EVEREST

EL MUNDO DE LOS ORDENADORES

Basic amena

Rolf Lohberg · Theo Lutz

Basic amend

Ilustraciones: Gerhard Utecht

EVEREST

EL MUNDO DE LOS ORDENADORES

Rolf Lohberg · Theo Lutz

Basic amano

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: Noch mehr Basic-Spass
Traducción: Tradutex

EDITORIAL EVEREST, S.A.
Ctra. León-La Coruña, km 5 - LEÓN
ISBN: 84-241-5329-4
Depósito legal: LE. 252 – 1987
Printed in Spain - Impreso en España

EDITORIAL EVERGRÁFICAS, S.A.
Carretera León-La Coruña, km 5
LEÓN (España)

Introducción

Más de uno habrá que posea un ordenador doméstico y no tenga programas. A otros, sin embargo, les gustaría tener más programas para su microordenador. En este libro encontrará doce problemas, uno en cada página doble, con programas de funcionamiento comprobado y con sus correspondientes instrucciones de manejo. Todos estos programas están escritos en un BASIC muy simplificado y han funcionado ya en un pequeño ordenador de bolsillo que puede almacenar 26 variables de BASIC de siete caracteres cada una, pudiendo ampliarse (en intercambio con 1424 pasos de programa) hasta 178 variables. Ello es posible gracias al lema de «cuanto más corto es el programa, tantos más datos podrá almacenar». Se asombrará de todo el jugo que se le puede sacar a este esclavo electrónico de dimensiones tan relativamente pequeñas.

No cabe duda de que estos doce programas funcionarán, bajo estas condiciones, en cualquier ordenador de mayor tamaño. Casi todos nuestros programas pueden ampliarse y mejorarse, sobre todo si disponemos de un ordenador con mayor capacidad. El libro ofrece, en este sentido, varias ideas orientativas, aclarando también el trasfondo de los problemas. Es, por tanto, una especie de manual de taller.

Si usted no es ningún experto en BASIC no debe preocuparse, nada más lejos de nuestra intención. Pero tampoco estaría de más si conociese algo más sobre este lenguaje de programación. De todas formas hemos configurado los programas dejando intencionadamente de lado todos los lujos y trucos de prestidigitador, que aunque puedan resultar muy elegantes, impiden comprender con toda claridad lo que el programa hace en cada paso. Más tarde podrá dedicarse a embellecerlos, si quiere. Por ejemplo, si nuestro programa de simulación de tráfico rodado contiene sólo dos cruces, ello no obsta para que usted lo convierta en toda la red de autopistas entre Hamburgo y Munich. No dude ni por un momento que ese ejercicio le supondría profundizar mucho en sus conocimientos de BASIC.

Y ya está dicho todo. Sólo nos queda desearle que disfrute a sus anchas con nuestros programas en su microordenador.

Doce veces BASIC

Con este librito pretendemos suministrarle doce programas que no necesitará programar usted mismo. Sólo tiene que introducirlos en su ordenador. Pero, puede que al hacerlo tenga algún problema. Todos los programas han funcionado ya en un ordenador. Han sido comprobados y verificados. Pero el hermano del demonio de la errata es el demonio del error en el programa. Por lo tanto, no se lleve las manos a la cabeza si en alguno de estos programas hay aún algo que hacer.
No obstante, y para que pueda disfrutar al máximo con estos programas, hemos añadido tanta información como nos ha sido posible, pues nuestro objetivo final no es otro que darle una mayor seguridad en el tratamiento de programas.
El motivo de haber elegido el BASIC como lenguaje de programación es que se trata de un lenguaje comprendido por la mayoría de los microodenadores y computadoras de bolsillo.

Así está estructurado el libro

Este libro es una especie de manual de taller. Por un lado le ofrece material y por el otro ideas y consejos de cómo utilizarlo. Ante todo necesitará un ordenador que entienda el lenguaje BASIC. El BASIC que hemos utilizado es muy elemental. Si considera que los programas que hemos formulado son extremadamente sencillos, debe tener en cuenta que nuestra intención es mantener los programas lo más comprensibles y transparentes posible. En cada página doble planteamos un problemita convertido ya en programa. También hay un texto que contiene cifras y fórmulas concretas y otro en el que se expone la idea según la cual se ha elaborado el programa en sí. Cada programa va acompañado de un comentario en el que se aclaran los pasos más importantes, incluyendo en su mayoría datos para la comprobación del programa, para que pueda asegurarse de que una introducido por el teclado hace aquello que usted se imagina que hará.

Así debe procederse

Cuando quiera ocuparse de un programa, le recomendamos que lo introduzca enseguida en su ordenador. No olvide borrar antes la memoria de programas (NEW) ni la de datos (CLEAR). Con ello evitará posibles dolores de cabeza.
Ya que ni usted puede garantizar no haberse equivocado en la introducción del programa, ni tampoco nosotros asegurar que no se ha colado algún fallo, deberá proceder a comprobar su funcionamiento. Aquí le serán de ayuda los datos de comprobación que le suministramos.
Si su ordenador dispone de un sistema de almacenamiento de datos en cintas o discos, le recomendamos que archive el programa ya verificado en su biblioteca; de esta forma se ahorrará volverlo a introducir cada vez que lo necesite.

Ayudas de comprobación

Casi todos los ordenadores, incluso los de formato de bolsillo, disponen de ayudas para la verificación, la búsqueda de errores y el análisis del desarrollo detallado del programa. Hay dos expresiones técnicas que debe conocer: DEBUG y TRACE.

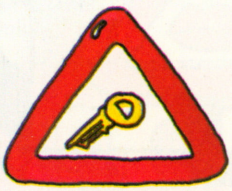
DEBUG es auténticamente americano y su traducción literal no significa nada más que «despiojar». Los piojos son, en este contexto, los fallos del programa. TRACE es la palabra inglesa para «rastro». Con ella se persigue o se sigue el rastro del desarrollo de un programa. El «Debugging» puede que funcione, por ejemplo, poniendo en marcha el programa con la orden DEBUG. Entonces puede pasarse de orden a orden comprobando qué valores encontramos en las variables afectadas y qué otras consecuencias tiene dicha orden. Las comprobaciones son pesadas, pero imprescindibles.

¿Qué son dialectos del BASIC?

El BASIC es un lenguaje de programación relativamente antiguo ya que procede de los años sesenta. En un principio, el BASIC debía mantenerse con una estructura sencilla, conteniendo pocas pero vitales órdenes y ser utilizado ante todo para fines educativos.

Sin embargo, al nacer los ordenadores domésticos y los microprocesadores, resultó que el BASIC era el lenguaje idóneo para estas pequeñas máquinas. La traducción de los programas en BASIC a código máquina es sensiblemente menos costosa que en otros lenguajes de programación.

Con la ampliación de los sistemas de procesamiento de datos y el posterior desarrollo de los ordenadores, el BASIC también tuvo que perfeccionarse. Y aquí empezó la confusión en los lenguajes; un fabricante añadía esto, el otro aquello... En la actualidad existe sin duda una base fija e inalterable de BASIC, pero ésta ha sido ampliada en muchas direcciones, saliéndose a veces de los márgenes. Así es como podemos encontrar hoy muchos dialectos del BASIC. Resulta apenas posible saber cuántos existen en la actualidad, por lo que no resultaría extraño que un programa, que en un determinado aparato se siente feliz y como en casa, no funcione en absoluto en otro ordenador.



¿Hasta qué punto nos estorbará aquí un dialecto de BASIC?

La sospecha de que nos hemos topado con un dialecto del BASIC se nos presentará siempre que un programa correctamente introducido se niegue a funcionar. Se queda simplemente parado en una orden. Si se trata de un programa en BASIC ampliado, dará a menudo mensajes de error por sí solo. Es posible que en esos casos se sepa ya cuál es la desviación del dialecto. Si no, habrá que echar un vistazo al manual. Se busca la descripción detallada de la orden en cuestión y se comprueba caracter por caracter hasta hallar dónde está la diferencia. Con frecuencia se tratará sólo de una nimiedad. En nuestros programas hemos utilizado un BASIC compatible con la mayoría de los dialectos. Resultaría muy extraño que su ordenador no entendiera alguna palabra clave utilizada en estos programas.

El manual

Para trabajar con un ordenador y un lenguaje de programación es necesario tener el manual a mano. Hay muchos a quienes les encanta enfrentarse y solucionar los problemas jugando. Pero a menudo cuesta demasiado tiempo; tiempo que nos puede ser ahorrado con un buen conocimiento del manual. Por lo demás, su ordenador es tan bueno como el manual que lo describe. Y esto es aplicable también para sus conocimientos de BASIC.

Los programas los encontrará

Un poco de BASIC

El BASIC es un «lenguaje de programación de alto nivel». Está destinado al usuario final y al principiante de la programación. De ahí viene su nombre: «basic», que significa «fundamental». El lenguaje es considerado como muy sencillo, aunque no siempre resultará fácil aprenderlo. Ante todo cuando lo que se quiere representar con él no es sencillo. En el marco de este librito resulta apenas posible ofrecer una introducción al BASIC. Para ello hay otras obras mejores (como, por ejemplo, en esta misma serie: «Programar en BASIC, muy sencillo»).
Por este motivo, estas dos páginas deben servir, ante todo, para refrescar la memoria; una especie de BASIC en un vistazo. Aquí se encontrará resumidos los principales conceptos de BASIC, así como las palabras utilizadas en los programas.

