de él. Esto se hace con la palabra SAVE («guardar»). Pero primero hay que consultar en el manual todos los detalles. Y a la inversa, con LOAD (de «cargar») se pasa un programa del disquete o de la cassette a la memoria de trabajo.

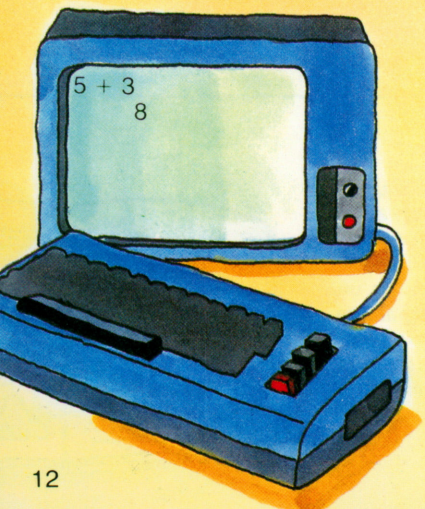


Calculamos con números

Hacer cálculos en BASIC no es difícil; sólo hay que aprender. Un ordenador BASIC se utiliza por lo general igual que una calculadora de bolsillo. Se indican los cálculos en el teclado y la máquina da el resultado sobre la pantalla.
En los programas, para cada una de las operaciones se puede escribir una orden. Existen programas largos, pero con cálculos complejos también es posible reunirlo en una fórmula. Esto puede hacerse en BASIC.
Si se quieren hacer cálculos, a cada una de las cifras que entran y que no tienen un valor fijo e invariable, se les asigna un nombre. A estas cifras se las llama variables.
Si escribimos CAPITAL, para cada nuevo cálculo detrás se puede ocultar un valor diferente.
El nombre CAPITAL se maneja igual que una cifra. A estas variables se las puede sumar, restar, multiplicar o dividir: se puede hacer con ellas todo lo que se hace con los números. En las fórmulas es posible mezclar estos nombres con valores fijos.
De todas maneras, un nombre numérico de este tipo únicamente puede emplearse en los cálculos si previamente se le ha asignado un valor fijo. Esto se hace por medio de la entrada con la instrucción INPUT, o bien con una instrucción de asignación como LET CAPITAL = 1250.
Según esta instrucción, CAPITAL tiene el valor 1250.



El BASIC para un ordenador de mesa
La mayoría de los microordenadores de BASIC se presentan en formato de mesa. Con ellos se trabaja con la misma facilidad que con una calculadora de bolsillo. Los paréntesis (y) significan que esto ha de teclearse. Y si escribimos (E) queremos decir la tecla ENTER.



Sumar
¿Cuántas son 3 + 5? Hay que teclear (5)(+)(3)(E). En la pantalla aparece lo que se ve en la imagen de la izquierda. Lo que hemos tecleado es lo que se encuentra en el extremo izquierdo, y la respuesta del ordenador nos aparece un poco más abajo, hacia la derecha.

Restar
Con el ejercicio «15 menos nueve» tecleamos: (1)(5)(−)(9)(E)

Multiplicar
Hay que multiplicar tres por cuatro, y para ello teclearemos (3)(×)(4)(E)

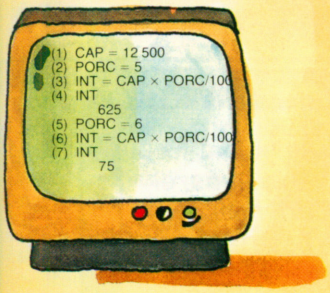
Dividir
Dividimos el número 10 entre cuatro. ¿Qué resulta? Marcaremos (1)(0)(/)(4)(E)

Elevar al cuadrado
Cuando se eleva un número al cuadrado se le multiplica por sí mismo. En los distintos dialectos del BASIC unas veces es de una manera, y otras de otra. Mostraremos aquí dos versiones para elevar el número cuatro al cuadrado:
Versión 1 (4)(↑)(2)(E)
Versión 2 (4)(×)(×)(2)(E)
El resultado es 16.

Asignación
Queremos dar un nombre a un número. Dicho de otra manera: bajo el nombre de CAPITAL queremos almacenar un valor numérico. Será el 1250. Entonces hay que marcar de esta forma: (C)(A)(P)(I)(T)(A)(L)(=)(1)(2)(5)(0)(E). Lo que vemos en la pantalla es: CAPITAL = 1250
Siempre que no se asigne nada más a la voz CAPITAL, bajo este nombre aparecerá siempre el valor numérico 1250.

Un ejemplo

Vemos sobre la pantalla de un ordenador doméstico lo siguiente:



Alguien tiene aquí problemas con intereses entre el 5 y el 6 por ciento. Resulta ventajoso poder tomar del ordenador las líneas (3) y (6) y no tener que escribirlas de nuevo.
Por esa razón se crearon los programas.

Los paréntesis

En los cálculos más complejos aparecen también paréntesis. Lo que está dentro de ellos va junto, y se calcula siempre primero. Después vienen las funciones (como por ejemplo la raíz), a continuación se hacen los cálculos con × y /, y por último con + y − . Cuando no se está seguro se utilizan paréntesis, pues todo lo que llevan dentro será lo primero que entra en los cálculos.

Las funciones

A menudo hacen falta relaciones más complejas entre los números: son las llamadas funciones, como por ejemplo la raíz. La raíz de 9 es tres porque 3 multiplicado por sí mismo da 9. Este caso es sencillo, pero por regla general aparecen muchos decimales detrás de la coma. Si se desea en BASIC la raíz de un número X, hay que escribir SQR(X). SQR son las siglas de la palabra inglesa **SQ**uare-**R**oot, es decir, raíz cuadrada.

Un programa completo

El programa del interés de nuestro ejemplo, puede hacer cálculos e imprimir. Cuando las cifras proceden de fuera, hace falta una instrucción de entráda: INPUT. A continuación vienen los nombres de datos: CAPITAL y PORCENTAJE.
Con INPUT para la entrada, LET para el cálculo y PRINT para la salida de datos, se pueden calcular ya muchas cosas:

```
10: INPUT (nombres de números)
20: LET (nombres de números,
    números y signos de cálculo)
30: PRINT (nombres de números)
40: END
```

Se comienza con RUN 20

Fórmulas

Para el BASIC un nombre numérico significa un número, y siempre aquél que se le ha asignado el último. Pero no sólo son números o cifras los que se pueden juntar, sino también nombres numéricos. Lo que resulta de todo ello son fórmulas. Unicamente hay que tener en cuenta que antes se le ha asignado a cada nombre un valor.

Números enteros y naturales

Los números más cómodos son siempre los naturales: 1, 2, 3, etc. Son los que conocemos al contar con los dedos. Con ellos es sencillo sumar, multiplicar y restar. Pero si se quiere restar 5 a 3, surgirá un pequeño problema: aparecen los números negativos. Los números naturales negativos, el cero y los números naturales positivos se agrupan dentro de un concepto global, al que llamamos «números enteros». En inglés se llaman INTEGER.

El camino hasta el programa

Nuestro ejemplo, arriba a la izquierda, muestra una sucesión de procesos para calcular dos veces el interés. Con ello se hace fácilmente un programa completo de BASIC. Sólo hay que completar las líneas 4 y 7 con la orden PRINT y los números de las líneas con dos puntos.

Muchos números

Por lo general, se tienen muchos números que hay que unir. Por ejemplo, el interés de 12 500 pesetas se obtiene cuando se multiplica esta cantidad por el porcentaje (en nuestro ejemplo es el 5%) y se divide después por cien. Lo que hay que marcar es:
(1)(2)(5)(0)(0)(×)(5)(/)(1)(0)(0)(E)
La pantalla muestra:
12 500 × 5/100
625
Si se quiere almacenar este interés, hay que hacer una asignación:
INTERES = 12 500 × 5/100

Fracciones y números racionales

Si se divide 1 entre 3 el resultado da infinitos decimales, un cero seguido de treses. Nos encontramos entonces en el campo de las fracciones o de los «números racionales». Son los que aparecen detrás de la coma, a menudo infinitos, y que se repiten periódicamente. Pero en el ordenador sólo hay un número determinado de posiciones, por ejemplo diez. Por lo tanto, si representamos 1/3 por 0,333... cometemos un pequeño error. Esto se observa muchas veces cuando se divide primero 1 entre 3 y se vuelve a multiplicar después por tres. Es frecuente enfrentarse a problemas de imprecisión.

Números reales

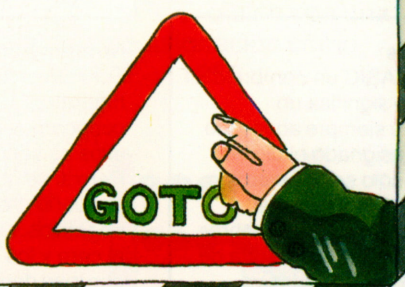
El valor numérico de la raíz de 2 es 1,4142135, aunque es muchos más largo: detrás de la coma hay infinitas posiciones. Sin embargo este número no es periódico. Las cifras se entremezclan. El término REAL, junto con INTEGER, constituye un tipo numérico importante para los ordenadores. El BASIC se las arregla perfectamente con ellos.

La coma flotante

¿Qué hace un ordenador con los números más grandes? En primer lugar, toma las diez primeras cifras que puede almacenar y sustituye el resto por ceros. De esta manera, con un número de 15 cifras, como puede ser 123456789876543, deja 123456789800000. A continuación desplaza la coma decimal desde la derecha detrás de la primera cifra e indica la cantidad de posiciones que hay que desplazar. En nuestro ejemplo son 14. Por lo tanto, la calculadora nos dará representado 1,23456789 E14, en donde E significa «exponente»; el número que hay detrás señala las posiciones que hay que desplazar hacia la derecha para llegar al número original. A esto se le llama representación de coma flotante.

El programa hace un lazo

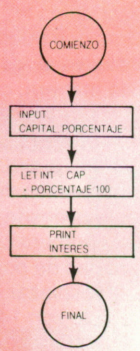
Los programas sencillos constan de tres partes. Primero se leen los datos. Después se hacen los cálculos con ellos y, por último, se imprime el resultado. Puesto que todo transcurre como en línea, a un programa de este tipo se le llama «lineal». Si se quieren calcular con este programa varios casos similares, se le repite desde el principio. Pero también puede hacerlo él por sí mismo.
Únicamente hay que indicarle cuándo ha de abandonar su curso lineal y dar la vuelta. A esto se llama «bifurcarse», y para ello tenemos la instrucción GOTO (=ve hasta...).
Le corresponde un número de instrucción que indica el punto de bifurcación. Con GOTO se puede bifurcar hacia donde se quiera. Si se vuelve de nuevo hacia atrás, el ordenador ya no se para nunca. Si se desea que lo haga hace falta una instrucción que sólo permita bifurcar hasta que se satisfaga una determinada condición. Es la instrucción IF (=si).
Si las bifurcaciones se hacen básicamente con IF, el programa lineal describirá un lazo.



El programa lineal
Volvemos al programa del interés:

```
10: INPUT CAPITAL,
    PORCENTAJE
20: LET INTERES = CAPITAL ×
    PORCENTAJE / 100
30: PRINT INTERES
40: END
```

Para un programa de este tipo puede hacerse un diagrama:



En el primer recuadro se leen los datos, en el segundo se procesan y en el tercero se editan. A un diagrama de este tipo se le llama organigrama. RUN y END consituyen el marco lógico. Es un «programa lineal», dispuesto en línea recta: una cosa detrás de otra.

Tenemos varios casos de intereses
Cuando hay que calcular varios intereses, se dejan INPUT, LET y PRINT tal como son, pero incorporamos una instrucción adicional:
35 INPUT «SI = SEGUIR, No = STOP»; M$. La respuesta está en M$. Con otra instrucción comprobamos lo que hay en M$, la instrucción IF:
37 IF M$ = «SI» GOTO 10
De este modo en el caso, SI, el programa retrocede a 10; comienza un nuevo paso:

```
10: INPUT CAPITAL,
    PORCENTAJE
20: LET INTERES =
    CAPITAL ×
    PORCENTAJE / 100
30: PRINT INTERES
35: INPUT «SI = SEGUIR,
    NO = STOP»; M$
37: IF M$ = «SI» GOTO 10
40: END
```

Hacemos un cuadro
Muchas veces es útil no tener que calcular cada caso particular de intereses. Es mejor preparar un pequeño cuadro que en diez etapas nos dé un interés. Un programa de este tipo será así:

```
10: INPUT CAPITAL
15: LET N = 0
17: LET PORCENTAJE = 1
20: LET INTERES = CAPITAL ×
    PORCENTAJE / 100
30: PRINT INTERES
33: LET N = N + 1
35: LET PORCENTAJE = PORCENTAJE
    / 0,5
37: IF N < = 10 GOTO 20
40: END
```

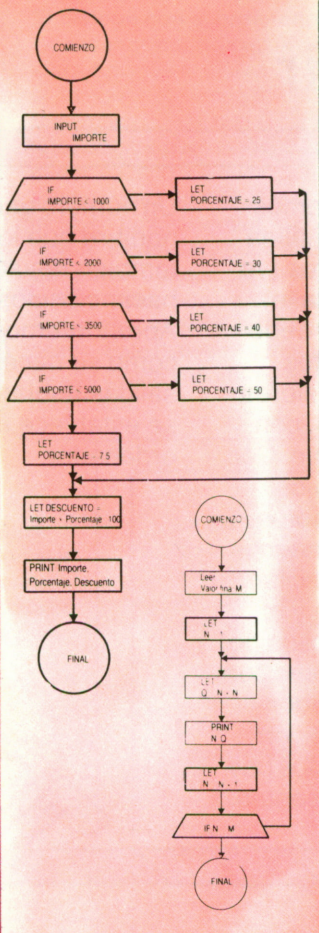
¿Puede leer usted este programa? En la instrucción 15, se ajusta una variable al valor 0. Esta variable sirve para contar hasta 10. Por esa razón, en la línea 33 a cada paso se suma 1. De manera similar, en cada paso en la línea 35 el campo del programa aumenta en 0,5. En la línea 37 se pregunta si se ha llegado ya al final. Si no, se vuelve al 20.