Programas, órdenes e instrucciones
Un programa en BASIC está compuesto por diferentes instrucciones. Cada instrucción se escribe en una línea, y cada línea comienza con un número: el número de instrucción o de línea. Las instrucciones se ejecutan en el orden correlativo de los números de línea. Si en lugar de numerar las líneas con números seguidos lo hacemos de diez en diez, al ordenador no le molestará en lo más mínimo, pero se dispondrá de una gran ventaja: podremos insertar otras instrucciones según se necesiten. Dentro de una instrucción hay siempre una o varias órdenes o palabras claves que se expresan en BASIC y forman las instrucciones o sentencias para el ordenador. Estas órdenes proceden del inglés y hay que aprender qué significan y cómo se las debe utilizar. Toda orden necesita, por último, datos y nombres de datos. Los nombres de datos se llaman también «variables», ya que se les puede dar una vez este valor, otras este otro, etc. Los datos fijos, los que no varian, se denominan «constantes».

Los datos son cifras o literales
Los datos son anotaciones, informaciones o enumeraciones y valores. Pueden constar de cifras o números, caso en que se denominarán «numéricos». Pero si constan de letras u otro tipo de caracteres, entonces de llamarán «alfanuméricos». Un dato alfanumérico puede llamarse también un «literal». Los literales en BASIC se introducen siempre entre comillas; los números y cifras se escriben normal con un punto decimal (un punto y no una coma).

Dando nombres a los datos
Los datos —cifras o literales— que pueden cambiar su valor a lo largo de un programa, se llaman variables. Se les asigna un nombre al que pertenecerá el valor válido en ese momento. Las variables numéricas, es decir aquellas cuyo valor es una cantidad, constan de una letra o de una letra seguida inmediatamente de un número; por ejemplo K15, ó F, ó B3. Las variables alfanuméricas —también llamadas literales— constan de un nombre, compuesto por una letra o una letra y un número, a los que se añadirá sin dejar espacios un signo de dólar: A$, G18$ ó B3$.
En nuestros programas nos limitaremos estrictamente a esta norma. En el manual encontrará de cuántas variables se compone su BASIC y cuántos caracteres o cifras puede tener como máximo una variable.

Fórmulas, expresiones y la orden LET

Las «expresiones» son configuraciones con sentido, compuestas de nombre de variables, constantes, paréntesis y signos como + (más), − (menos), * (multiplicado por) o / (dividido). En una expresión de este tipo se juntan los valores numéricos. Todas las variables que aparezcan en una expresión tienen que haber sido definidas previamente en el desarrollo del programa. Si M debe tener el valor −3,14, habrá que escribir (por ejemplo en la línea 150) la orden

```
150 M = −3,14
```

La utilización del signo igual (=) se define aquí como «asignación». Tras el signo de igual puede encontrarse también una expresión. En este caso se llamará una «fórmula». Cuando en determinadas órdenes aparezcan fórmulas deberá escribirse la orden LET delante de la fórmula.

Bucles, o las órdenes FOR, TO y NEXT

Cuando una misma secuencia de órdenes o instrucciones debe realizarse varias veces, el proceso se define como «bucle de programa». Un bucle que deba ejecutarse diez veces comenzará en BASIC (por ejemplo, en la línea 250) con

```
250 FOR N = 1 TO 10
```

y podría acabar, por ejemplo en la línea 350, con

```
350 NEXT N
```

Este tipo de programación obliga a que todas las líneas entre la 250 y la 350 se ejecuten diez veces. Con cada ejecución, el contador del bucle (llamado aquí N) se incrementa en 1.

Bifurcaciones, o la orden GOTO

Si en la línea 750 escribimos la orden

```
750 GOTO 100
```

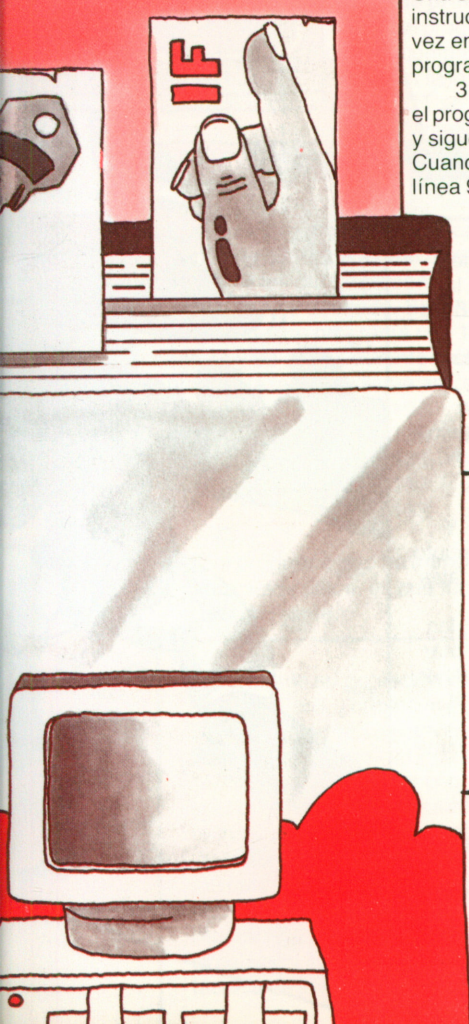
el programa seguirá con las instrucciones que aparezcan en la línea 100. A este tipo de bifurcaciones se les otorga el nombre de «bifurcación absoluta».

Subrutinas, o GOSUB y RETURN

Una subrutina es una secuencia de instrucciones que puede ser utilizada de vez en cuando en las partes del programa que se necesite. Si la orden es

```
310 GOSUB 900
```

el programa se bifurca hacia la línea 900 y sigue las instrucciones a partir de allí. Cuando se encuentra (por ejemplo en la línea 990) con la orden RETURN, regresa a la línea que viene a continuación de la 310. Lo que hay entre las líneas 900 y 990 es lo que se llama subrutina, palabra de la que procede la abreviatura SUB (pues en inglés, para variar, se dice «subroutine»). GOSUB significa «ve a la subrutina». La palabra RETURN significa, en español, «¡regresa!».



Observaciones con REM

En los programas resulta muy útil insertar de vez en cuando líneas que no contengan instrucciones, sino sólo observaciones. Para ello se utiliza la abreviatura de la palabra inglesa «remark», REM:

```
390 REM AHORA SIGUE
    LA PARTE B DEL
    PROGRAMA
```

STOP y END

Es recomendable indicar cuándo ha finalizado un programa, STOP señaliza un final de programa lógico —el ordenador debe parar—, mientras END indica un final físico: Ya no quedan instrucciones por ejecutar.

Bifurcación condicional, o la orden IF

A menudo hay que comprobar si una variable tiene un determinado valor. Si de una CUENTA debe deducirse una CANTIDAD, sólo podrá hacerse si en la cuenta hay suficiente dinero para ello. Con la orden

```
520 IF (CUENTA−CANTIDAD=0)
    GOTO 730
```

el ordenador comprueba (IF en español es el «si» condicional) si la diferencia entre estado de cuenta y cantidad a deducir da como valor cero o incluso un valor negativo. Si éste es el caso, es decir, la cuenta está vacia, el programa seguirá en la línea 730. En caso contrario seguirá de forma normal la línea de instrucciones 521.

Introducir, o INPUT

Con la instrucción INPUT, el ordenador avisa que necesita que le den datos. Si en la línea 470 escribimos la instrucción

```
470 INPUT «INTRODUCIR
    VALOR B»; B
```

el programa se para al llegar a la línea 470 e imprime, o muestra en pantalla, el comentario «INTRODUCIR VALOR B». Si entonces pulsamos una cifra en el teclado, ésta es asignada a la letra B y el programa continúa.

Salida, o PRINT

Si en la línea 770 se escribe la orden

```
770 PRINT «CANTIDAD B = »; B
```

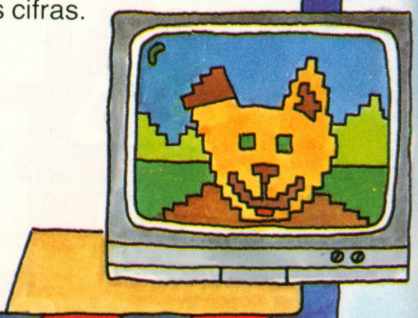
el ordenador imprimirá (o presentará por pantalla) aquello que está almacenado en B —junto con el comentario entre comillas de la orden PRINT.

Dialectos del BASIC y el Manual

Actualmente no existe un BASIC único, sino muchos dialectos de este lenguaje. Para el profesional será todo un reto; si el programa introducido no funciona, puede haber una errata de imprenta, un error en el programa, un fallo de programación, un malentendido o una desviación dialectal del BASIC. Para ello disponemos de dos árbitros: el manual y el ordenador. ¡No se desanime si el programa no funciona a la primera! Ahora viene un interesante trabajo de detectives. ¿Dónde se ha escondido el demonio que queremos cazar? Como siempre, en el más insospechado detalle.

Zorros y conejos

«Zorro dormilón no caza gallinas», aunque en nuestro caso se trate de conejos. En los antiguos libros escolares había un problema de matemáticas que decía: ¿Cuánto necesitará el zorro más veloz para cazar al conejo?
En los libros actuales de matemáticas hay, sin embargo, otro problema: en una isla viven zorros y conejos. Alimento para los conejos hay en abundancia, por lo que los conejos pueden reproducirse cada año en un porcentaje considerable. Pero en la isla viven también zorros, y su medio de sustento no es otro que comerse los conejos. Si hay muchos conejos, los zorros se reproducen, y cuantos más zorros hay, menos conejos quedan. Pero cuando hay pocos conejos, la cifra de zorros desciende, pues se mueren de hambre.
Lo que acabamos de expresar en palabras hay que precisarlo en cifras, y para ello nos harán falta fórmulas que se pueden programar. De esa forma crearemos en el ordenador un modelo que simule el desarrollo de la población de zorros y conejos. Así podremos utilizar el ordenador para saber qué ocurre cuando variamos las cifras.



Cifras para el modelo

Nuestro programa parte de la idea de que en un principio hay una cantidad C de conejos y una cantidad F de zorros. Al final de cada año hay un 40% más de conejos. Pero de los C conejos caen C × F/250 en las fauces de sus vecinos, (250 es una especie de factor de «voracidad»). Cuando al inicio de un año se tiene C conejos y Z zorros, para el año siguiente habrá X conejos, según la siguiente fórmula:

$X = C + C \times 40/100 - C \times Z/250$

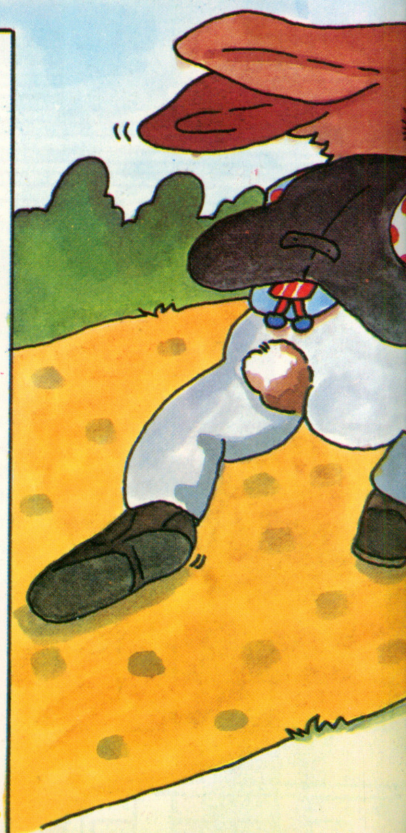
Y ahora los zorros: el procentaje de crecimiento y muerte es $(C/5 - 30)$ %. De los C conejos y Z zorros resultará al pasar el año una cantidad Y de zorros:

$Y = Z + Zx\ (C/5 - 30)/100$

Según estas fórmulas puede calcularse año por año la población de zorros y conejos.

Así será el programa

El programa pide, en la parte de introducción de datos (INPUT) la cifra de conejos C, la de zorros Z y la cantidad de años A.
Un bucle FOR-NEXT se ejecutará desde 1 hasta A. En este bucle, los conejos (C) y zorros (Z) del año pasado se convierten en los conejos (X) y zorros (Y) del nuevo año (N). Se calculan mediante las dos fórmulas de la izquierda, pero tomando de las cantidades X e Y sólo la parte a la izquierda del punto decimal, ya que no existen animales fraccionados.
Entonces se imprimen (PRINT) las cifras N, X e Y. Si queremos aceptar animales negativos, podemos dejar que el bucle FOR llegue hasta el punto final A; pero será mejor comprobar que X e Y sean mayores de cero para que el programa continúe sólo en esos casos. No obstante, también hay que actualizar las cantidades iniciales de zorros y conejos para cada año, es decir X de C e Y de Z (C = X y Z = Y). Llegados a este punto podemos cerrar el bucle con la orden NEXT N.



Comentario sobre el programa

Las instrucciones que se encuentran en las líneas 20 a 50 sirven para dar al ordenador los valores iniciales de nuestra historia de zorros y conejos, presentándolos al mismo tiempo por pantalla. De esta forma podremos saber más tarde en qué punto nos encontramos. Lo que sucede en un año se encuentra entre las instrucciones 100 (FOR) y 300 (NEXT). Trabajamos con un bucle FOR-NEXT que debe realizarse un número A de veces. En 110 y 120 se han programado las fórmulas del modelo. Entre 150 y 156 se comprueba si quedan conejos. En caso afirmativo, el programa continúa comprobando si existen todavía zorros. Cuando no quedan conejos vivos se pone el contador N del bucle a FINAL, es decir N = A. Ahora se ejecuta la orden de imprimir, que no muestra ningún conejo, o sólo conejos muertos. Entre 160 y 165 sucede lo mismo para los zorros.

Las instrucciones 200 y 210 nos preparan los datos para el año siguiente. Con la instrucción de la línea 250 se presenta el año calculado y con la línea 300 se va a la siguiente vuelta, si no se ha alcanzado aún el valor de A.

El programa

```
 10  REM »POBLACION DE ZORROS Y CONEJOS«
 20  INPUT »CONEJOS?«;C
 25  PRINT »CONEJOS;«,C
 30  INPUT »ZORROS?«;Z
 35  PRINT »ZORROS;«,Z
 40  INPUT »AÑOS?«;A
 45  PRINT »AÑOS;«,A
 50  PRINT

100  FOR N = 1 TO A
110  X = INT (C + C*40/100 - C*Z/250)
120  Y = INT (Z + Z*(C/5 - 30)/100

150  IF X > 0 THEN 160
155  N = A
156  GOTO 200
160  IF Y > 0 THEN 200
165  N = A
166  GOTO 200

200  C = X
210  Z = Y

250  PRINT N; « «; C; « «; Z

300  NEXT N
```

El programa en la mesa de pruebas

Una vez haya introducido el programa en su ordenador, deberá comprobar que no ha cometido ningún fallo al escribirlo. Con la orden LIST se imprime (en impresora o pantalla) el listado del programa. Léalo detenidamente y corríjalo si es necesario.

Si desea saber si el programa calcula bien los datos, deberá ofrecerle unos valores de los que sepa con anterioridad el resultado. Si pone en marcha nuestro programa con 100 conejos y 80 zorros durante 5 años obtendrá la tabla siguiente:

CONEJOS:	100.	1.	108.	72.
ZORROS:	80.	2.	120.	65.
AÑOS:	5.	3.	136.	61.
		4.	157.	59.
		5.	182.	56.

»

Si el programa que ha introducido le da estos datos, es que funciona correctamente.

Alteremos el modelo

Si desea jugar con el modelo, debe conseguirse primero un caso estandard. Calcule para un período de tiempo mayor (A = 20) con C= 100 y Z = 100 lo que ocurre. Primero, los conejos aumentan fuertemente su población y los zorros reducen drásticamente su número. Pero, ¿qué ocurre cuando los conejos aumentan sólo en un 30% y los zorros se vuelven más glotones? Para saberlo deberá cambiar, en la línea 110, la cifra 40 por 30, y en lugar de 250 poner 150. Vuelva a empezar con C = 100 y Z = 100. ¿Qué ocurre ahora? Más difícil de responder sería la cuestión de si existe una relación numérica que mantenga estabilizada la cantidad de conejos y zorros, de tal forma que tanto conejos como zorros se mantengan en una misma cantidad. A los conejos les interesa sin duda saber si se están extinguiendo, y a los zorros si se están muriendo de hambre. La representación de estas soluciones por ordenador se llama simulación. En muchos campos científicos se ha convertido en una técnica de investigación muy importante.

Posibilidades de ampliación del modelo

No cabe duda de que podemos mejorar considerablemente nuestro programa. Esta ampliación dependerá, entre otras cosas, del tipo de ordenador que se posee. El programa, tal como está impreso aquí, funcionará en el ordenador más pequeño, pues se las arregla con un mínimo de BASIC. El grado de comodidad para la introducción y presentación de los datos se limita a lo imprescindible.

El calendario perpetuo

En España, al igual que en la mayoría de los países, la fecha suele representarse en el orden Día-Mes-Año. En América y en Inglaterra lo hacen en otro orden: Mes-Día-Año. A menudo queremos saber cómo se llama el día de la semana del que tenemos en número, o cómo se llama el mes. Nuestro programa puede hacerlo. También puede calcular, partiendo de los números de una fecha, el día de la semana en que recae, aunque hacia el año 1700, pues el calendario que tenemos actualmente (llamado gregoriano en honor al Papa Gregorio XIII) no fue aceptado en Alemania, de donde procede este programa, hasta el año 1700. Antes de esta fecha sólo existía el caos.
El primero de abril de 1700 cayó en martes. Partiendo de este único dato, el ordenador calcula semana a semana hasta encontrar la fecha solicitada. Con este programa podemos hacernos nuestros propios calendarios.

Así se convierten los números en cifras

¿Cómo conseguimos saber el nombre del mes con sólo la cifra (entre 1 y 12)? En la línea 100 se multiplica el número del mes por 10 y se suman 100. De esta forma obtenemos una «dirección». Por ejemplo, para marzo, (100 + 10 × 3) obtenemos el 130. Esta «dirección» es precisamente la línea 130. Allí se encuentra la orden que carga el nombre de variable 0$ con el nombre correspondiente del mes.
Y para poder encontrar los días de la semana procederemos de forma similar, aunque el cálculo aquí es algo más complicado.

Así se calcula en gregoriano

Cuando queremos encontrar el día de la semana correspondiente a una fecha posterior al 1 de abril de 1700, primero calcularemos, mediante los días, meses y años, la cantidad total de días desde el año cero. Para ello existe una fórmula que tiene en cuenta también los años bisiestos y la diferente longitud de los meses. De esta cantidad restamos 621049, que son el total de días desde cero al 1 de abril de 1700. La diferencia se divide en paquetes de 7, es decir en semanas enteras. Lo que nos queda es el número del día de la semana, con el cero equivalente al lunes. El séptimo día es domingo, tal como manda la Biblia. En el mismo programa podrá observar en detalle el proceso de funcionamiento.