«IF» significa «si (condicional)», incluso en descuentos

Con la instrucción IF no solamente se puede prever la interrupción de un programa, sino que también es posible distinguir dentro del programa los casos más diversos.
Así, un comerciante hace descuentos según el valor de la compra. El que sepa leer el programa podrá conocer cuál es el descuento que corresponde para cada cantidad:

```
10: INPUT IMPORTE
20: IF IMPORTE < 1000 LET
    PORCENTAJE = 2.5: GOTO 70
30: IF IMPORTE < 2000 LET
    PORCENTAJE = 3.5: GOTO 70
40: IF IMPORTE < 3500 LET
    PORCENTAJE = 4.0: GOTO 70
50: IF IMPORTE < 5000 LET
    PORCENTAJE = 5.0: GOTO 70
60: LET PORCENTAJE = 7.5
70: LET DESCUENTO =
    IMPORTE × PORCENTAJE
    / 100
80: PRINT IMPORTE;
    PORCENTAJE; DESCUENTO
90: END
```

Vale la pena estudiar cuidadosamente el programa para ver lo que sucede con cada caso de descuento en particular.



Resulta también más cómodo

Cada línea de un cuadro supone todo un recorrido por el programa. En una variable contadora, que podría llamarse N, se incluye el recuento. Antes de la bifurcación que lleva hacia atrás hay que aumentar en una unidad la variable contadora (LET N = N + 1). Una instrucción IF puede comprobar entonces si se bifurca o no hacia atrás, al compararlo con el valor final M. La ilustración de abajo a la izquierda muestra un diagrama de este tipo para un cuadro de los números cuadrados.
El BASIC tienen una solución más elegante para este problema, la instrucción FOR (=para). TO (=a) lleva al valor final, STEP es un paso. NEXT significa la siguiente variable contadora.

```
10: INPUT M
20: FOR N = 1 TO M STEP 1
30: LET Q = N × N
40: PRINT N; Q
50: NEXT N
60: END
```

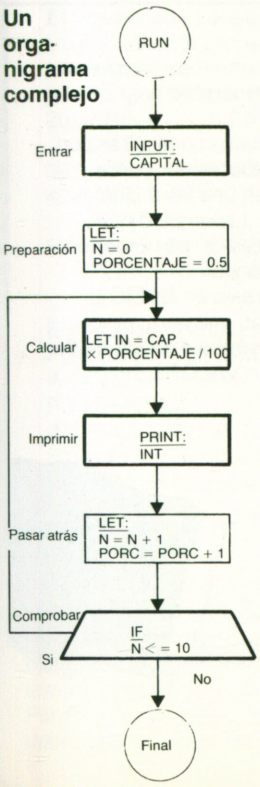
Gramática para BASIC = bifurcaciones

Todo lenguaje tiene sus reglas gramaticales. En el caso de los lenguajes de programación, a esa gramática se la denomina «sintaxis». Si en el programa aparece un error, conviene comprobar primero si no hay errores en la sintaxis.
Damos aquí algunas reglas sintácticas para las instrucciones de bifurcación de BASIC (n es siempre un número de instrucción):

GOTO n	GOTO n
IF condiciones	GOTO n
IF condiciones	THEN n
IF condiciones	LET expresión
IF condiciones	LET expresión :GOTO n

FOR variable contadora = valor inicial
TO valor final
STEP paso
Programa
NEXT variable contadora

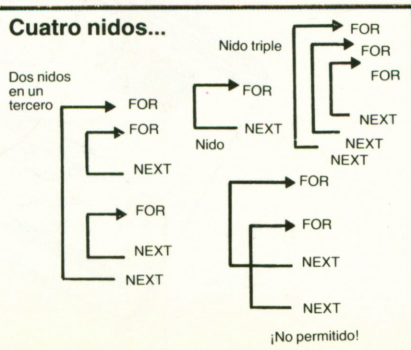
Un organigrama complejo



Comprobación de los programas

Es evidente que al escribir un programa pueden cometerse errores. Estos errores se encuentran por medio de datos de ensayo, que han de construirse de tal manera que recorran todas las posibles ramificaciones del programa. En un sencillo programa de descuentos existen cinco ramas. Por lo tanto hay que dar cinco importes de tal manera que a cada uno de ellos se le adjudique un interés distinto. Muchas veces se necesita un valor intermedio de una posición del programa se puede introducir una instrucción de PRINT, que después se vuelve a retirar.

Cuatro nidos...



... y cómo se originan

Un lazo en el programa se origina cuando se bifurca con GOTO, FOR o IF. Cada par homogéneo de FOR y NEXT (NEXT significa realmente sólo el retroceso a FOR, al que pertenece) constituye un nido. Un nido FORT/NEXT permite albergar otro dentro. El número de los que caben depende del dialecto BASIC. No está permitido que dentro del nido sólo esté el FOR y que el NEXT quede fuera.

Leer y escribir

En las páginas anteriores hemos entrado, procesado y dado salida fundamentalmente a números (ya sean impresos o sobre pantalla). Pero un ordenador también debe poder tratar con letras y textos. Los números representan valores numéricos, ya sean enteros, con decimales o con coma flotante. Con ellos se pueden hacer cálculos. Pero lo mismo que hay números, también hay «literales» formados por textos. El texto puede oscilar entre una breve descripción y el contenido de un libro. Con ellos no se pueden hacer cálculos como con los números, pero el ordenador también los elabora. En relación con la entrada y salida de datos hay algunos comentarios interesantes.

Un número gana cuando se le da un nombre que lo identifica, y esto también resulta útil a la hora de hacer cálculos.
Aprenderemos en estas páginas cómo se hacen las entradas y salidas hablando de datos. Y cómo se establece un diálogo con el ordenador.

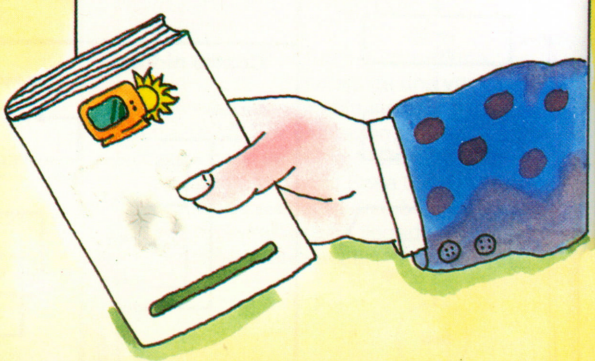
Los literales son textos cortos

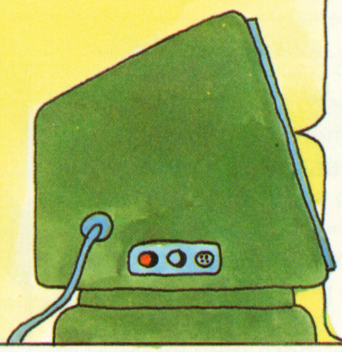
«Literal» es una voz artificial. Quiere decirse con ella una sucesión arbitraria de caracteres del teclado de una máquina de escribir (como la de los microordenadores).
Pueden ser letras pero también cualquier otro símbolo o signo. Para que haya claridad en el ordenador acerca de lo que corresponde a un determinado literal y lo que no, cada uno de ellos se pone entre comillas o comas altas indistintamente. Hay ordenadores que comprenden ambas. Un literal presenta el siguiente aspecto:
«12 es un número»
o bien
«POR FAVOR ENTRAR CAPITAL»

Incluso puede haber una sucesión de cinco espacios: » «.
0 en otro dialecto BASIC ‹ ›.

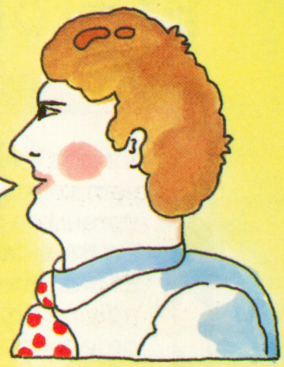
Tratar con nombres

Cuando se trabaja con números y literales, a estos datos se les da un nombre por comodidad.
Los nombres de los números comienzan con una letra. En los ordenadores pequeños consisten únicamente en una letra, que es abreviatura del nombre. La longitud que pueden tener los nombres de los números viene señalada en el manual.
Los nombres de los literales en BASIC comienzan asimismo con una letra, pero acaban siempre con el símbolo del dólar, $. Tenemos así: A$ o Z$, o VALOR$.





El diálogo con el ordenador
Se entiende por diálogo una conversación entre dos interlocutores. En este caso, entre el hombre y el ordenador.
Sin embargo, hay que programárselo previamente pues la máquina por sí sola no puede hacerlo. Es decir, que en las instrucciones de INPUT y de PRINT se incluyen tantos literales explicativos como caben. Sirven para guiar al usuario de modo que pueda trabajar con el ordenador, de manera adecuada y a ser posible sin errores. Si comete una falta, la máquina le ayudará a seguir. Este diálogo «adaptado al usuario» es una ayuda importante en el trato con el ordenador.



Los comentarios
Cuando se bifurca en un programa, suele resultar de utilidad hacer una anotación al IF o al GOTO o dar un comentario al usuario. Sucede así en todos los modernos lenguajes de programación. En BASIC se utiliza para ello la palabra de instrucción REM (abreviatura de REMARK, «observación»). REM tiene un número de programa y aparece con él. A REM le sigue un literal. Por ejemplo:
25 REM «ENTRADA PARA PORCENTAJE»
Sobre la pantalla aparece entonces el literal «ENTRADA PARA PORCENTAJE».

La instrucción INPUT
En nuestros ejemplos siempre la hemos utilizado de manera que a la palabra INPUT sigue una lista de nombres de variables, a las que debe darse entrada. Cada uno de los nombres puede designar números o literales. Los ordenadores sencillos permiten sólo la entrada de un único nombre. Para cada valor solicitado hay que escribir su propio INPUT. El BASIC es más cómodo y permite también añadir literales.
Estos aparecen sobre la pantalla y dicen al usuario no sólo que debe dar entrada a algo, sino también qué es lo que significa ésto.
INPUT «IMPORTE=»; IMPORTE
produce en la pantalla no únicamente un signo de interrogación u otra solicitud igual de lapidaria, sino el texto literal
IMPORTE=
El usuario sabe entonces que debe entrar una cifra de importe.

El interés en el diálogo
Un pequeño programa para el cálculo del interés es este:

```
10: REM «PROGRAMA DEL INTERES»
20: PRINT «POR FAVOR TECLEE SUS
    DATOS»
30: INPUT «IMPORTE = »; IMPORTE
40: INPUT «PORCENTAJE = »;
    PORCENTAJE
50: PRINT «INTERES =»; (IMPORTE ×
    PORCENTAJE/100); «PTS»
    60: END
```

El correspondiente acta en pantalla dice:
POR FAVOR ESCRIBA SUS DATOS
IMPORTE = 12 500
PORCENTAJE = 5
INTERES = 625 PTS
Es un acta correcta que refleja todo lo que ha sucedido en el ordenador.

La instrucción PRINT
Esta instrucción imprime o lleva a la pantalla el resultado (depende del ordenador). Después de PRINT se puede escribir también un literal que represente un texto explicativo. En el caso de
PRINT «POR FAVOR INTRODUCIR IMPORTE»
indica al usuario lo que se espera de él. No hay que ser parco con esta versión de la instrucción PRINT, porque de ella resultan muchos comentarios esclarecedores. Cuando se quiere controlar si un programa pasa en una determinada posición, se envía en la fase de ensayo una instrucción PRINT con el correspondiente texto. Después se puede borrar. A PRINT le puede seguir asimismo una expresión aritmética.
La instrucción
PRINT «INTERES=»; (IMPORTE × PORCENTAJE/100); «PTS»
lleva (en caso de IMPORTE= 12 500 y PORCENTAJE= 5) sobre el papel o a la pantalla:
INTERES= 625 PTS.

La influencia sobre la imagen impresa
En el interior del ordenador los números suelen tener una representación distinta a como se les imprime: por ejemplo, más decimales de los que se necesitan detrás de la coma.
Por eso la instrucción USING indica al ordenador cómo deben representarse los datos.

Si se entra PRINT USING 30 y se ponen después uno o varios nombres de datos, el BASIC esperará encontrar bajo el número de programa 30 una orden detallada de como hay que expresar los datos.