Un poco de historia sobre el calendario

Nuestro calendario actual parte del hecho de que el Sol necesita un año para volver a estar allí donde empezó. Si dividimos este giro por 365, nos sobrará alrededor de un cuarto de día. Si para eliminar esta molestia introducimos un día extra cada cuatro años, nos acercaremos al giro del Sol con una exactitud tal, que sólo hará falta un día más de cambio cada 3000 años.
En la Edad Media se disponía de un calendario, regalo de Julio César a los romanos en el año 46 a.C.: el calendario juliano. En el Renacimiento se inició la discusión

La arquitectura del programa del calendario

Nuestro programa está construido de tal forma que exige números para el día, el mes y el año. Del número de mes se extrae una dirección en el programa. Allí se carga la variable 0$ con el nombre del mes correspondiente. Se pasa entonces a la parte del programa que convierte la fecha, según el calendario gregoriano, en un número del 0 al 6. De este número se extrae nuevamente una dirección en la que se carga la variable A$ con el nombre correspondiente de la semana. El programa imprime entonces una línea en la que muestra primero el nombre de la semana, el número del día, el nombre del mes y, finalmente, el año.

Algunos detalles del programa

Las instrucciones contenidas en las líneas 10 a 30 preguntan por los valores de día, mes y año, sin comprobar la exactitud de dichos datos. El programa calcula, por tanto, también fuera del ámbito de validez del calendario gregoriano y dice que el 31 de febrero de 1983 es viernes incluso cuando ese día no existe.

En 100 se convierte el número del mes en una dirección entre 110 y 220, donde se halla, con O$ el nombre del mes.

Entre 300 y 350 se conmuta la fecha en día de la semana. Hasta la línea 330 se equiparan los números de los meses de tal forma que luego podemos calcular con valores promediados (365, 25 días al año; 30,6 días al mes).

Con INT se coge sólo la parte entera, a la izquierda de la coma, y se construye con ello los años bisiestos y las oscilaciones de los meses. Resulta muy interesante la línea 350. Con INT (N/7) se calcula de N días la cantidad de semanas; INT (N/7) + 7 nos dice cuántos días tienen estas semanas y si restamos esta cifra de N se consiguen los días restantes que no llegan a completar una semana; una cifra del 0 al 6. Mediante las entradas 400 a 460 se llega así al día de la semana deseado. Entonces, en la línea 500 se junta nuevamente todo el programa para imprimir la fecha completa.

El programa

```
 05 REM «EL PROGRAMA DEL CALENDARIO»
 10 INPUT»DIA«;D
 20 INPUT»MES«;M
 30 INPUT»AÑO«;A
 40 W = A
 50 REM »CONVERSION DE LOS MESES«
100 N = 100 + (M*10)
105 GOTO N
110 O$ = »ENERO«:GOTO 300
120 O$ = »FEBRERO«:GOTO 300
130 O$ = »MARZO«:GOTO 300
140 O$ = »ABRIL«:GOTO 300
150 O$ = »MAYO«:GOTO 300
160 O$ = »JUNIO«:GOTO 300
170 O$ = »JULIO«:GOTO 300
180 O$ = »AGOSTO«:GOTO 300
190 O$ = »SEPTIEMBRE:GOTO 300
200 O$ = »OCTUBRE«:GOTO 300
210 O$ = »NOVIEMBRE«:GOTO 300
220 O$ = »DICIEMBRE«:GOTO 300
250 REM »CALCULO DEL DIA DE LA SEMANA«
300 IF M - 2 > 0 THEN LET M = M + 1: GOTO 330
310 M = M + 13
320 A = A - 1
330 N = INT (365,25*A) + INT (30,6*M) + D - 621049
340 N = N - 1
350 Y = ((N - INT(N/7)*7)*10) + 400
360 GOTO Y
400 A$ = »LUNES«: GOTO 500
410 A$ = »MARTES«: GOTO 500
420 A$ = »MIERCOLES«: GOTO 500
430 A$ = »JUEVES«: GOTO 500
440 A$ = »VIERNES«: GOTO 500
450 A$ = »SABADO«: GOTO 500
460 A$ = »DOMINGO«: GOTO 500
490 REM »EXPRESAR FECHA DEL CALENDARIO«
500 PRINT A$; «»; D; « DE »; O$; « DE »; W
510 END
```

Comprobación del programa

Debido a los posibles fallos en la introducción del programa habrá que comprobarlo sistemáticamente antes de utilizarlo. La comprobación no resultará difícil y le recomendamos la realización de dos tests.

En primer lugar se toman siete días seguidos de lunes a domingo. Así podremos comprobar si el cálculo del día de la semana es correcto.

En el segundo test se comprobarán los nombres de los meses, solicitando el día 1 de cada mes, de enero a diciembre. En este proceso podemos comprobar, de paso, si los días de la semana coinciden.

Y ahora ya podemos empezar. (El 1 de abril de 1700 fue martes, dicho sea de paso). Si no podemos localizar directamente un determinado fallo, habrá que repasar el programa a mano. Para cada línea del programa se calculan los valores que deberían salir, avanzando paso a paso. En general, los ordenadores suelen disponer de determinadas ayudas. Para ello existe la palabra TRACE. Significa «rastro» y con ello se quiere decir que avanzaremos «rastreando» paso a paso el programa.

sobre un nuevo calendario. En 1582, el Papa Gregorio XIII ordenó la reforma del calendario. De aquéllo surgió el calendario gregoriano, con actual validez en todo el mundo. En la Alemania protestante no fue adoptado de forma oficial y definitiva hasta 1700. En consecuencia, nuestro programa trabaja con el calendario gregoriano. Por ello, no sirve de nada consultar en años anteriores a 1700. Por intentarlo, desde luego que no quede, pues es posible calcular días de semana anteriores a esta fecha. Pero si aquel día se llamaba realmente así es ya harina de otro costal.

Refinamiento en el programa

Puede mejorar la parte de introducción de datos, añadiendo instrucciones de comprobación, para que el programa sólo acepte datos reales dentro del calendario gregoriano. Por tanto, sólo deberá aceptar números de años mayores de 1700. Con las fechas de los días puede comprobarse si están dentro del margen mensual. Los 29 de febrero sólo se permitirían si el número del año es divisible por cuatro sin que quede resto. La forma de comprobación es similar a la que hemos utilizado para convertir la fecha en paquetes semanales y hallar los días restantes. Ahora se trata de paquetes de cuatro y será:

$$A - INT (A/4)*4 = 0$$

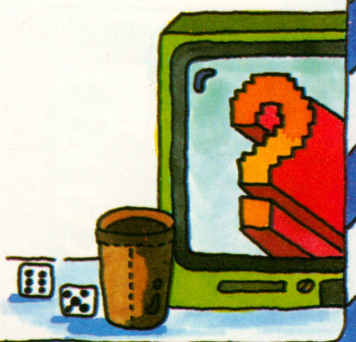
cuando se trate de un año bisiesto.

Variaciones del calendario

Una variante inteligente del programa de calendario podría ser la creación de una subrutina en la que las variables D, M y A sólo acepten valores correctos, suministrando los correspondientes A$ y O$. Donde ahora nos encontramos con un END, deberá escribirse un RETURN. Desde el programa principal se puede acceder a la subrutina mediante GOSUB. Esta subrutina puede utilizarse entonces para diversos fines, por ejemplo, si deseamos imprimir nuestro propio calendario o si queremos que el calendario permanezca en funcionamiento en nuestro ordenador. La solución a este problema la encontrará en esta serie, en el libro «Datos domésticos» con mayor detalle.

Adivinando números por casualidad

El juego funciona de la manera siguiente: el ordenador extrae una cifra al azar entre 0 y 99. Usted deberá adivinar este número disponiendo para ello de ocho posibilidades.
Una adivinanza de este tipo ganará en interés y diversión, si el ordenador, a lo largo del juego, nos va imprimiendo comentarios. Estos nos resultarán especialmente sorprendentes si son totalmente imprevisibles; se trata de comentarios casuales. Siempre que el ordenador deba imprimir un comentario, se extrae un número al azar entre 0 y 9. Si sale el 5, se imprimirá el comentario número 5. Estas cifras extraídas al azar para los números a adivinar y para los comentarios son creadas por lo que se llama un generador de números aleatorios.
En nuestro juego hay tres momentos donde pueden aparecer comentarios: cuando el ordenador pide continuar adivinando, cuando se gana y cuando se pierde.



Borrador del programa de adivinanzas
El programa crea primero un número aleatorio de dos cifras. En el programa se llamará C (de Casualidad). Ya que en total no se puede adivinar más de ocho veces, se organizará un contador K que al comienzo del programa se pone a 0 y que a lo largo del mismo va aumentando de uno en uno. En cada vuelta se comprueba si se ha alcanzado ya el 8. En cada vuelta se pide al jugador que introduzca su número, para lo cual se recurre al generador de comentarios aleatorios (comentarios 01 a 10) a través de una subrutina generadora de un número aleatorio de una sola cifra. Hemos llamado G al número que introducimos para adivinar. En cada vuelta se compara G con C mediante una instrucción IF. Si G = C, el jugador ha ganado.
Si G > C, el programa imprimirá «G ES DEMASIADO GRANDE», y si G < C, el programa imprimirá «G ES DEMASIADO PEQUEÑO». En ambos casos se entrará en otra vuelta con un comentario casual (comentarios 11 a 20). Si tras ocho vueltas no se ha adivinado el número correctamente, el jugador habrá perdido y se le hará saber un comentario al respecto (comentarios 21 a 30). También aquí se propone un nuevo juego.

Así se consiguen números aleatorios
Haremos que el ordenador tire los dados: determinaremos el número a adivinar mediante un generador de números aleatorios. Se trata, ni más ni menos, de un programa. Para ello se toma una cifra de partida que en nuestro programa es el número 0,3141592654 (las primeras diez cifras del número Pi). Al multiplicar este número por sí mismo, es decir, al elevarlo al cuadrado, volviendo a elevar al cuadrado el resultado, etc., se obtiene una secuencia de números que contiene cifras muy variadas y mezcladas. Sin embargo, si elevamos al cuadrado varias veces cualquier número entre el 0 y el 1, se va volviendo más pequeño hasta alcanzar el cero. Por ello, con cada vuelta comprobaremos que la cifra que se está utilizando en ese momento sea mayor de 0,4. Si es menor, se le suma 0,35.

Los números aleatorios
La cifra de salida Q = 0,3141592654 cambia constantemente su valor al multiplicarse por sí misma, convirtiéndose en una nueva Q. Si necesitamos un número de dos cifras, tomaremos las cifras que se hallan en la posición cuarta y quinta detrás del punto decimal. Si sólo se necesita una cifra, se toma la de la quinta posición. Si Q es menor de 0,4, se suma 0,35.

	Número aleatorio
Q0 0,3141592654	
Q1 0,4411075298	10
Q2 0,1945758528	7
Q3 0,2965628595	6
Q4 0,4180435313	4
Q5 0,1747603941	6
Q6 0,2753734712	7
Q7 0,3910919785	9
Q8 0,5492173206	1
Q9 0,3016396652	3

Los comentarios casuales
Se multiplica por 10 un número aleatorio entre 0 y 9, y se le suma la dirección de la orden precedente (200, 400 ó 600). Así se obtiene la dirección del comentario que se introduce en una variable N.

El programa

```
005 REM «ADIVINAR NUMEROS CON COMENTARIOS
    CASUALES
010 PRINT »ADIVINE UN NUMERO ENTRE 0 y 99«
020 PRINT »PUEDE PROBAR HASTA 8 VECES«
030 PRINT »INTRODUZCA PRIMERO UN NUMERO ENTRE 0 y 0.3«
040 INPUT T
050 Q = 0.3141592654
060 Q = Q + T
065 REM »SE EXTRAE EL NUMERO A ADIVINAR«
070 IF (Q > 0.4) THEN LET Q = Q - 0.35)
071 Q = Q*Q
072 R = INT (Q*100000)
073 S = INT (Q*1000)*100
074 C = R - S


095 REM »COMIENZA EL JUEGO«
100 K = 0
110 IF (K > = 8) THEN 600
120 K =K + 1
130 GOSUB 900
140 N = N*10 - 200
150 GOTO N
195 REM »10 COMENTARIOS SOBRE LA ELECCION«
200 PRINT »COMENTARIO 01 "ADIVINE AHORA"«; GOTO 300
210 PRINT »COMENTARIO 02«; GOTO 300
220 PRINT »COMENTARIO 03«; GOTO 300
230 PRINT »COMENTARIO 04«; GOTO 300
240 PRINT »COMENTARIO 05«; GOTO 300
250 PRINT »COMENTARIO 06«; GOTO 300
260 PRINT »COMENTARIO 07«; GOTO 300
270 PRINT »COMENTARIO 08«; GOTO 300
280 PRINT »COMENTARIO 09«; GOTO 300
290 PRINT »COMENTARIO 10«; GOTO 300
300 INPUT G
310 IF (G = C) THEN 400
320 IF (G > C) THEN 350
330 PRINT »G ES DEMASIADO PEQUEÑO«
340 GOTO 110
350 PRINT »G ES DEMASIADO GRANDE«
360 GOTO 110


400 REM »COMENTARIOS SOBRE EL TEMA GANADO«
410 GOSUB 900
420 N = N*10 - 440
430 GOTO N
440 PRINT »COMENTARIO 11, TEMA GANADO«; GOTO 550
450 PRINT »COMENTARIO 12«; GOTO 550
460 PRINT »COMENTARIO 13«; GOTO 550
470 PRINT »COMENTARIO 14«; GOTO 550
480 PRINT »COMENTARIO 15«; GOTO 550
490 PRINT »COMENTARIO 16«; GOTO 550
500 PRINT »COMENTARIO 17«; GOTO 550
510 PRINT »COMENTARIO 18«; GOTO 550
520 PRINT »COMENTARIO 19«; GOTO 550
530 PRINT »COMENTARIO 20«; GOTO 550
550 PRINT »DE TODAS FORMAS NECESITO«; K; »INTENTOS«
560 GOTO 800


600 REM »COMENTARIOS SOBRE EL TEMA PERDIDO«
610 GOSUB 900
620 N = N*10 - 640
630 GOTO N
640 PRINT »COMENTARIO 21, TEMA PERDIDO«; GOTO 750
650 PRINT »COMENTARIO 22«; GOTO 750
660 PRINT »COMENTARIO 23«; GOTO 750
670 PRINT »COMENTARIO 24«; GOTO 750
680 PRINT »COMENTARIO 25«; GOTO 750
690 PRINT »COMENTARIO 26«; GOTO 750
700 PRINT »COMENTARIO 27«; GOTO 750
710 PRINT »COMENTARIO 28«; GOTO 750
720 PRINT »COMENTARIO 29«; GOTO 750
730 PRINT »COMENTARIO 30«; GOTO 750
750 PRINT »DICHO SEA DE PASO, MI NUMERO ERA EL«; C
760 GOTO 800


800 REM »AHORA VIENE LA SEGUNDA VUELTA«
810 PRINT »JUGAMOS DE NUEVO? 1 = SI, 0 = NO«
820 INPUT B
830 IF (B = 0) THEN 850
840 GOTO 70
850 PRINT »HA PERDIDO LA CONFIANZA EN SI MISMO, EH?«
860 STOP
870 END


895 REM »ESTE ES EL GENERADOR DE NUMEROS
    ALEATORIOS«
900 IF (Q < = 0.4) THEN LET Q = Q + 0.35)
910 Q = Q*Q
920 R = INT (Q*100000)
930 S + INT (Q10000)*10
940 N = R - S
950 RETURN
```

Comentarios sobre el programa

Entre las líneas 005 y 074 se toma un valor inicial para Q. A partir de 070 se crea el número aleatorio C. En la línea 100 se pone el contador de vueltas a cero (K = 0). La línea 110 es la entrada en una vuelta. Por ello se comprueba aquí si no se han hecho ya ocho intentos.

En la línea 120 se numera la vuelta actual y se emite un comentario casual que incitará al jugador a que intente adivinar. En la orden 300 se intenta adivinar (INPUT G). Si se ha acertado el número se bifurca el programa de 300 a 400. En caso contrario se comprueba si el número propuesto es demasiado grande o demasiado bajo y en 110 se inicia la vuelta siguiente.

Entre 900 y 950 se encuentra el generador de números aleatorios. Ya que trabaja con las mismas variantes (R, S, N, Q) del generador que crea el número C, se proporciona al mismo tiempo el número C para el juego siguiente.

Ampliando el programa

Este programa puede ampliarse de muchas formas. Si queremos ampliar el número de comentarios hará falta que el generador de números aleatorios cree números de dos o más cifras. Entre las líneas 065 y 074 puede observar cómo se hace. Pueden conseguirse 100 números aleatorios distintos. Si se desea menos de 100, pero más de 10, habrá que hacer algún que otro cálculo. Si el generador produce, por ejemplo, 100 resultados entre 00 y 99, puede dividirse cada número por 4 y se obtienen 25 posibles resultados entre 00 y 24. La orden de división es

M = INT (C/4)

C es un número aleatorio que debe sufrir una conversión, y M sólo puede ser un número entero. No obstante habrá que comprobar si todas las cifras aparecen con una frecuencia similar.

La comprobación es todo un problema

Ya que la estructura de nuestro programa es relativamente sencilla, la comprobación no debería ser un problema, si no hubiésemos introducido la generación de números aleatorios. Y ya que estos números no pueden preverse, todo test sistemático será muy costoso. Lo mejor será moverse con la orden TRACE instrucción a instrucción hasta que esté seguro que el mando para ocho vueltas funciona correctamente.

El generador de números aleatorios

A menudo, el BASIC de los ordenadores incorporan ya un generador propio de números aleatorios (buscar en el manual «RANDOM»; random significa «casualidad», «azar»). Si su BASIC no dispone de ninguno, utilice las líneas 900 a 950 como subrutina creadora de números aleatorios. Con las instrucciones de la línea 900 se consigue que el ordenador no llegue a presentar sólo ceros (lo que ocurriría si se multiplicara el número Q varias veces seguidas por sí mismo). La línea 920 corre la coma cinco posiciones hacia la derecha, y la 930 hace lo mismo con las cuatro primeras cifras, añadiendo al resultado un cero a la derecha. Si Q fuese, por ejemplo,

Q = 0,4246342533, tras la orden de 920 R = 42463,00. Tras la orden en 930, S tiene el valor 42460,00. Si en 940 hacemos N = R − S, N será 42463 − 42460 = 3. Por tanto disponemos de la sexta cifra 3, recortada de Q. Este será nuestro número aleatorio. Por tanto, si empezásemos siempre con la misma cifra Q, obtendríamos siempre la misma serie de números. No obstante, podemos evitarlo al pedir al jugador que introduzca (línea 030) una cifra cualquiera entre 0 y 0,3.

Algunos datos de comprobación

Aquí presentamos algunos datos de comprobación con sus correspondientes resultados que pueden ayudarle en la comprobación. Para T se introduce el cero. Con ello obtenemos el primer número aleatorio C = 10.

Por tanto hay que adivinar el 10:

Vuelta K	Casualidad N	Comentario	INPUT G	G?C
1	7	08	50	mayor
2	6	07	25	mayor
3	4	05	12	mayor
4	6	07	6	menor
5	7	08	9	menor
6	9	10	10	igual
ganado	1	12	¿nuevo juego?	