El programa de las adivinanzas

Vamos a reunir ahora en un ejemplo todo lo que hemos aprendido en las últimas páginas. Se trata de adivinar números. El programa le pide a usted que adivine un número que el ordenador ha echado a los dados. (Veremos así, entre otras cosas, que el ordenador puede jugar a los dados.)
Usted puede adivinar ocho veces. El ordenador comentará las entradas que hace cada vez. Le felicitará si acierta y le consolará cuando no lo consiga. Después preguntará amablemente si quiere continuar jugando. En caso afirmativo tirará un nuevo número y así dispondrá usted de otra oportunidad más. Es interesante el azar, al que nos referiremos más adelante, pero lo más interesante son los numerosos comentarios y bifurcaciones del programa.
A todo esto se le llama la «técnica del diálogo».
Intente primero leer el programa, instrucción por instrucción. No es difícil. Con la instrucción 40 se origina el número de azar. Elija un número cualquiera entre 1 y 100 y trabaje con él. (Si su ordenador no tiene número de azar, podrá conseguirlo con un programa que sí lo haga.)

El ordenador jugador
Toda tirada con el dado proporciona una cifra aleatoria, al azar, comprendida entre 1 y 6. Se puede escribir un programa en BASIC que haga lo mismo con las cifras 0 a 9. Si por ejemplo multiplicamos el número π (3.1415965) por sí mismo varias veces, las cifras del centro del número se mezclan de tal modo que se las puede considerar como cifras aleatorias. El modo de sacar las cifras se muestra abajo a la derecha.

La función llamada azar
Muchos ordenadores de BASIC tienen una función de azar.
Se llama RND o de manera similar y proporcionan cada vez un número distinto comprendido entre 0 y 9, como por ejemplo 0,382934802. De aquí se hace una cifra entre 0 y 9 al multiplicar por 10 y eliminar todo lo que queda detrás de la coma. De esto último se encarga la función INT, que es elemento integrante del BASIC.
La instrucción

N = INT(RND × 10)

proporciona cifras al azar que se comportan lo mismo que las caras de un dado. (Multiplicando por 100 se obtienen números enteros comprendidos entre 00 y 99.

Cuando su BASIC no tiene función RND
Lea en la página 20 sobre el modo de construir un subprograma a partir de lo que hasta allí se ha aprendido.

Sacrificamos un número
Del número Q = 0,5837208 hay que eliminar la cifra situada en el punto medio, o sea, el 7. Para ello se multiplica Q por 10 000; la función INT separa entonces la parte entera que hay delante de la coma:

R = INT (Q × 10 000) = 5837

Multiplicamos ahora Q por 1000, volvemos a tomar la parte entera (583) y multiplicamos por 10:

S = INT (Q × 1000) × 10 = 5830

La diferencia R − S da el 7:

```
LET R=INT(Q × 10 000)
LET S=INT(Q × 1000) × 10
LET N = R − S
```

Ahora comentamos el programa
Encontrará aquí una pequeña explicación de cada una de las líneas del programa. Le ayudará en caso de duda. Lea nuestros comentarios únicamente cuando tenga alguna dificultad.
Hágalo más tarde cuando ya entienda el programa, a fin de poder conocer también los detalles.
INSTRUCCION 10: El programa le dice lo que le corresponde a usted.

El programa de las adivinanzas

```
 10: PRINT «ADIVINE UN NUMERO ENTRE 1 Y 100 QUE
     HE PENSADO»
 20: PRINT «Y PUEDE PROBAR A ADIVINARLO 8 VECES»
 30: LET T=0
 40: LET N=INT (RND × 100) + 1
 50: IF T>=8 GOTO 180
 60: LET T=T + 1
 70: PRINT «ADIVINE AHORA»
 80: INPUT G
 90: IF G=N GOTO 150
100: IF G>=N GOTO 130
110: PRINT G; «DEMASIADO BAJO»
120: GOTO 50
130: PRINT G; «DEMASIADO ALTO»
140: GOTO 50
150: PRINT «¡FELICIDADES! LO HA CONSEGUIDO»
160: PRINT «DE TODAS MANERAS TENDRA QUE
     ADIVINAR «;T;»
170: GOTO 190
180: PRINT «LO SIENTO, DE TODAS MANERAS TENIA
     8 POSIBILIDADES»
185: PRINT «DICHO SEA DE PASO, MI NUMERO ERA N»
190: PRINT «¿JUEGA DE NUEVO? 1=SI, 0=NO»
200: INPUT B
210: IF B=0 GOTO 230
220: GOTO 30
230: PRINT «¿NO SE ATREVE MÁS?»
240: STOP
250: END
```

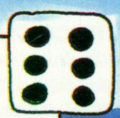
INSTRUCCION 20: El programa le explica una de las reglas del juego.
INSTRUCCION 30: Con el contador T se cuenta hasta 8. Por lo tanto, al comienzo hay que ajustarlo a cero.
INSTRUCCION 40: Aquí se forma el número de azar que usted tiene que adivinar. Para ello, en la mayoría de los ordenadores existe la función RND, o una abreviatura similar (en este caso es de RANDOM «azar»). Genera un número entre 0 y 9, por ejemplo 0,32567. Nuestra instrucción multiplica este número por 100 y hace de él 32,567. La función INT (de INTEGER «número entero») toma de ahí la parte entera situada por delante de la coma. Por consiguiente, en nuestro ejemplo sería N=32

INSTRUCCION 50: El programa comprueba si se han hecho más de ocho ensayos. En caso afirmativo pasa a 180. El jugador no lo ha conseguido.
INSTRUCCION 60: El contador del número de juegos aumenta en 1.
INSTRUCCION 70: Se solicita dar un número de adivinanza.
INSTRUCCION 80: Se solicita el número para adivinar y pasa al programa bajo el nombre G.
INSTRUCCION 90: Ha habido éxito al adivinar: G=N. El programa se bifurca a las felicitaciones.
INSTRUCCION 100: El número que se da es mayor que el que hay que adivinar. Este caso se trata en 130.
INSTRUCCION 110: G no es igual ni mayor que N, por lo que es menor. Esto se comenta.
INSTRUCCION 120: Ya que se ha dado un número demasiado bajo hay que volver al paso anterior, que comienza en 50.
INSTRUCCION 130: Aquí se llega desde la instrucción 100. Allí se establecía que se había dado un valor demasiado alto. Se imprime.
INSTRUCCION 140: Ya que se ha dado un valor demasiado alto hay que volver al paso anterior.
INSTRUCCION 150: Aquí se llega cuando se ha adivinado el número, o sea, cuando G=N, antes de que T sea mayor que 8. Se felicita.
INSTRUCCION 160: Aunque se ha ganado, el programa no puede renunciar a burlarse un poco.
INSTRUCCION 170: El juego se ha terminado. En 190 comienza la pregunta de si se quiere volver a jugar.
INSTRUCCION 180: Aquí se llega desde 50 cuando se ha perdido. El programa hace una observación al respecto.
INSTRUCCION 185: Se dice al perdedor cual es el número que había que adivinar.
INSTRUCCION 190: Esta es una rama del programa en la que se comunica si el ganador (que viene de 170) o el perdedor (que viene de 185) quieren seguir jugando. Se informa que con el 1 se sigue y con el 0 se abandona.
INSTRUCCION 200: Se exige respuesta a la pregunta de 190. Con 1 se continúa, con 0 no.
INSTRUCCION 210: El que, a la pregunta de si quiere continuar jugando, ha respondido «no», ha marcado cero en 200. Pasa entonces a 230.
INSTRUCCION 220: El que, a la pregunta de si quiere continuar jugando ha respondido «sí», ha marcado uno en 200.
Se ha pasado de largo por 210; pasa a 30 para la siguiente ronda.
INSTRUCCION 230: Aquí se viene de 210. No se sigue jugando. El programa no puede renunciar a un comentario.
INSTRUCCION 240: El programa se detiene debido a que ya no hay interés por seguir jugando.
INSTRUCCION 250: Aquí finaliza el programa.

Ahora jugaremos a los dados con un subprograma

No es raro que en un programa extenso se necesite varias veces algo que no está allí: una sucesión de instrucciones o una función. Se aplica entonces un subprograma. Se trata de un trozo de programa como cualquier otro, pero que es solicitado mediante una instrucción. A uno de estos subprogramas no se accede a través de GOTO sino de GOSUB (SUB viene de «subroutine», que significa subrutina o subprograma). A GOSUB sigue el número de la instrucción del subprograma. La última instrucción de éste se llama RETURN («volver»). GOSUB se da cuenta de hacia donde debe volver el subprograma; RETURN le lleva hasta allí. Esto es muy sencillo porque el BASIC lo hace todo automáticamente.

Desarrollaremos en esta páginas un subprograma que produce cifras de azar comprendidas entre 0 y 9. Utilizaremos primero este dado decimal como un dado verdadero para las cifras 1 a 6. Comprobamos la frecuencia con la que aparecen cada una de las cifras al echar 100 veces los dados. Producimos entonces una serie de números de azar de dos cifras comprendidos entre 00 y 99. Con ello hacemos prácticas de programación y aprendemos los diversos trucos.



El dado decimal

Un dado normal de juego tiene seis caras. Uno decimal proporciona los diez números que van del 0 al 9. Su programa ya lo hemos visto en la página 18. Nuestro subprograma del dado tiene algunas adiciones:

```
500: IF (Q = 0,4) GOTO 502
501: LET Q=Q+0,35
502: LET Q=Q×Q
503: LET R=INT(Q+100000)
504: LET S=INT(Q×10000)×10
505: LET N=R−S
506: RETURN
```

El subprograma comienza con el número de instrucción 500. Se le solicita con GOSUB 500. Regresa siempre a la instrucción que sigue a GOSUB. De esto se encarga la instrucción 506: RETURN

El dado electrónico de seis dados

Veremos un programa muy sencillo que produce veinte cifras de azar comprendidas entre 1 y 6. Por consiguiente, nuestro dado decimal se convierte en uno normal. Para ello hacemos un bucle de 1 a 20, conducido por la instrucción GOSUB 500. A cada número de azar se le añade 1 porque con cero hacemos un sí. Lo que es mayor de 6 se elimina. Todo lo demás se representa o imprime. Tenemos así:

```
100: LET Q=0,314159265
110: FOR K=1 TO 20
120: GOSUB 500
130: LET N=N+1
140: IF (N>6) GOTO 120
150: PRINT N
160: NEXT K
170: END
```

Si completa este pequeño programa en un ordenador de BASIC, tendrá la oportunidad de comprobar lo rápido que es.

¿Qué rapidez tiene mi ordenador?

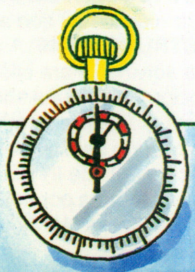
Queremos saber exactamente cuanto tiempo tarda en dar 20 números de azar, comprendidos entre el 1 y el 6, la calculadora de bolsillo BASIC representada abajo a la derecha tiene una instrucción que se llama BEEP. Si antes del programa de ensayo se pone la instrucción

```
99 BEEP 2
```

y detrás de NEXT K la instrucción

```
165 BEEP 5
```

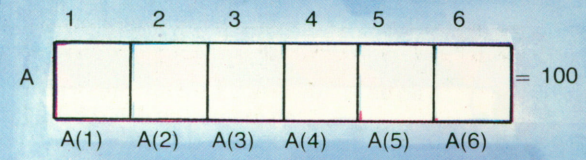
entonces se puede detener el tiempo. La calculadora da dos pitidos cuando comienza y cinco cuando ha acabado. Sin contar el tiempo de los pitidos, necesita 85 segundos, incluida la representación. Son 4,25 segundos por tirada. ¡No está mal para una de las calculadoras BASIC más pequeñas!

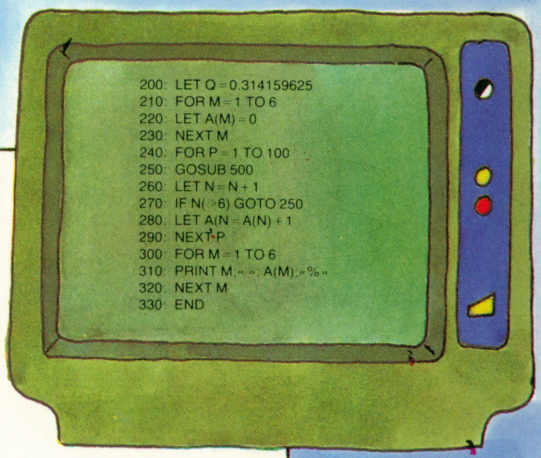


Un pequeño sector de datos

Un «sector de datos» o «vector» es una memoria para muchos datos semejantes bajo un nombre común. Si queremos comprobar si nuestro dado funciona correctamente, debemos contar cuantas veces aparece cada cifra al tirar cien veces los dados. Las diferencias no deberán ser muy grandes.

Para ello tomamos el vector A con las seis direcciones A(1) hasta A(6). Dentro de este contexto, a las direcciones se las llama también «índices». Por lo tanto, con los números 1 a 6 indexamos (asignamos índice) nuestro vector. Esto se explicará detalladamente más adelante. Bajo estas direcciones se cuenta la frecuencia con la que aparece cada número comprendido entre el 1 y el 6, como por ejemplo aquí:





La prueba del azar

Revisaremos cómo nuestro generador de azar produce uniformemente cifras de 1 a 6. Para ello ajustamos primero a cero una zona de datos A(1) a A(6), concretamente por medio de un bucle FOR. En otro bucle, o lazo, FOR producimos 100 cifras de azar comprendidas entre 1 y 6 bajo el nombre M. Con N indexamos (asignamos índices) nuestra zona de datos y añadimos siempre 1. Con un tercer bucle FOR expresamos la cantidad añadida a cada cifra. Ya que hemos hecho cien intentos, estos números son los porcentajes de la aparición de cada cifra entre 1 y 6.