Listín electrónico de teléfonos

Prácticamente todo el mundo posee una colección de números de teléfono almacenados en algún sitio. Por ejemplo, al final del calendario de bolsillo.
Almacenar y buscar es una de las funciones principales de un ordenador. En consecuencia deberá estar capacitado para poder manejar un listín telefónico. En estas dos páginas encontrará media docena de programas unidos entre sí en un solo paquete de programas. Se trata de un listín telefónico. El paquete de programas ha sido reducido a su más mínima expresión para que pueda trabajar también en un computador de bolsillo en BASIC, aunque sólo puede almacenar y administrar diez nombres y números.
El paquete de programas tiene su razón de ser en el hecho de que no sólo queremos buscar números de teléfono, sino también tenerlo al día, extrayendo nombres y números, alterándolos o introduciendo nuevos. Quien quiera imprimir su listín telefónico en papel, dispondrá para ello también de un programa; sólo le hará falta una impresora.

El paquete de programas
Nuestro programa trabaja en un ámbito de datos que posee el nombre A$ y una longitud de 20. Almacenaremos no sólo nombres, sino también números de forma alfanumérica. La longitud del campo es de 7, de forma que no puede haber ningún nombre ni número con más de siete caracteres.
De A$ (1) a A$ (10) están los nombres, de A$ (11) a A$ (20) los números de teléfono correspondientes. En los campos de nombre no ocupados hay un grupo de ZZZZZZZ, y en el campo numérico correspondiente hay nueves (9999999). Con ello nos aseguramos mejor de que no quede campo sin rellenar por error.
Disponemos de los programas siguientes:

I — Información sobre el sistema.
II — Buscar número con nombre dado.
III — Buscar nombre con número dado.
IV — Introducir nuevo número.
V — Borrar dato actual.
VI — Imprimir todos los nombres con sus números.
VII — Inicialización de todos los datos. Todos los campos de nombre se ponen a «ZZZZZZZ», y todos los campos de números se ponen a «9999999».

La técnica de la búsqueda
La forma en que nuestros programas realizan la búsqueda es muy sencilla. Lo hace de forma que cada elemento es comprobado con el concepto buscado desde el punto de vista de la EQUIDAD, hasta que se tiene éxito. En la búsqueda con diez elementos se necesitan, en esta técnica, cinco pasos de búsqueda de promedio. Esta búsqueda puede programarse en BASIC con un bucle del tipo FOR-NEXT. Si se desea buscar en qué posición de A$ aparece el valor X$, el programa de búsqueda será:

```
120  FOR Z = 1 TO 10
130  IF X$ = A$(Z) THEN 180
140  NEXT Z
150  PRINT »NO CONSTA«
170  STOP
180  PRINT A$ (Z) (¡Encontrado!)
```

El contador del bucle Z es, al mismo tiempo, la posición en la tabla de búsqueda. La tabla que buscamos no tiene por qué estar ordenada —al contrario que en programas de búsqueda más largos, rápidos y complejos—. Pero si los valores de la tabla están ordenados alfabéticamente, podemos buscar de forma mucho más rápida.

¿Cuál es la longitud de la tabla?
Nuestra tabla tiene siempre la misma longitud. Podría buscarse, sin embargo, también en una tabla cuya longitud fuese desconocida. En ese caso debe establecer que el último campo sea reconocible por unos caracteres especiales —por ejemplo AAAAAAA—. Nuestro programa de búsqueda deberá gobernar por sí mismo el contador de bucle. Lo llamaremos nuevamente Z, al comienzo se pone a 1 y se aumenta de uno en uno con cada vuelta.

```
900  Z = 1
910  IF A$ (Z) = »AAAAAAA«
     THEN 950 (¿Final?)
920  IF A$ (Z) = X$ THEN
     970 (¿Encontrado?)
930  Z = Z + 1
940  GOTO 910
950  PRINT »¡Desconocido!«
     (¡Final!)
960  STOP
970  PRINT A$ (Z)
     (¡Encontrado!)
```

Programa I

```
005 REM »¿QUE PUEDE HACER EL PROGRAMA?«
010 PRINT »BUSCAR NUMERO CON NOMBRE
    DADO — RUN 100«
020 PRINT »BUSCAR NOMBRE CON NUMERO
    DADO — RUN 200«
030 PRINT »INTRODUCIR DATOS — RUN 300«
040 PRINT »BORRAR DATOS — RUN 400«
050 PRINT »LISTIN TELEFONICO — RUN 500«
060 PRINT »INICIALIZAR LISTIN — RUN 600«
070 STOP
```

Programa II

```
099 REM »BUSQUEDA DE NUMERO CON
    NOMBRE DADO«
100 INPUT »INTRODUZCA EL NOMBRE«;X$
110 PRINT »EL NOMBRE ES:«;X$
120 FOR Z = 1 TO 10
130 IF X$ = A$(Z) THEN 180
140 NEXT Z
150 PRINT »NOMBRE «;X$;« DESCONOCIDO«
160 PRINT »VUELVA A EMPEZAR«
170 STOP
180 Z = Z + 10
181 PRINT »EL NUMERO ES, «;A$(Z)
190 STOP
```

Programa III

```
199 REM »BUSCAR NOMBRE CON NUMERO DADO«
200 INPUT »INTRODUCIR NUMERO«;Y$
210 PRINT »EL NUMERO ES«;Y$
220 FOR Z = 1 TO 10
230 W = Z + 10
235 IF Y$ = A$(W) THEN 280
240 NEXT Z
250 PRINT »NUMERO «;Y$;« DESCONOCIDO«
260 PRINT »VUELVA A EMPEZAR«
270 STOP
280 PRINT »EL NOMBRE ES«;A$(Z)
290 STOP
```

Programa IV

```
299 REM »INTRODUCCION DE NUEVOS
    DATOS«
300 PRINT »ESPERO QUE QUEDE ALGUN
    SITIO LIBRE«
310 FOR Z = 1 TO 10
320 IF A$(Z) = »ZZZZZZZ« THEN 350
330 NEXT Z
340 PRINT »LO SIENTO, EL LISTIN ESTA
    LLENO«
345 STOP
350 INPUT »EL NOMBRE ES «;A$(Z)
355 Z = Z + 10
360 INPUT »EL NUMERO ES «;A$(Z)
370 STOP
```

Programa V

```
399 »BORRAR NUMERO«
400 PRINT »PROGRAMA DE BORRADO«
410 INPUT »INTRODUCIR NOMBRE A BORRAR«;X$
420 FOR Z = 1 TO 10
430 IF A$(Z) = X$ THEN 460
440 NEXT Z
450 PRINT X$;»NO CONSTA«
455 STOP
460 W = Z + 10
465 PRINT A$(Z);» «;AS(W);»BORRADO«
470 A$(Z) = »ZZZZZZZ«
480 A$(W) = »9999999«
490 STOP
```

Programa VI

```
499 REM »LISTIN TELEFONICO«
500 PRINT »LISTIN TELEFONICO«
510 FOR Z = 1 TO 10
515 IF A$(Z) = »ZZZZZZZ« THEN 530
520 PRINT A$(Z)
521 W = Z + 10
522 PRINT A$(W)
530 NEXT Z
540 PRINT »FINAL«
550 STOP
```

Programa VII

```
599 REM »BORRAR LISTIN ENTERO«
600 PRINT »BORRADO DEL LISTIN«
610 FOR Z = 1 TO 10
620 A$(Z) = »ZZZZZZZ«
630 A$(Z + 10) = »9999999«
640 NEXT Z
645 PRINT »BORRADO«
650 STOP
```

Programa I — »¿Que puede hacer el programa?«
Las seis órdenes de impresión entre las líneas 005 y 070 ayudan cuando no se sabe qué programa utilizar. Este dispositivo suele llamarse en los grandes ordenadores HELP o AYUDA. Aquí sólo hace falta escribir el RUN correspondiente.

Programa II — »Se busca número«
Se introduce un nombre (100) que por motivos de seguridad es repetido (110). Las líneas 120, 130 y 140 son el bucle de búsqueda: buscan si hay algún campo entre A$(1) y A$(10) con el mismo contenido del nombre introducido. Si no aparece, se da un mensaje (150) y se para (170).

Programa III — »Se busca el nombre«
Funciona de forma similar al programa II. Como Y$ se pide el número del teléfono. En la segunda parte es comparado con los que hay en A$ entre las posiciones 11 y 20. El programa imprime un comentario si no lo encuentra. En caso contrario ofrecerá el nombre que se encuentra en la posición del bucle (Z). Es decir, se busca con Z + 10, partiendo de A$(Z).

Programa IV — »Nueva introducción de datos«
En la línea 300 el programa avisa por si acaso no queda sitio y busca un lugar libre. Tomará el primer campo que encuentre con contenido »ZZZZZZZ«. Si el registro está lleno se da a conocer en 340. En caso contrario, el programa solicita en la línea 350 un nombre y en la 360 el número de teléfono correspondiente. El nombre va a A$(Z), y el teléfono se encontrará 10 posiciones más allá, en A$(Z + 10).

Programa V — »Borrar números«
Busca, mediante el nombre, la posición que debe ser borrada, escribe sobre el nombre »ZZZZZZZ« y sobre el número de teléfono »9999999«.

Programa VI — »Listín telefónico«
Sirve para imprimir el listín completo de teléfonos y en la línea 515 distingue si hay o no un nombre registrado. De esta forma imprime sólo los datos introducidos.

Programa VII — »Borrar listín«
Sólo se necesita cuando se almacena por primera vez un registro. Debe llamarse, por tanto, sólo una vez, cuando se vaya a realizar la primera introducción. Garantiza que el listín estará preparado para aceptar datos válidos y que no sucederán tonterías. Todos los campos de A$(1) a A$(10) son rellenados de ZZZZZZZ y los campos A$(11) a A$(20) son rellenados con 9999999. Ahora está todo en orden.

Comprobaciones sin problemas
Una vez introducido todo el paquete de programas comience con probar el programa VII, en caso necesario compruebe que al menos en A$(1) hasta A$(10) hay valores ZZZZZZZ. Luego compruebe si funciona el programa de impresión VI. Si funciona correctamente sólo debe imprimir »LISTIN TELEFONICO« y luego FINAL. Ahora le toca al programa IV, el de introducción de datos. Si funciona podrá introducir nombres y teléfonos y comprobar con el programa VI lo que ha entrado. Si su ordenador puede almacenar más de siete caracteres por alfabeto, amplíe consecuentemente las líneas 320, 470, 480, 515, 620 y 630. En el manual encontrará cuántos caracteres le caben. Compruebe allí también si puede ampliar A$ en más de 10. Dependerá de si su BASIC tiene la función DIM, con la que se pueden definir ámbitos mayores, y utilizarlos posteriormente.
Todos los programas deberán funcionar en ambos casos tanto si A$ se utiliza totalmente, como si se usa sólo en parte.

El prefijo
Con las siete cifras del número telefónico se las podrá arreglar con las cercanías. Pero si desea almacenar también los prefijos, deberá ampliar A$ en otras diez posiciones, para que entre A$(21) y A$(30) pueda guardar los prefijos. Después deberá introducirlo en el paquete de programas, lo cual exigirá mucho cuidado, pero resultará un estupendo ejercicio para aprender a alterar paquetes de programas. La alteración tendrá lugar en todos los programas y afecta también al programa VII.

El ordenador adivina cifras

En las páginas 14 y 15, el problema consistía en adivinar un número dado. Mostramos también cómo pueden completarse los diferentes pasos del programa con algunos comentarios originales y de aparición inesperada. El problema ahora es al revés. El ordenador adivinará el número entre 1 y 127 que usted pueda pensar, con menos de ocho preguntas. Como máximo, a la séptima vuelta ya lo habrá acertado. Esto dependerá, naturalmente, de su honestidad y no cambie de número a mitad del juego. El ordenador ya le advertirá sobre este aspecto. Detrás de todas estas preguntas se esconde algo importante. Si se sabe con cuántas preguntas puede adivinarse un número, podemos prever también los pasos de búsqueda necesarios para encontrar en una cierta cantidad o tabla un determinado elemento. En esta doble página encontrará el método.

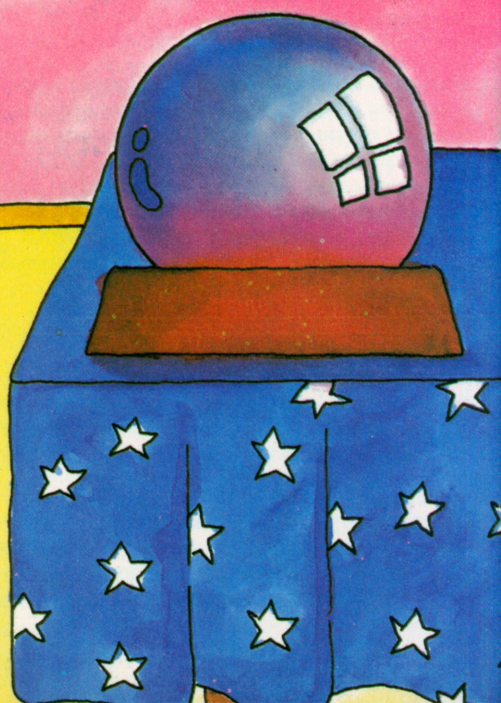
Así se busca

Cuando se trata de encontrar un número entre 1 y 3 preguntaremos primero, lógicamente, por el que está en medio. ¿Se trata del 2? Si no es el 2, la siguiente pregunta será: ¿Es mayor o menor que 2? Y así habremos llegado ya al final, pues 1 ó 3 son menor o mayor que 2.

Los números entre 1 y 7 se adivinan del mismo modo. Entre 1 y 7 está el 4. ¿Es 4? ¿No? Entonces viene la pregunta: ¿mayor o menor? Menor significa entre 1 y 3. Esto ya lo vimos antes. Mayor significa entre 5 y 7. Este problema también se soluciona con un máximo de dos preguntas. ¿Es el 6? y ¿mayor o menor?, lo que nos llevará al 5 ó al 7.

Esto suena complicado pero no lo es. Tome una hoja de papel y escriba los números 1 al 15 en una columna. Empiece por la mitad, en el 8. Suba hacia arriba, hacia la mitad entre 1 y 8, el 4, luego nuevamente a la mitad, etcétera. ¿Cuántos pasos necesita para alcanzar cada uno de los números? ¡Cuatro!

Si tiene 31 números necesitará cinco pasos, con 63 serán seis pasos.

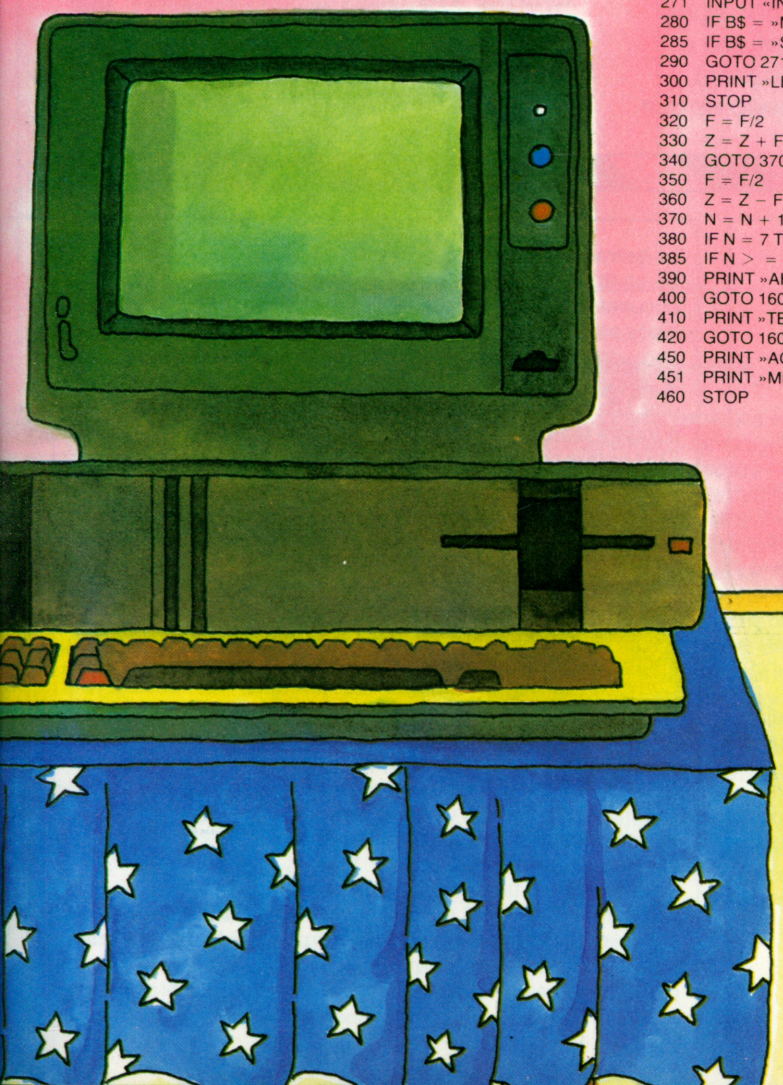


Encontrando un determinado número

Al buscar un determinado número entre 1 y 127 necesitaremos un máximo de siete pasos. Justo en la mitad entre 1 y 127 se encuentra el 64. El 64 puede dividirse en seis pasos:
$2 \times 2 \times 2 \times 2 \times 2 \times 2 = 64$. Para cada cifra creada por multiplicación de ella misma por 2 se aplica la norma de: cada nuevo 2 es un paso, una pregunta más. Al mismo tiempo se dobla la cantidad de elementos por los que se puede preguntar. Nuestro programa toma el 64 como centro entre 1 y 127. Si es »menor« tomará la mitad de 64 como nuevo centro. Si resulta ser »mayor« suma a 64 la mitad de 64. Es decir, que en el primer paso hay que tomar la mitad de 64 y sumarlo o restarlo —según las respuestas— al 64. El ordenador trabaja según este sistema hasta toparse con que es IGUAL, como máximo al cabo de siete pasos. O puede no hallarlo si el jugador le ha tomado el pelo jugando sucio.