Este es el resultado

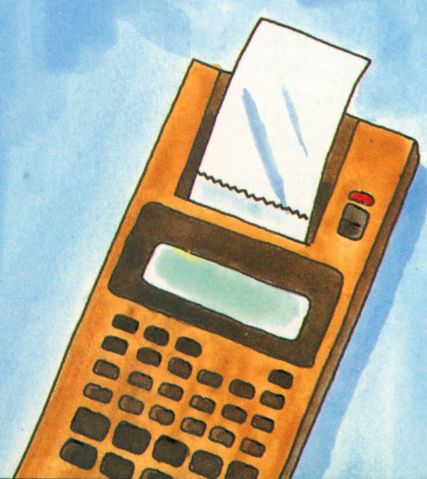
Pocos minutos después de la instrucción RUN 200 nuestra calculadora de bolsillo BASIC da la siguiente estadística, que hemos representado gráficamente arriba:

1. 7.%
2. 22.%
3. 11.%
4. 22.%
5. 17.%
6. 21.%

Determinar si el 1 con su 7% (algo escaso) está bien o no, sólo es posible mediante cálculos matemáticos más complejos. Pero también podemos hacer lo siguiente: si al final no retiramos Q con RUN 210, el generador sigue funcionando. Al cabo de pocos minutos da una nueva tabla, que será distinta: bajo el 1 hay ahora un 18%. Si sumamos esta tabla a la anterior estadística (tercera hilera) y en la cuarta tomamos el valor medio, se obtiene:

1.	18.%	25	12.5%
2.	17.%	39	19.5%
3.	17.%	28	14.0%
4.	15.%	37	18.5%
5.	18.%	35	17.5%
6.	15.%	36	18.0%

La estadística se ha vuelto más aceptable. Y se puede seguir haciendo lo mismo.



El dado de cien

Aquí hay un pequeño programa que hemos de leer. Con el mismo generador (GOSUB 500) produce dos cifras de azar, ambas entre cero y nueve. La una se multiplica por 10 y se suma a la otra. Se obtienen así números entre 00 y 99:

```
350: LET Q=0.314159625
360: FOR M=1 TO 100
370: GOSUB 500
380: K=N
390: GOSUB 500
400: K=10×N+K
410: PRINT M;« »;K
420: NEXT M
430: END
```

El BASIC es muy adecuado para jugar y experimentar.

Las tablas en el programa

Los datos proceden del exterior y son tratados en el ordenador. Los resultados vuelven a salir fuera. Pero existen también trabajos en los que los datos no varían durante mucho tiempo; datos que se arrastran con el programa.
Se les denomina «constantes de programa».
En BASIC, tales datos se pueden arrastrar en el programa en forma de tablas, que pueden ser mixtas, tanto alfabéticas como numéricas.
Para trabajar con tablas fijas existen las instrucciones DATA, READ y RESTORE.
En BASIC son posibles tablas con una o dos filas, que se pueden escribir y leer. Son las llamadas «matrices», o «filas» (en inglés «arrays»). Sin embargo, cuestan mucha memoria. Se las reserva con la instrucción DIM, que designa tanto a la memoria como a la matriz. DIM es abreviatura de la palabra inglesa «dimensión». Si una matriz se llama A y se quiere su tercer número, hay que escribir A(3).
Cuando en el diálogo se toman datos del exterior, se escribe INPUT. Para visualizar se usa PRINT. Si los datos están en casetes o disquetes, se les da entrada en GET y se les vuelve a recoger con PUT.

Un ejemplo sencillo de cuadro
Alguien quiere imprimir un cuadro que tiene una línea para cada mes. Se comienza con una abreviatura para el nombre del mes. Viene después la temperatura media del mes del año anterior, y a continuación un campo vacío en el que puede registrarse una temperatura medida.
Cada una de las instrucciones DATA tiene un número de instrucción y almacena los datos fijos. Se pueden reunir también varios meses en un DATA e incluir en una línea tantos datos como se pueda. De esta manera se ahorra espacio y líneas.

```
201: DATA ENE , 3.4
202: DATA FEB , 5.2
203: DATA MAR , 9.7
204: DATA ABR , 11.3
205: DATA MAY , 15.0
206: DATA JUN , 17.8
207: DATA JUL , 21.2
208: DATA AGO , 24.8
209: DATA SEP , 20.2
210: DATA OCT , 13.9
211: DATA NOV , 7.4
212: DATA DIC , 1.2
```

Se imprime el cuadro
Si en un programa hay varios cuadros de DATA, el BASIC los reúne en el orden de los números de instrucción y los contempla como un cuadro único. Cada READ toma de DATA tantos datos como nombres de estos haya. De esta manera, de un READ a otro READ se apura la reserva de DATA. Sin embargo deben disponerse los DATA de tal modo que tengan tantos datos como haga falta. Por lo tanto, DATA y el correspondiente READ deben armonizarse.
Lo que se imprime es lo siguiente:

```
200: REM «EJEMPLO DE IMPRESION PARA DATA»
201: DATA ENE , 3.4 , FEB , 5.2 , MAR , 9.7
202: DATA ABR , 11.3 , MAY , 15 , JUN , 17.8
203: DATA JUL , 21.2 , AGO , 24.8 , SEP , 20.2
204: DATA OCT , 13.9 , NOV ,7.4 , DIC , 1.2
210: FOR N=1 TO 12
220: READ MES$ , TEMP
230: PRINT USING 231 , MES$ , TEMP
231: « # # # # # . #          +--------+»
240: NEXT N
250: END
```

Así se imprime un cuadro con un único PRINT, aun cuando el FOR genere doce instrucciones de impresión. La sucesión de signos +--------+ sirve para la entrada de la correspondiente temperatura que se ha medido.

Una observación sobre USING

Ya hemos usado anteriormente USING. Indica cómo deben expresarse cada uno de los datos. Sin embargo, este USING ocupa toda una línea. Hemos indicado así la estructura de una línea de la tabla, y hemos determinado la separación entre los datos dentro de la línea. La línea 231 para USING se lee en BASIC de la siguiente manera: imprime tres caracteres tal como vienen (en esta caso son letras), incluye cuatro espacios. Imprime un número de dos cifras como máximo, una coma decimal y una cifra y una cifra detrás. Incluye tres espacios e imprime la sucesión de signos
+ – – – – – – – +.
Más información se puede encontrar en el manual.

¿Qué es RESTORE?

Si en un programa se escribe
290: RESTORE
esta instrucción lleva DATA al comienzo. Puesto que todos los DATA están reunidos, consecuentemente la siguiente instrucción de READ vuelve a comenzar. Si se da a RESTORE un número de instrucción, como por ejemplo 290: RESTORE 203 la siguiente instrucción de READ empieza con el número de programa 203, en nuestro ejemplo será julio.

Sectores de datos, matrices y vectores

Por doquier hay sectores de números que son del mismo tipo, por ejemplo la lista de los descuentos comprendidos entre el 1 y el 10 por ciento. A esto se le llama en matemáticas un vector.
Surgirán dos vectores cuando el descuento dependa de la cantidad de piezas compradas. En la primera hilera estará el número de piezas, y en la segunda el porcentaje de descuento. Sucede lo mismo cuando hay que ver las cantidades que hay que pagar por un seguro para los distintos sueldos y los diversos grupos de edades. A una figura numérica de esta clase la denominan los matemáticos matriz. Diez niveles de sueldo y cinco de edades dan una matriz numérica de 10 por 5, o sea, de 50 números.

Vectores

Se determinan en BASIC mediante un nombre, que es común a todos los números allí dispuestos, y mediante la cantidad de estos. Si se quiere fijar en BASIC un vector de este tipo se escribe
325: DIM DESCUENTO (25)
El BASIC reserva entonces para el programa una memoria para 25 números bajo el nombre de DESCUENTO. Si se escribe
240: DIN DEN$ (125)
se define así un campo de memoria para 125 campos alfabéticos, por ejemplo denominaciones de artículos, bajo el nombre DEN.

La matriz de los seguros

Como ejemplo de matriz teníamos un cuadro en el que de arriba abajo se registraba el sueldo y de izquierda a derecha el grupo de edades, a fin de determinar el importe correspondiente del seguro. Para la definición de este cuadro se escribe
DIM IMPORTE (5,10)
5 es para la edad, 10 para el sueldo.
Con IMPORTE (2,8)
se obtiene el importe del seguro en el grupo 2 de edades con el grupo 8 de sueldos.

Como ejemplo: un programa semanal

```
100: REM «DATA Y DIM PROGRAMA DE
     PRACTICAS»
110: PRINT
     «DIAbbbbbbPLANbbbbbbbbESbbbbPORC»
120: DIM EJERCICIOS (2,7)
130: FOR K=1 TO 7
140: INPUT EJERCICIOS (2,K)
150: READ DIA$, VALOR
160: DATA LU , 100 , MA , 80 ,
     MI , 75 , JU , 90
170: DATA VI , 110 , SA , 120 ,
     DO , 110
180: LET EJERCICIOS (1,K)=VALOR
190: PRINT USING 195 DIA$; VALOR;
     EJERCICIOS(1,K);
     (EJERCICIOS (2,K) – VALOR/
     VALOR × 100
195: «## ### bbb ### ### . ## ## bbb ## ##
     ## ##      . ## ## bbb ## . ## »
200: NEXT K
210: END
```

La explicación

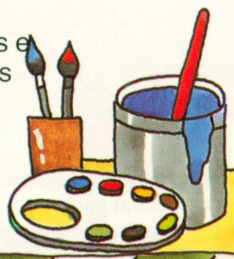
A la izquierda tenemos un programa con DIM, DATA y USING, y a la derecha el correspondiente comentario. Se trata del bolsillo. En una cadena de DATA se almacena en primer lugar una abreviatura para cada día de la semana, y en segundo lugar un importe para cada día, que se quiere gastar según un plan. En una hilera de la matriz se entran a través de INPUT siete importes con el nombre GASTOS. El programa trabaja cada día con un bucle FOR. Entonces se imprime: nombre del día, valor del plan e importe gastado. En la fila de gastos se calculará el porcentaje del plan que se ha gastado.

El comentario

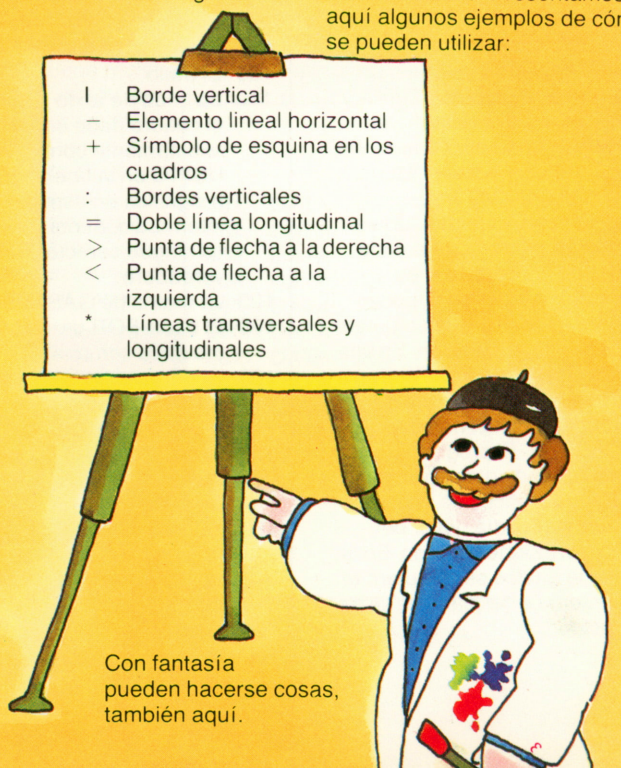
100 El título
110 La línea de texto sobre el cuadro debe coincidir exactamente con USING en la línea 195, b significa un espacio intermedio, aunque no es ningún carácter verdadero.
120 La matriz de GASTOS.
130 El bucle FOR para cada día de la semana.
160 La primera parte de las constantes de programa.
170 La segunda parte de las constantes de programa.
180 Valor de plan de los DATA para la primera fila de la matriz de GASTOS.
190 Se imprime.
200 De vuelta a FOR con el siguiente valor K. O final del bucle cuando K es mayor que 8.

Pintar con puntos y signos

La pantalla y la impresora de un ordenador doméstico trabajan con diminutos puntitos luminosos o impresores. Por eso les da igual lo que muestran: letras, números, signos o dibujos. Básicamente dependen del lenguaje de programación. Por lo tanto sólo pueden representar lo que hay disponible en BASIC. Algunos de los caracteres del BASIC son adecuados para dibujar, o al menos para hacer pequeños cuadros como marco para lo que se imprime. De esta manera también es posible hacer formularios. Pero hay que tener algo de fantasía gráfica. Muchos ordenadores domésticos disponen de caracteres especiales gráficos que se pueden incluir en el BASIC: líneas inclinadas, cuadrados, medios campos y pequeñas figuras geométricas, que pueden agruparse para dar imágenes.
Los microordenadores más grandes y los ordenadores personales tienen en su BASIC algunas instrucciones con las que es posible acercarse a cualquier punto de la pantalla o del papel de la impresora, y en el caso de la pantalla además en colores. Se pueden producir entonces imágenes sorprendentes e incluso dibujos técnicos.

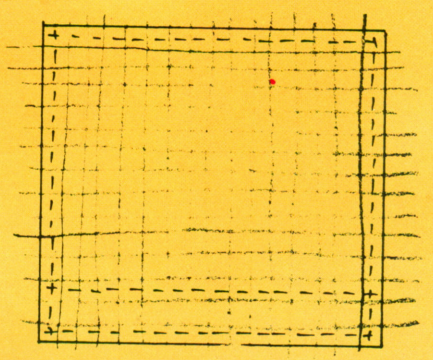


Símbolo a símbolo
Incluso en un conjunto estándar del BASIC existen algunos símbolos, o caracteres, con los que se puede dibujar modestamente. Presentamos aquí algunos ejemplos de cómo se pueden utilizar:



Con fantasía pueden hacerse cosas, también aquí.

La tarjeta de visita
Cuando se quieren hacer imágenes con caracteres BASIC con ayuda de la impresora o de la pantalla, hace falta un croquis primero. Los caracteres deben coincidir con él.
Se toma un papel cuadriculado y se marca el campo disponible o en el que se quiere imprimir. La tarjeta de visita la hemos impreso con un pequeño ordenador doméstico. Tiene transversalmente 16 espacios disponibles. Nuestra imagen de abajo muestra el aspecto del croquis.



El programa de la tarjeta de visita
Si se quiere imprimir después, para cada línea necesitamos un PRINT. Quien quiera teclear mucho, pondrá unas comillas («) antes y después de cada línea, y delante un número y la palabra PRINT. Esto nos da 15 instrucciones PRINT y un END. Pero también puede hacerse con dos subprogramas, con un PRINT y un RETURN cada uno.

Sin subprograma

```
100 PRINT "+----------+"
110 PRINT "I          I"
120 PRINT "I          I"
130 PRINT "I          I"
140 PRINT "I          I"
150 PRINT "I          I"
160 PRINT "I          I"
170 PRINT "I          I"
180 PRINT "I          I"
190 PRINT "I          I"
200 PRINT "I          I"
210 PRINT "I          I"
220 PRINT "+----------+"
230 PRINT "I TEL 379318 I"
240 PRINT "+----------+"
250 END
```

Con subprograma

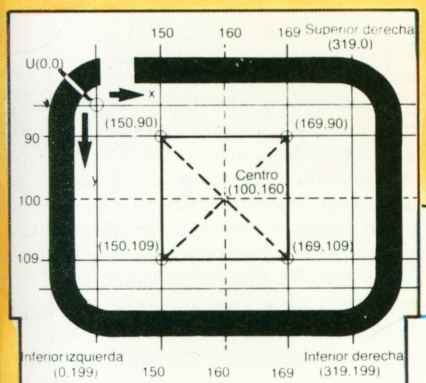
```
100 GOSUB 300
110 GOSUB 320
120 GOSUB 320
130 PRINT «I RUDOLF I»
140 PRINT «I SONDERMAIER I»
150 GOSUB 320
160 PRINT «I STUTTGART I»
170 GOSUB 320
180 PRINT «I KOENIGSSTR I»
190 GOSUB 320
200 PRINT «I    18    I»
210 GOSUB 320
220 GOSUB 300
230 PRINT «I TEL 379318 I»
240 GOSUB 300
250 END
300 PRINT «+----------+»
310 RETURN
320 PRINT «I          I»
330 RETURN
340 END
```

Caracteres gráficos especiales
Muchos ordenadores domésticos tienen caracteres gráficos especiales que se pueden pasar a la impresora mediante BASIC. Sin embargo, por regla general es necesario adaptar el teclado para que el ordenador sepa la versión que es válida. Para hacer estos trabajos es imprescindible estudiar el manual.

La pantalla y sus puntos
Una pantalla puede tener por ejemplo 25 líneas con 80 caracteres cada una de ellas. Esto hacen en total 2 000 caracteres, que es lo que cabe más o menos en una hoja de papel en formato DIN A-4. Los caracteres están formados por puntos aislados. Visto así, nuestra pantalla tiene 200 líneas y cada una 320 puntos. En total esto serían 64 000 puntos luminosos, que se pueden iluminar o no. Existen asimismo pantallas que en la misma superficie tienen 128 000 puntos, 200 líneas con 640 puntos luminosos cada una. La imagen que dan es más precisa. Para las aplicaciones gráficas se tiene acceso a cada uno de los puntos individualizados.

También la impresora tiene puntos
Una impresora de agujas tiene para cada carácter 7 por 7 agujas, que equivalen más o menos a los puntos de una pantalla. Cada una de ellas genera un punto fino. Se forma un carácter o signo cuando se pulsan todas las agujas que le corresponden.

Así se accede a los puntos
Cada punto de la pantalla se abarca en BASIC mediante sus coordenadas. Su posición en la línea se designa con X y el número de la línea con Y. El punto situado en el centro tiene las coordenadas M(159,99).



Dibujamos un cuadrado
A la izquierda puede ver un esquema de la manera de dibujar un cuadrado con sus diagonales. Se pueden leer en la imagen los valores de coordenadas de cada punto. Dibujamos de izquierda a derecha y de arriba abajo, o sea, desde el punto SI(150,90) al punto SD(169,90) e igualmente del punto II(150,109) al punto ID(169,109). Las diagonales van desde SI hasta ID y desde II hasta SD. Aquí tenemos el programa con sus bucles FOR:

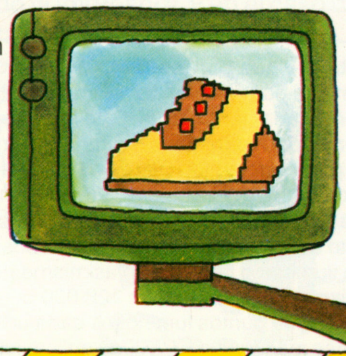
```
100: CLS
110: FOR N=150 TO 169
120: POINT(N,90) De (150,90) a (169,90)
130: POINT (N,109) De (150,109) a (169,109)
140: NEXT N
150: FOR N=90 TO 109
160: POINT (150,N) De (150,90) a (150,109)
170: POINT (169,N) De (169,90) a (169,109)
180: NEXT N
200: FOR N=0 TO 20
210: LET K=150+N
220: LET L=90+N
230: LET M=109-N
240: POINT (K,L) De (150,90) a (169,109)
250: POINT (K,M) De (150,109) a (169,90)
260: NEXT N
270: END
```

Los puntos en BASIC
Al tratar con puntos, cada dialecto BASIC tiene sus propias maneras. Veremos una instrucción artificial que conecta y desconecta los puntos. Su aspecto es:
300: POINT (x,y);F
Si F=0, se desconecta el punto con las coordenadas x e y. Si falta F o tiene el valor 1, se conecta. Se dan las coordenadas x e y, y después la instrucción POINT.

La pantalla de colores
Toda pantalla tiene dos colores elementales: el fondo y el primer plano. Sólo este último es el que se ilumina. F=0 significa fondo, F=1 o nada, F, primer plano. Existen otros valores para F si son necesarios más colores. Hay también instrucciones con las que se puede dar color a toda la pantalla o a determinadas zonas. Sobre estas cuestiones es necesario leer el manual.

La gran excursión al azar

Encontrará aquí un ejemplo de cómo se unen el azar del ordenador con el grafismo de la pantalla. Se trata de una excursión al azar. Tenemos un excursionista que siempre que ha dado un paso no sabe hacia donde dará el siguiente. Hay ocho posibilidades en ocho direcciones. El guía es el generador de azar, que da una cifra comprendida entre 0 y 7. A este desplazamiento errático se le da en física el nombre de «movimiento molecular browniano» porque las moléculas se comportan de un modo similar. También las personas, cuando han bebido una copa de más, se mueven muchas veces así.
En el programa se hacen múltiples bifurcaciones. Utilizaremos también muchas de las cosas que hemos aprendido en las páginas precedentes. Si se quiere enfocar el asunto profesionalmente, primero hay que leer el programa y entenderlo. Es como un rompecabezas, aunque se aprenden muchas cosas al hacerlo. Además, sirven de ayuda los comentarios que se incluyen en esta página.

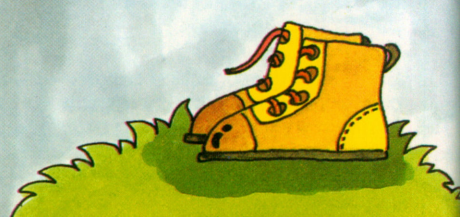


El azar en el ordenador
Ya hemos visto en la página 21 la manera de regir sobre el azar. Se multiplica por sí mismo constantemente un número de partida Q. En cada multiplicación, las cifras situadas en la parte central del producto de este número se mezclan al azar igual que sucede al tirar los dados. Esto ya sirve. El programa correspondiente está a la derecha.
La mejor manera de comprenderlo es hacer como si uno mismo fuera el ordenador e hiciera cada una de las instrucciones.
Un generador de azar empieza siempre por el mismo número Q y produce siempre la misma sucesión aleatoria. (Esto sería una especie de azar previsible). Por esa razón, en nuestro programa damos inicialmente el primer número que se nos ocurre. De esta manera se obtiene una sucesión distinta a la anterior.

El subprograma de azar
Aquí tenemos de nuevo nuestro subprograma para el dado del ordenador.
Proporciona una de las diez cifras comprendidas entre 0 y 9. Se le solicita por medio de GOSUB 800
En N hay disponible entonces una cifra aleatoria.
Si, como en nuestro programa, únicamente queremos los ocho sucesos de 0 a 7, se necesita un IF que prescinda de las cifras 8 y 9. Este es GOSUB 800:

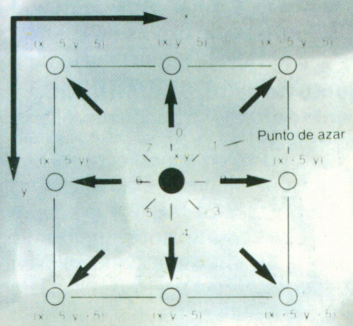
```
800: IF (Q>=0.4) GOTO
     802
801: LET Q=Q+0.35
802: LET Q=Q×Q
803: LET
     R=INT(Q×100000)
804: LET S=INT(Q×10000)
     × 10
805: LET N=R−S
806: IF N > 7 GOTO 800
807: RETURN
```

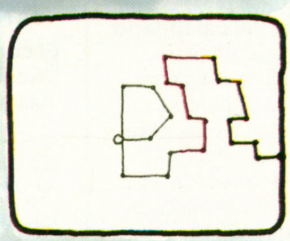
El paseo al azar
Nuestro caminante avanza a grandes pasos por la pantalla, es decir, que cada vez recorre varios puntos. Cuando llega a uno cualquiera, de coordenadas **x** e **y**, el programa baraja ocho cifras entre 0 y 7. El caminante tiene ocho posibilidades. Si sale el 0, avanza cinco puntos perpendicularmente hacia arriba. La cifra 2 le lleva cinco puntos hacia la derecha. El 1 le lleva en diagonal hacia arriba a la derecha, y así sucesivamente. La imagen de la derecha nos lo muestra.
Nuestro programa tiene que dibujar sobre la pantalla el camino recorrido en forma de una línea, o sea, iluminar los puntos que se encuentran en ese camino. Lo haremos con un bucle FOR con cinco pasos (FOR K=1 TO 5). Todo esto nos dará el nuevo punto (x,y), a partir del cual podemos echar de nuevo a los dados.



Los comentarios al programa

A(100-120)	Se ajusta el generador de azar.
B(130-140)	La excursión comienza en el centro de la pantalla, en el punto x = 160, y = 100
C(160-180)	Cuando la excursión llega al borde de la pantalla, se detiene. El programa se acaba entonces.
D(190)	Pasa al subprograma para echar a los dados.
E(200-290)	Para cada cifra de azar, aleatoria, comprendida entre N = 0 y N = 7 se calculan las coordenadas del punto que hay que relevar.
F(0)(300-330)	Se dibuja la línea que une el punto antiguo y el nuevo. Para cada cifra aleatoria N hay un pequeño programa de dibujo.
F(1)-F(7) (350-680)	Los pequeños programas para cada uno de los pasos.
G(700-720)	Las coordenadas calculadas se convierten en nuevas coordenadas de partida para la siguiente tirada de los dados.
I(750-760)	El programa acaba con un comentario.
K(800-807)	Aquí se producen las cifras aleatorias, en un subprograma que se ha iniciado a partir de 190 con GOSUB 800.





El programa

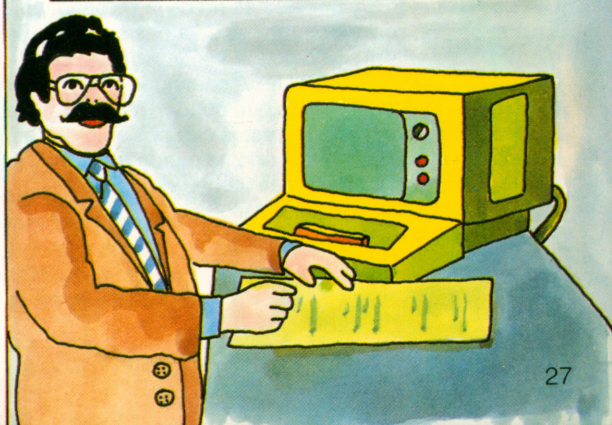
```
100 PRINT «DE UN NÚMERO ENTRE 0 Y 0.3»
110 Q=0.3141592654
115 INPUT R
120 LET Q=Q+R
130 LET X=160
140 LET Y=100
150 IF X>315 GOTO 750
160 IF X<5   GOTO 750
170 IF Y>195 GOTO 750
180 IF Y<5   GOTO 750
190 GOSUB 800
200 IF N=0 THEN LET X1=X, Y1=Y-5, GOTO 300
210 IF N=1 THEN LET X1=X+5, Y1=Y-5, GOTO 350
220 IF N=2 THEN LET X1=X+5, Y1=Y, GOTO 400
230 IF N=3 THEN LET X1=X+5, Y1=Y+5, GOTO 450
240 IF N=4 THEN LET X1=X, Y1=Y+5, GOTO 500
250 IF N=5 THEN LET X1=X-5, Y1=Y+5, GOTO 550
260 IF N=6 THEN LET X1=X-5, Y1=Y, GOTO 600
270 LET X1=X-5
280 LET Y1=Y-5
290 GOTO 650
300 FOR K=1 TO 5
310 POINT (X,Y-K)
320 NEXT K
330 GOTO 700
350 FOR K=1 TO 5
360 POINT (X+K, Y-K)
370 NEXT K
380 GOTO 700
400 FOR K=1 TO 5
410 POINT (X+K,Y)
420 NEXT K
430 GOTO 700
450 FOR K=1 TO 5
460 POINT (X+K, Y+K)
470 NEXT K
480 GOTO 700
500 FOR K=1 TO 5
510 POINT (X,Y+K)
520 NEXT K
530 GOTO 700
550 FOR K=1 TO 5
560 POINT (X-K, Y+K)
570 NEXT K
580 GOTO 700
600 FOR K=1 TO 5
610 POINT (X-K,Y)
620 NEXT K
630 GOTO 700
650 FOR K=1 TO 5
660 POINT (X-K, Y-K)
670 NEXT K
680 GOTO 700
700 X=X1
710 Y=Y1
720 GOTO 150
750 PRINT «SE HA ACABADO LA EXCURSIÓN»
760 END
800 IF (Q>=4) GOTO 802
801 LET Q=Q+.35
802 LET Q=Q×Q
803 LET R=INT(Q×100000)
804 LET S=INT(Q×10000)×10
805 LET N=R-S
806 IF N>7 GOTO 800
807 RETURN
```

Una observación final

Deliberadamente, hemos dicho pocas cosas acerca de nuestro paseo errático a fin de que el amigo lector tenga oportunidad de ocuparse por sí mismo de los detalles. Si dispone usted de su propio ordenador BASIC, lo mejor es que haga en él el programa.

Puede ser también muy útil transformar el programa en un organigrama, tal como lo hemos visto en la página 15.

Si cree además que este problema de azar ha sido programado de manera incompleta, entonces se encuentra usted donde quisiéramos que estuviera. Allí donde usted mismo comienza a crear un programa.



¿Qué se hace cuando hay muchos datos?

La instrucción INPUT facilita al programa datos aislados. PRINT los imprime o los lleva sobre la pantalla. DATA proporciona constantes de programa fijas. Con READ se las da entrada en el curso del programa, y con DIM se definen cuadros y tablas que se pueden llenar y leer. Para estructurar la entrada y la salida de grandes cantidades de datos, BASIC dispone de la instrucción de entrada GET y de la de salida PUT. Con las variaciones pertinentes, son válidas para cassettes y disquetes que puedan acoger cantidades tan grandes de datos. Aprenderemos en estas páginas cómo tratar con esas cantidades. De todas maneras, conviene tener en cuenta que existen considerables diferencias entre los distintos ordenadores de BASIC y sus dialectos. También en este caso, los ordenadores de mayor tamaño ofrecen más que los pequeños, que sólo con dificultad pueden asimilar lo que hace una grabadora de cassette.

¿Qué es un conjunto de datos?
En el lenguaje normal, la frase consta de un sujeto y un complemento. Algo análogo sucede en los ordenadores con los conjuntos de datos, que están formados por varios «campos de datos».
El sujeto es la «clave» que describe un objeto. A esto van unidos los «campos de valores» que hacen una afirmación determinada sobre el objeto, o sea, que son un complemento. Un conjunto de datos en un ordenador está formado siempre por varios campos de datos; unos son las claves y los otros son los valores.

Fechas del calendario
Si configuramos un conjunto de datos para cada día, tendremos el «calendario». La clave para él es la fecha en la sucesión año, mes y día. El 16 de marzo de 1984 es 840316. (Mediante esta representación, el último día del año tiene siempre el número más alto, el máximo «valor de clave»). Como suele decirse, el inventario es «creciente». Además, almacenamos MES y DIA como números y también el día de la semana, en texto completo. En otro campo con 15 caracteres dejamos espacio para una efemérides general como puede ser DOMINGO DE PASCUA. Este campo se llama EFEMERIDE. A continuación vienen cinco campos, PLAZO 1 a PLAZO 5, en los que pueden registrarse acontecimientos individuales. Al principio, en estos campos sólo hay espacios en blanco. En el transcurso del año se puede registrar y borrar lo que se quiera. En un programa de cuestiones imprimimos en etiquetas diarias lo que se almacena y solicitamos las entradas.

El conjunto de datos CALENDARIO
Un conjunto de datos se puede representar más o menos así:

6	FECHA	2	4	0	4	2	2														
2	MES	0	4																		
2	DIA	2	2																		
10	N. DEL DIA	D	O	M	I	N	G	O													
15	EFEMERIDE	D	O	M	I	N	G	O		P	A	S	C	U	A						
15	PLAZO	1	9	O	O	R	E	C	O	J	O	A	B	U	E	L	A				
15	PLAZO	1	3	O	O		C	O	M	I	D	A									
15	PLAZO	1	7	O	O		C	O	N	C	I	E	R	T	O						
15	PLAZO																				
15	PLAZO																				
CAMPO	← CONJUNTO DE DATOS →																				

Se dibuja un esquema en el que pueden registrarse los nombres de los distintos campos. Se indica además la longitud máxima que pueden tener. El conjunto de datos que se puede almacenar tiene una longitud igual a la de todos los campos juntos. En nuestro caso, será de 110 caracteres. De esta manera, para el año bisiesto 1984 tenemos exactamente 366 × 110 caracteres, que hacen 40 260. Todo esto cabe en un disquete.
Pero no se almacenan también los nombres de los campos, sino únicamente su contenido. Por regla general, la clave va delante. Sirve para poder encontrar cada uno de los conjuntos.

PUT o escribir

Para fijar un inventario en un disquete o un cassette, se tiene la instrucción PUT. Escribe en el disquete o cassette una frase completa de la memoria central. Para el programa, cada inventario recibe un número, de FL1 a FL9. De esta manera, se pueden tratar al mismo tiempo como máximo nueve inventarios. Un microordenador apenas tiene tantos aparatos de almacenamiento. Se escribe 250: PUT FL3, CAMPO 1, CAMPO2, CAMPO3,...
Después del número de instrucción y de PUT vienen el número de inventario y después los campos de la frase, o conjunto, en el orden correcto. Los campos deben tener datos definidos. Los campos de cifras y de letras pueden aparecer mezclados.

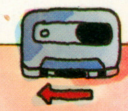
GET o leer

La memoria central «lee», frase tras frase, mediante la instrucción GET un inventario cualquiera que se haya escrito con PUT en un disquete o en una cassette. Cada inventario tiene dentro del programa un número entre FL1 y FL9 al que va unido.
Se escribe:
210: GET FL3, CAMPO1, CAMPO2, CAMPO3 ...

Después del número de instrucción y de su nombre GET aparece el número de inventario, y a continuación vienen los números de los campos. Pueden surgir problemas cuando se quiere devolver a su sitio un conjunto, o frase, recién leído, especialmente en el caso de los aparatos sencillos y con los cassettes.
Es recomendable leer de una cassette y escribir en otra que esté virgen.

RESET o reajustar

En nuestro ejemplo de programa tenemos bajo el número 200 la instrucción
200: RESET FL1
En el primer recorrido, el programa tiene para cada día un conjunto de datos, pero sin nombre para cada uno de los días de la semana. Esto se realiza a partir de la instrucción 200.
RESET sustituye a tener que dar un nuevo OPEN. En muchos ordenadores se encarga también de rebobinar la cassette.



El programa para nuestro calendario está en la página siguiente

La lectura y escritura de inventarios

No se preocupe si en su BASIC todo resulta un poco distinto a como lo describimos aquí. Se debe a que hay diversos dialectos de BASIC. El trabajo con los inventarios no es un problema para principiantes por mucho que el BASIC lo simplifique. Mostraremos aquí sólo los principios del procedimiento y además con un inventario que se escribe frase a frase, y también se lee frase a frase. A estos inventarios se les llama «secuenciales».

OPEN o abrir

Todo inventario que se quiera leer o escribir, debe abrirse primero. Esto se hace con la instrucción OPEN, que va delante de la primera instrucción de lectura o de escritura. Después de OPEN debe estar el número de inventario (entre FL1 y FL9), tras él el número de aparato (que se indica en el manual), más tarde el nombre para el inventario y por último IN o OUT. Se pone IN («hacia dentro») cuando hay que escribir y OUT («hacia fuera») cuando hay que leer.

CLOSE o cerrar

Antes de finalizar un programa, en muchos sistemas hay que dar la instrucción CLOSE. Con ello se cierra técnicamente el inventario. Si no se hace así, se corre el riesgo de que en el siguiente trabajo aparezcan problemas con el soporte de datos. Nuestro programa tiene la instrucción:
340: CLOSE FL1
Es suficiente con el número de inventario. En todos estos casos, siempre debe leerse el manual.

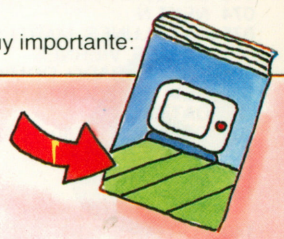
¿Qué es una clave?

Para nuestro inventario CALENDARIO la fecha es la clave de acceso. A través de ella se reúnen los conjuntos de datos de distintos inventarios.
A menudo se utiliza la misma clave en varios inventarios de un banco de datos. En la libreta de ahorro está el número de cuenta, lo mismo que en una bancaria. De esta manera se sabe qué libreta corresponde a qué cuenta corriente.

WRITE y otros nombres

Muchos ordenadores tienen suficiente con PUT y GET en su BASIC para disquetes y cassettes. Pero existen dialectos en los que con PUT y GET sólo pueden tratarse cassettes. Los disquetes tienen entonces nombres de instrucción propios, como por ejemplo WRITE e INPUT. Todo esto viene en el manual. No se desconcierte si todo el asunto parece un poco complicado. En realidad no lo es tanto. Solamente hay que practicar un poco, acumular experiencia y leer el manual. Es frecuente también que varios aficionados a los ordenadores se reúnan para compartir las dificultades, y en los clubs de BASIC hay especialistas en inventarios.

Muy importante:



¡Estudiar constantemente el manual!

Todavía más: el calendario

Abajo tenemos, entre las instrucciones 010 y 350, un programa en BASIC que le recomendamos que lea atentamente. Alcanza hasta 065 preparaciones a fin de poder construir un conjunto de datos nuevo para cada día. Entre 070 y 076 y a partir de dos órdenes DATA se crean en una tabla los nombres de los días de la semana. Esto lo necesitaremos más adelante. Después se pasa por los doce meses del año tras haber abierto primero el inventario con el número FL1. Para cada mes se toma el número del día (TZ) a partir de varias entradas de DATA; para cada mes se pone también un bucle FOR. En 150 se fabrica la clave en la que se aunan el año 84, el MES y el DIA en un mismo número de clave. Se hace con un sencillo truco de adición. Más tarde se escribe el JUEGO DE DATOS con PUT.
En total hay que dar 366 juegos con 110 caracteres cada uno de ellos. En 200 se reajusta (RESET). El 1 de enero es un domingo. Se da entrada a este nombre.
A continuación vienen 52 bucles FOR con siete pasos cada uno, en los que se colocan los nombres de los días.

El programa del calendario

```
010 REM «ABRIR INVENTARIO CALENDARIO»
020 DIM SEMANA(7)
030 LET VACIO$ = «bb/bbbbbbbbbbbbbb»
040 LET NOMBRE DEL DIA$ = «bbbbbbbbbbb»
050 LET EFEMERIDE$ = VACIO$
061 LET PLAZO 1$ = VACIO$
062 LET PLAZO 2$ = VACIO$
063 LET PLAZO 3$ = VACIO$
064 LET PLAZO 4$ = VACIO$
065 LET PLAZO 5$ = VACIO$
070 REM «TABLA DE LOS DIAS DE LA SEMANA
    DE LOS DATA»
071 FOR N = 1 TO 7
072 READ W$
073 LET SEMANA$(N) = W($)
074 NEXT N
075 DATA LUNES, MARTES, MIERCOLES,
    JUEVES, VIERNES
076 DATA SÁBADO, DOMINGO
099 REM «SE ESCRIBEN LOS CONJUNTOS DE
    DATOS»
100 OPEN FL 1, «E80», CALENDARIO, OUT
110 FOR MES = 1 TO 12
120 READ TZ
130 DATA
    31,29,31,30,31,30,31,31,30,31,30,31
140 FOR DIA = 1 TO TZ
150 LET FECHA =
    840000 + DIA + (MES × 100)
160 PUT FL1, FECHA, MES, DIA, NOMBRE DEL
    DIA$, EFEMERIDES, PLAZO 1$, PLAZO 2$,
```

El programa continúa en la página 31, abajo a la derecha.

Imprimimos la hoja de un calendario

El programa de la derecha imprime una hoja de calendario. Primero exige hasta la línea 030 una fecha que ha de buscarse. El programa de búsqueda es un bucle FOR con 366 recorridos, en caso de que la fecha no se encuentre hasta el final como puede suceder si hay un error en la entrada. La instrucción 060 cada vez que es solicitada lee un conjunto de datos. Si la fecha buscada y la del conjunto coinciden, hemos acertado. Irá hasta 200 para ser impresa. A partir de la instrucción 100 comienza la parte de programa que nos comunica que a la fecha ofrecida no le corresponde ningún conjunto de datos. Devolvemos el inventario y preguntamos si se ofrecerá otra fecha, como por ejemplo una corregida. Si no es así, el programa finaliza con CLOSE y un END. El caso en el que se acierta comienza con 200. Se imprime el contenido del conjunto de datos y el programa finaliza con CLOSE/END.

El programa de impresión

```
010 REM «IMPRESIÓN DE UNA
    HOJA DE CALENDARIO»
020 OPEN FL 1, «E80»,
    CALENDARIO, IN
030 PRINT «POR FAVOR DAR
    FECHA: A-M-D»
040 INPUT BUSC
050 FOR N = 1 TO 366
060 GET FL1, FECHA, MES, DIA,
    NOMBRE DEL DIA$,
    EFEMERIDE$, PLAZO1$,
    PLAZO2$, PLAZO3$,
    PLAZO4$, PLAZO5$
070 IF BUSC = FECHA, GOTO 200
080 IF BUSC > FECHA, GOTO 100
090 NEXT N
100 PRINT «FECHAb»; BUSC; «b
    DESCONOCIDO»
110 RESET F 1
120 PRINT «¿LO INTENTA OTRA
    VEZ?»
130 INPUT RESP$
140 IF RESP$ = «SI» GOTO 030
150 CLOSE FL 1
160 END
200 PRINT «SERVICIO DE
    CALENDARIO DE
    ORDENADOR»
210 PRINT USING 220, NOMBRE
    DEL DIA$, DIA, MES
220 :## ## ## ## ## ## ## ## ## ##b
    bb## ##. bb## ##. bb 1984
230 PRINT EFEMERIDE
240 PRINT PLAZO1$
250 PRINT PLAZO2$
260 PRINT PLAZO3$
270 PRINT PLAZO4$
280 PRINT PLAZO5$
280 CLOSE FL 1
290 END
```

¿Qué hacer con el programa?
Una vez que el programa está completo y se sabe que es correcto, ¿qué hay que hacer entonces? Si no se quiere quitar sitio para el almacenamiento de otros programas, hay que guardarlo en cassettes o disquetes, como un conjunto de datos de mayor tamaño. Casi todos los ordenadores pueden hacerlo. El BASIC tiene mandatos especiales para ello: SAVE y LOAD. Si se tienen muchos programas y un buen ordenador, se puede montar una biblioteca de programas en la que se toma sólo lo que se necesita. También se pueden utilizar programas prestados. Lo importante es que, lo mismo que hacen los profesionales, tengamos una documentación de cada uno de ellos en la que aparezca todo lo que es importante, a fin de poder más tarde utilizar y en caso necesario modificar el programa. Una de las cosas es un prueba impresa del programa.

Los programas son costosos
La mayoría de los programas conllevan tiempo, esfuerzos y mucha experiencia. Por esa razón se tratan con cuidado los programas que funcionan, y no es de extrañar que mucha gente pida dinero por ello. Al comprar programas hay que estar atentos a que no nos hagan una chapuza. Un programa únicamente debe comprarse cuando está perfectamente escrito, almacenado y documentado. Debe llevar una denominación, una pequeña descripción de lo que hace y qué peculiaridades tiene, y datos de ensayo con los que se puede comprobar si se adapta al ordenador que tenemos. Hay que poner también cuidado para que no nos den un dialecto que nuestro ordenador no entiende. Y antes de pagar, hay que hacer siempre una prueba de funcionamiento.

LOAD o «cargar el programa»
Lo que GET hace con los archivos lo hace LOAD para los programas que deben sacarse de un disquete o una cassette. Para utilizar LOAD o el mando de SAVE hay que haber leído el manual. Para LOAD hay que indicar el nombre del programa y muchas veces también el del aparato de almacenamiento. Una sencilla versión de LOAD con un interesante complemento es:

LOAD CALENDARIO, R

En este caso se lee un programa llamado CALENDARIO y se pone inmediatamente a funcionar; R significa RUN. Le deseamos que su ordenador de BASIC tenga un LOAD de manejo cómodo. Aunque esto no está siempre garantizado, en particular en los ordenadores más antiguos.

Bibliotecas de programas
A todo buen ordenador le corresponde una biblioteca de programas: disquetes o cassettes, en los que se registran. Se transmiten con una instrucción de carga (como LOAD) y se les puede utilizar directamente con RUN. Los sistemas perfeccionados de BASIC brindan la posibilidad que un programa inicie al final otro de continuación, que siga el trabajo. De esta manera, hay paquetes completos de programas.

SAVE o «guardar el programa»
Los que PUT es para los conjuntos de datos lo es SAVE para los programas que deben almacenarse en un biblioteca de programas: el regrabado desde la memoria central a disquete o cassette. Los sistemas complejos llevan una lista para cada soporte de datos en el que hay programas. Se la denomina «directorio», como un anglicismo de «directory» que significa diccionario. Bajo el nombre de cada programa se almacenan todos los datos que el ordenador necesita para almacenar los programas (SAVE) y para llevarlos a la memoria central (LOAD). Con SAVE y LOAD puede verse el grado de comodidad que presenta el ordenador para el usuario.

RUN o inicio del programa
Como recordatorio: un programa se inicia con el mandato RUN. A RUN sigue el nombre del programa o el número de instrucción, a partir del cual debe comenzarse. De lo contrario, el programa comienza con la instrucción que lleva el número más bajo.

LIST o «hacer una lista del programa»
Como recordatorio: con el mandato LIST se lleva a la pantalla un programa que está en la memoria central para estudiarlo, o a la impresora con el fin de, por ejemplo, incorporarlo a la documentación de programas. A LIST sigue un nombre de programa o un número de instrucción a partir del cual debe hacerse el listado.

Continuación del programa de la página 30

```
        PLAZO3$, PLAZO4$,
        PLAZO5$
170     NEXT DIA
180     NEXT MES
199     REM «PONER NOMBRE
        DEL DIA DE LA SEMANA»
200     RESET FL 1
210     GET FL 1, FECHA, MES,
        DIA, NOMBRE DEL DIA$,
        EFEMERIDE$, PLAZO1$,
        PLAZO2$, PLAZO3$,
        PLAZO4$, PLAZO5$
220     LET NOMBRE DEL
        DIA$ = «DOMINGO bbb»
230     PUT FL 1, ------ Nombre dado
        -------
240     FOR N = 1 TO 52
250     FOR M = 1 TO 7
260     GET FL 1, -------- nombre
        dado ------
270     LET NOMBRE DEL
        DIA$ = SEMANA$(M)
280     PUT FL 1, --------- nombre
        dado ------
290     NEXT M
300     NEXT N
310     GET FL 1, --------- nombre
        dado ------
320     LET NOMBRE DEL
        DIA$ = «LUNESbbbbb»
330     PUT FL 1, --------- nombre
        dado ------
340     CLOSE FL 1
350     END
```

Problemas al margen

El BASIC es, en primer lugar, un lenguaje de programación. Sirve para formular programas, pero los modernos ordenadores que conocen BASIC están construidos de tal manera que también realizan BASIC de modo inmediato, en diálogo con el usuario. Por eso el BASIC es la vía directa al ordenador, y a menudo se dice también que es un «sistema» y no sólo un lenguaje. Esto resulta especialmente claro con los mandatos. En la práctica no solamente hay que saber lo bueno que es el BASIC y las características del dialecto que conoce el ordenador, sino que hay que averiguar asimismo el modo de cooperación del BASIC y del ordenador. Lo mejor es sentarse delante de éste, elaborar un programa desde el principio y pasarlo después. De esta manera nos damos cuenta de lo que tiene el sistema. También se puede coger un programa que funciona bien en otro ordenador y probarlo en el nuestro. Se ven entonces las diferencias, tanto en el curso del programa como en el propio BASIC. Cuando compre un ordenador debe poner especial cuidado en saber qué es lo que el correspondiente BASIC ofrece.

¿Hasta qué punto es bueno un BASIC?
La mejor manera de verlo es al escribir un programa, o bien tomar uno de los mayores de los que aparecen en este libro y ponerlo en el ordenador. Por regla general, uno de estos programas no funcionará a la primera porque se cometen errores y porque hay diferencias en los dialectos. Su BASIC se negará a funcionar, pero el problema no es tan grave. Sobre la pantalla aparecerán comentarios y notas sobre errores. En la mayoría de los ordenadores las instrucciones se almacenan en lenguaje BASIC y el microprocesador las convierte en microprogramas. El tiempo de traducción le dirá lo bueno que es su BASIC en una determinada máquina.

El BASIC y su procesador
Puesto que en los microordenadores el BASIC es transformado y utilizado de modo inmediato, la calidad del BASIC y de los programas en este lenguaje depende de la calidad del procesador. A cada voz de instrucción en BASIC le encuentra en su memoria ROM un microprograma que traduce la instrucción en pequeños pasos. También la calidad de estos microprogramas influye sobre lo bueno o lo comprensible que es un BASIC.

El BASIC y la memoria
Una memoria se puede ir llenando hasta rebosar. La descripción técnica le indicará cuántas instrucciones de BASIC puede admitir su ordenador, si bien tales cifras son siempre un poco teóricas. Es importante tener en cuenta que en el caso de los programas grandes no es suficiente con el espacio de la memoria. Es un inconveniente puesto que el BASIC, junto con el procesador, manejan la memoria y por lo general poco se puede intervenir. Cuando un programa es demasiado grande hay que hacer dos de él, de tal suerte que el segundo trabaje con los datos que el primero ha calculado. Algunas versiones de BASIC tienen instrucciones y dispositivos que sacan programas consecutivos de una memoria de programas.

El BASIC en los ordenadores grandes
Cuando se trabaja con una pantalla o una estación de datos que están conectados a un ordenador grande, todo es distinto. El BASIC estará siempre dentro de un programa de gestión llamado «sistema operativo». Procura que los numerosos usuarios puedan compartir las ventajas del ordenador grande. A esto se le llama «sistema de información». Para estos sistemas tiene especial importancia el contacto con el entorno. ¿Qué le parece que su ordenador doméstico pueda ponerse en contacto con el de su amigo? Esto pronto será posible a través del teletexto.

A un BASIC pequeño le corresponde un ordenador pequeño, y a un ordenador pequeño un BASIC que no sea extenso. Todo trabajo de ordenador está relacionado con los cálculos. Por esa razón, incluso los dialectos BASIC más sencillos disponen de todas las posibilidades actuales de cálculo.

El BASIC grande

Cuando el BASIC como lenguaje de programación se vuelve más extenso y dispone de más instrucciones y mejores representaciones de datos, hace falta también un ordenador más grande. En consecuencia, a un ordenador mayor le corresponde asimismo un BASIC más grande. Un BASIC excesivamente grande ya no sirve para el ordenador personal o el microordenador. Requiere una máquina grande, de cuya utilización participan varias personas. Cada una tiene su propia pantalla con el correspondiente teclado y se le asigna su zona del ordenador, o sea, memoria y tiempo de cálculo.

Para este tipo de utilización de un ordenador grande existe igualmente un BASIC. Esto presenta varias ventajas. Se dispone de un acceso común a los datos y los programas. Y dentro de este contexto es también importante el acceso a los grandes sistemas informativos tales como el teletexto. Así por ejemplo, es posible escribir cartas en la pantalla y transmitirlas a otra pantalla. Esto es una comunicación electrónica a través de un buzón electrónico, y con ayuda del BASIC.

¿Qué es un compilador?

La idea de los lenguajes de programación de alto nivel consiste en liberar al usuario del lenguaje de máquina, que siempre es muy técnico. Se busca, como en el caso del BASIC, un lenguaje que el usuario pueda utilizar. No debe ir cargado de manera innecesaria con cuestiones técnicas. Pero un lenguaje de este tipo únicamente tiene sentido si el ordenador puede traducir automáticamente al lenguaje de máquina los programas que hay escritos, y puede comprobar además si están bien escritos. (Para cada error hay en el programa de traducción un comentario).

El programa que se encarga de la traducción y de la comprobación de los errores se llama «compilador». Es una palabra de inglés americano y significa más o menos «recopilar» o «agrupar». El ordenador solamente puede entender un lenguaje de programación cuando tiene un compilador para ello.

El intérprete

Un lenguaje de programación de alto nivel debe ayudar al usuario y liberarle de las peculiaridades técnicas de la máquina. No tienen nada que ver con sus problemas. Pero un lenguaje de programación de alto nivel solamente tiene sentido cuando el ordenador puede trasladar, o traducir, él mismo y de modo automático al lenguaje de máquina programas escritos en un lenguaje de este tipo. Puede hacerse de manera que una instrucción, escrita en el lenguaje de programación de alto nivel, es transformada y ejecutada de inmediato. A continuación viene una nueva instrucción. Sucede lo mismo que en la interpretación de lenguas; no se traduce, sino que se interpreta. Un programa que sabe hacerlo es un «intérprete». El intérprete es una cosa distinta a un compilador que, como traductor, traslada programas en su totalidad. Se les puede traducir una vez que la última instrucción ha sido traducida. Por regla general, el BASIC se interpreta y no se compila.

¿Qué es un sistema operativo?

El BASIC conoce dos clases de órdenes al ordenador. Unas son las instrucciones que aparecen en los programas: LET, GOTO, IF, FOR, INPUT. Otras proceden del usuario y se las llama mandatos: RUN, LIST, LOAD. El sistema operativo, que lo hay tanto para ordenadores domésticos grandes como para microordenadores, no es otra cosa que una extensa colección de mandatos con los que puede controlarse al ordenador. También en los grandes ordenadores está el sistema operativo para apoyar la actividad del usuario, que en este caso se llama operador.

El BASIC en resumen

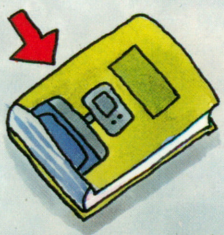
En estas dos páginas encontrará las palabras de instrucción y de mando más importantes del BASIC. La mayoría de ellas funcionan también en los dialectos más sencillos de este lenguaje. ¿Se acuerda de la diferencia esencial que existe entre las instrucciones y los mandatos? Estos últimos los da el usuario al ordenador directamente a través del teclado. Sirven para el manejo directo y son ejecutados inmediatamente por el microprocesador. Las instrucciones sólo vienen en los programas. Estos indican al ordenador lo que hay que hacer, qué es lo que se espera de él. Y por último, están las funciones. En realidad también son instrucciones puesto que se utilizan en el programa.
Un buen ordenador BASIC, además del manual tiene también el llamado mapa de referencias, que a menudo consta de varias páginas. En él se resumen con brevedad y claridad todas las reglas para las instrucciones y los mandatos.

Los mandatos del BASIC

AUTO Proporciona durante la programación una sucesión automática de números de instrucción. Después de que se ha dado entrada a una instrucción, en pasos hasta 10 se va al siguiente número.

LIST Muestra sobre la pantalla o en la impresora el programa que está haciendo sola. Cuando no sigue ninguna instrucción, comienza por la que tiene el número más bajo.

LOAD Lleva un programa desde la memoria de datos hasta la memoria de trabajo (de manera diferente según el tipo de dialecto BASIC).



RENUM Numera de nuevo las instrucciones: en serie creciente y cerrada.

RUN Pone en marcha el programa que espera en la memoria. Si no hay número de instrucción, comienza por la que tiene el número más bajo.

REWIND Rebobina el cassette.

SAVE Almacena el programa en un soporte de datos (diferente según el tipo de dialecto BASIC).

CLEAR Borra todos los datos de la memoria de trabajo.

NEW Borra todos los programas de la memoria de trabajo.

CONT Pulsando una tecla, continúa un programa interrumpido en el lugar en que se encontraba.

MEM Indica la cantidad de espacio que queda libre en la memoria.

Las instrucciones del BASIC

CLOSE	Cierra al final del programa un inventario. CLOSE viene seguido del número de inventario (diferente según el tipo de dialecto BASIC).
DATA	Abre una línea en la que hay constantes de programa.
DIM	Define un sector numérico con su longitud, que aparece entre paréntesis.
END	Designa el final de un programa.
FOR	Comienza un bucle con número fijo de recorridos. Se escribe FOR N = 1 TO 100.
GET	Recoge un conjunto de datos de una memoria de datos.
GOSUB	Se bifurca hacia un subprograma. Una vez que éste ha transcurrido, regresa a GOSUB.
GOTO	Es la bifurcación hacia una instrucción, cuyo número sigue a GOTO.
IF	Dirige una bifurcación en dos sentidos. A IF sigue siempre una condición. Si se cumple ésta, se hará lo que sigue en IF. Pero si no se cumple, le toca el turno a la siguiente instrucción.
INPUT	Escribe ? en la pantalla y espera a que se dé entrada a los datos.
LET	Inicia una línea de instrucción en la que se calculará. El resultado aparecerá en el campo cuyo nombre sigue a LET. LET B = A × A multiplica A por sí misma y almacena el resultado bajo el nombre de B.
NEXT	Dice al programa donde finaliza un bucle FOR. A NEXT le sigue siempre el nombre de la variable que se cuenta en FOR.
OPEN	Abre un inventario para su tratamiento (diferente según el tipo del dialecto BASIC).
PRINT	Da los datos de salida sobre la pantalla y/o la impresora.
PRINT	Es la instrucción para una impresión exigente.
USING	A USING siguen los nombres de datos y después una máscara de estampación para disponer lo que se imprime.
PUT	Almacena un programa en un soporte de datos de programa.
READ	Lee los datos almacenados en DATA, uno tras otro, a lo largo de todo el programa.
REM	A REM sigue una frase que explica algo.
RESET	Ajusta el inventario de datos al comienzo.
RESTORE	Ajusta al DATA con el número más pequeño la instrucción con la cual deben leerse los datos que hay en DATA, o bien a una cuyo número se da.
RETURN	Busca un GOSUB que lleve de regreso el subprograma a la posición correcta.
STOP	Finaliza un programa.

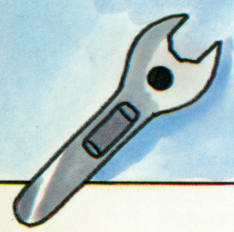
Las funciones del BASIC

ABS(X)	Se toma el valor absoluto de X, o sea, la sucesión de cifras sin el signo.
INT(X)	Se toma la parte entera de X. Se desprecia lo que va detrás de la coma.
RND	Se genera una cifra aleatoria, o al azar, comprendida entre 0 y 1.
&PI	3,1415926... el número
SQR(X)	Se halla la raíz cuadrada de X.
SUM(A)	A es un sector de datos que se suman.
SIN(X)	El seno de X.
COS(X)	El coseno de X.
TAN(X)	La tangente de X.
COT(X)	La cotangente de X.
LGT(X)	El logaritmo de base 10.
LOG(X)	El logaritmo de base e.
MAX(A)	Se calcula el máximo número del sector A.
MIN(A)	Se calcula el mínimo número del sector A.

Los otros lenguajes son útiles

El BASIC no es en modo alguno el único lenguaje de ordenadores que hay. Y lo mismo que sucede en la vida diaria, es ventajoso no hablar únicamente una lengua.
El que sólo domina un lenguaje de programación es fácil que lo sobrevalore. Se cree entonces que sabe todo y es un experto. Pero esto sólo se logra cuando se sube de nivel y, entre otras cosas, se conocen varios lenguajes de programación.
Existen varias docenas y algunos de ellos son especiales para distintos fines. El COBOL es un lenguaje para cuestiones comerciales, el FORTRAN es el de los ingenieros y científicos. Un lenguaje universal es el PL/1, utilizado igualmente por comerciantes, técnicos y científicos. El APL es un lenguaje para el diálogo de cálculo, y el PASCAL es uno relativamente nuevo. Todos ellos son de los llamados lenguajes de programación de alto nivel porque están dirigidos al usuario, lo mismo que el BASIC.

ASSEMBLER para el ordenador



Este es un lenguaje de programación muy técnico que requiere algún esfuerzo por parte del usuario. A diferencia de lo que sucede con los lenguajes de programación de alto nivel, con él se consigue un gran acercamiento a la máquina; es casi un lenguaje de máquina. «Assembler» en inglés es un montador. Los programadores en ASSEMBLER son especialistas.

FORTRAN para los técnicos
El BASIC ha surgido del FROTRAN; es en cierto sentido el FORTRAN de los no iniciados. La palabra viene de **FOR**mula **TRAN**slator «traductor de fórmulas». El FORTRAN es el lenguaje de programación de alto nivel más antiguo, creado en los años cincuenta.

COBOL para comerciantes
La palabra COBOL viene de **CO**mmon **B**usiness **O**riented **L**anguage, que significa «lenguaje general de orientación comercial». Los programas COBOL se leen fácilmente; las palabras de las instrucciones proceden del inglés americano. Al campo CUENTA se le añade en COBOL un IMPORTE con la siguiente instrucción:
ADD IMPORTE TO CUENTA GIVING CUENTA.
El COBOL es el lenguaje de programación más utilizado y apenas tiene dialectos.

PL/1 para todos los gustos
Hace unos veinte años se intentó con el PL/1 crear un lenguaje que fuera para todo: para los comerciantes, los técnicos y los científicos. PL/1 significa **P**rogramming **L**anguage **1**, o sea, «lenguaje de programación número uno». No se ha convertido en el número uno, pero ha conseguido imponerse entre los ordenadores grandes, inmediatamente por detrás del COBOL.

APL para el diálogo de cálculo
Con un lenguaje de diálogo como el BASIC se pueden dar al ordenador directamente instrucciones. También el APL es un lenguaje de diálogo para cálculo, pero más complejo y moderno que el BASIC. Es la abreviatura de **A** **P**rogramming **L**anguage, «un lenguaje de programación». En lugar de palabras de instrucción, el APL tiene símbolos como los que se utilizan en matemáticas. Con ello los programas son muy breves y parecen fórmulas.

PASCAL, ADA y muchos otros
PASCAL y ADA son lenguajes modernos para el ámbito científico. Los nuevos lenguajes de programación son difíciles de introducir porque nadie se toma el molesto trabajo de convertir un programa FORTRAN en uno PASCAL sólo porque éste sea muy nuevo y presente algunas ventajas.

Indice alfabético

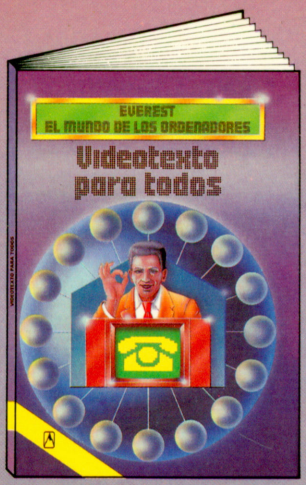
Indice alfabético

Contenido

Contenido

En esta colección han aparecido, con el mismo formato:

El ordenador personal
Desde hace algunos años todos podemos acceder al mundo de los ordenadores. Se trata de unos aparatos fascinantes, cuya estructura y modo de funcionamiento se describen en este libro.



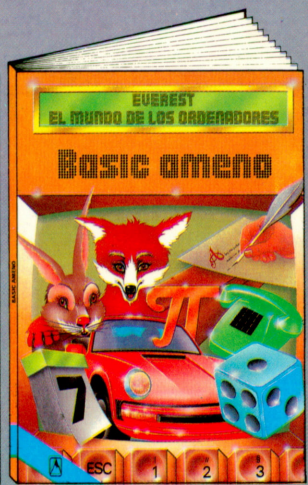
Videotexto para todos
Con la aparición del Videotexto (Vtx) se abre un inmenso campo en el mundo tecnológico. Este libro le proporciona toda la información sobre el Videotexto.

Basic. Programar es fácil
El Basic es uno de los lenguajes de programación más utilizados, con el que se programan ordenadores domésticos y otros de mayor volumen, para que hagan todo aquello que queramos.

Su calculadora. Cómo aprovecharla mejor
Todo el mundo posee una calculadora de bolsillo pero, ¿sabe usted aprovecharla al máximo? Este libro le explica todo lo que puede hacerse con el teclado de su calculadora.

Su casa, llena de datos
Este libro le ofrece una guía y una ayuda imprescindible para poder incorporar las múltiples aplicaciones informáticas a la economía de su propia casa.

Diccionario básico del ordenador
En este libro se hallan los conceptos informáticos básicos sobre el ordenador, en un lenguaje sencillo, sin tecnicismos. Está orientado hacia todos los que, hoy en día, utilizan ordenadores.



BASIC ameno
Se presentan doce programas, en Basic, ya comprobados, que funcionarán en su microordenador. Se muestra también lo que cada programa es capaz de realizar, cómo tratarlo y qué otras cosas podemos hacer con ellos.

El microordenador, como máquina de escribir
Mucha gente utiliza ya su microordenador como si se tratara de una máquina de escribir, redactando cartas y otros textos en la pantalla. Este libro le explica, de una forma clara y completa, cómo aprovechar y utilizar estas funciones de su microordenador.

Editorial Everest s.a.