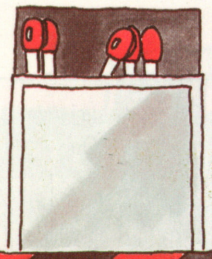
El programa

```
050  REM »EL ORDENADOR ADIVINA NUMEROS«
100  PRINT »POR FAVOR, JUEGUE LIMPIO«
101  PRINT »Y SEA HONESTO MANTENIENDO EL MISMO NUMERO«
102  PRINT »TENGO PLENA CONFIANZA EN USTED.
103  PRINT »ADIVINARE CON UN MAXIMO DE 7 PREGUNTAS«
104  PRINT »CUALQUIER NUMERO QUE USTED SE APUNTE«
105  PRINT »EN UN PAPEL, ENTRE EL 1 Y EL 127.«
110  PRINT »NO CAMBIE DE NUMERO MIENTRAS JUGAMOS.«
111  PRINT »Y AHORA OJO, QUE EMPIEZO...«
115  PRINT »COMO RESPUESTA AL NUMERO QUE YO LE DE, PULSE;«
116  PRINT »A PARA ACERTADO«
117  PRINT »G PARA DEMASIADO GRANDE«
118  PRINT »P PARA DEMASIADO PEQUEÑO«
150  Z = 64
151  N = 1
152  F = 64
160  PRINT »ES EL NUMERO «;Z;« POR CASUALIDAD?»
170  PRINT »DIGAME QUE TAL VOY»
180  INPUT A$
185  PRINT «USTED ME DICE»;A$
200  IF A$ = »A« THEN 260
210  IF A$ = »P« THEN 320
220  IF A$ = »G« THEN 350
230  PRINT »ME ASEGURO QUE JUGARIA LIMPIO!«
240  PRINT »INTRODUZCA A, G o P.«
250  GOTO 180
260  PRINT »GANE CON LA PREGUNTA NUMERO »;N;«, SOY
     UN GENIO!«
270  PRINT »QUE, LO INTENTA OTRA VEZ?«
271  INPUT «INTRODUZCA SI O NO«;B$
280  IF B$ = »NO« THEN 300
285  IF B$ = »SI« THEN 110
290  GOTO 271
300  PRINT »LE ABANDONO EL VALOR, ¿EH?«
310  STOP
320  F = F/2
330  Z = Z + F
340  GOTO 370
350  F = F/2
360  Z = Z - F
370  N = N + 1
380  IF N = 7 THEN 410
385  IF N > = THEN 450
390  PRINT »AHORA TOCA LA PREGUNTA NUMERO«;N
400  GOTO 160
410  PRINT »TEORICAMENTE DEBERIA ESTAR YA MUY, MUY CERCA«
420  GOTO 160
450  PRINT »AQUI HAY ALGO QUE NO FUNCIONA«
451  PRINT »ME RINDO, DE USTED NO HAY QUIEN SE FIE«
460  STOP
```



Un juego llamado NIM

El juego de NIM es tan antiguo como sencillo. Para poder jugar hacen falta dos personas y un puñado de cerillas. Antes de empezar hay que acordar cuántas cerillas tendrá el montón. Entonces los jugadores toman cada uno, de forma alternativa, algunas cerillas del montón —como mínimo una. Al inicio del juego se ha acordado también cuál es el número máximo de cerillas que pueden tomarse. El que se lleve la última cerilla pierde. (Esta es la versión que hemos elegido para el ordenador, pues hay otras versiones, con dos e incluso con tres montoncitos).
La atracción de este juego es que puede jugarse con una estrategia que hará perder a quien no la conozca, siempre que no se hagan trampas, claro. Este tipo de estrategia puede también programarse y jugar contra un oponente eléctrico, como es el ordenador. En estas dos páginas puede ver qué aspecto tiene un juego de este tipo y cómo se hace el programa.

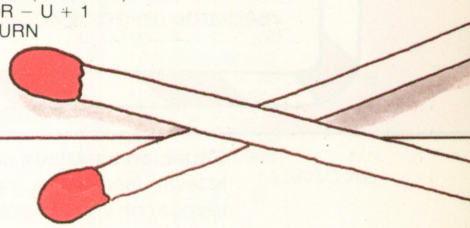


Construimos un juego de ordenador
En la configuración de un juego de computadora se introducirá toda la ayuda y buena presentación posibles, con comentarios y titulación para proporcionar un poco de desenfado y diversión. Al comienzo del juego, el ordenador dará a conocer las reglas del juego y exigirá las cifras o datos que debe dar el jugador —que aquí será la cantidad total de cerillas y la cantidad máxima de cerillas que pueden cogerse por jugada. El ordenador deberá comprobar si las cantidades dadas son correctas o si el contrincante está introduciendo algo que no resulte limpio.
Entonces hay que decidir quién empieza. Lo más honesto es ceder el turno al contrincante. Perderá de todas formas si no conoce la estrategia. En la tercera parte de nuestro programa, entre las líneas 199 y 310, juega el contrincante. Entre las líneas 319 y 480 juega el ordenador. La estrategia del ordenador se encuentra en la línea 320. Hay una posibilidad que no cubre el programa, y ahí está el punto flaco del ordenador. Para confundirnos trabaja con un número aleatorio generado por una subrutina (GOSUB).
En todos aquellos lugares en que se puede pescar al contrincante haciendo cochinas trampas no habrá que ahorrarse ningún comentario «picante» para avisar.

La estrategia del éxito
Su contrincante perderá cuando en su penúltima jugada se encuentre con dos puntos o cerillas que la cantidad máxima fijada para retirar cerillas. Si toma el máximo M, quedarán dos para usted. Usted coge una de las dos y su contrincante cogerá la última —y perderá. Si en esta penúltima jugada coge sólo una. Usted deberá coger M cerillas, con lo que al contrincante le quedará también sólo una.
Coja lo que coja el contrincante, usted debe igualar la cifra para que los dos juntos hayan cogido M + 1 cerillas. Usted conoce ya la cantidad total S. La última cerilla será para el contrincante, por lo que usted contará con S − 1. Esta cantidad la divide en paquetes de M + 1. Tomemos por ejemplo que M fuese 4 y que S, la cantidad total, fuese de 22. S − 1 = 21 dividido en paquetes de 5 (M + 1). Obtenemos así cinco paquetes y sobra una cerilla, que será la que cogerá usted al comenzar. Haga lo que haga ahora su compañero, usted lo completa hasta cinco. Y gana.
Con esta estrategia hay que empezar pronto. A ser posible ya en el comienzo, pues si no podría ser demasiado tarde (y además debería hacer muchos cálculos mentales). La estrategia tiene, no obstante, una escapatoria, un punto débil, en cuanto le toca a usted como segundo en la partida y se encuentra ya con un montón en el que sobra una cerilla para ser múltiplo de M + 1. Si su contrincante también conoce este juego, mala suerte. Si ha cogido bien las cerillas sólo por casualidad, tendrá que vigilar cuándo podrá tomarle la delantera (con M + 1). En ese caso podrá incluso calcular en cuantas partidas habrá ganado.

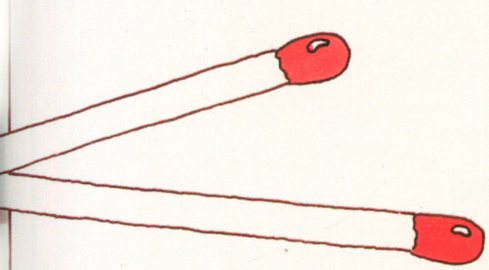
El programa

```
 10 REM »EL JUEGO DE NIM«
 11 REM »GENERADOR DE NUMEROS ALEATORIOS«
 12 PRINT »INTRODUZCA UN NUMERO ENTRE 0 y 0,3«
 13 INPUT T
 14 Q = 0,3141592654 + T
 19 REM »REGLAS DEL JUEGO Y PREPARACION«
 20 PRINT »DE UN MONTONCITO DE CERILLAS TOMAREMOS
    CADA«
 21 PRINT »UNO COMO MINIMO UNA, PERO COMO
    MAXIMO UNA«
 22 PRINT »CANTIDAD QUE MARCARA USTED POR
    ADELANTADO.«
 23 PRINT »EL QUE COJA LA ULTIMA CERILLA HABRA
    PERDIDO«
 30 PRINT »CUANTAS CERILLAS DEBE TENER EL MONTON?«
 32 INPUT »CERILLAS ;«;S
 33 PRINT S;» CERILLAS«
 90 IF (S > = 20) THEN 120
100 PRINT »POR LO VISTO TIENE MUCHA PEREZA«
120 PRINT »Y CUANTAS PUEDEN COGERSE COMO MAXIMO?«
130 INPUT »MAXIMO ?«;M
140 IF (M > = 2) THEN 170
150 PRINT »ESTE TRUQUITO ESTA YA MUY VISTO«
160 GOTO 120
170 IF (S > M + 1) THEN 200
180 PRINT »EL MONTONCITO DEBERIA SER ALGO MAYOR«
190 GOTO 30
199 REM »EL JUGADOR JUEGA«
200 PRINT »AHORA LE TOCA A USTED«
210 INPUT »SU JUGADA? «;Z
220 IF (Z > 0) THEN 230
221 PRINT »VERGÜENZA DEBERIA DARLE, INTENTAR
    HACERLE«
222 PRINT »TRAMPAS A UN ORDENADOR«
223 PRINT »INTRODUZCA ALGUN VALOR DIFERENTE
    DE CERO«
224 GOTO 200
230 IF (Z < = M) THEN 260
240 PRINT »EH, EH, EH, A VER SI NOS COMPORTAMOS...«
250 GOTO 200
260 S = S - Z
261 PRINT »USTED QUITA «;Z« DEL MONTON«
262 PRINT »CON LO QUE NOS QUEDAN «;S
270 IF (S > = 1) THEN 320
280 IF (S = 0) THEN 460
290 PRINT »HAGA EL FAVOR DE RESPETAR LAS
    NORMAS«
300 S = S + Z
310 GOTO 200
319 REM »AHORA LE TOCA AL ORDENADOR«
320 Z = (S - 1) - INT((S - 1)/(M + 1))*(M + 1)
330 IF (Z < > 0) THEN 370
340 IF (M < = S) THEN LET V = M; GOTO 342
341 V = S
342 GOSUB 800
343 IF (N < = V) THEN 346
344 N = N - V
345 GOTO 343
346 Z = N
350 PRINT »ME ESTA HACIENDO LA VIDA DIFICIL«
360 GOTO 380
370 PRINT »NO PARECE TENER GANAS DE
    GANAR, ¿EH?«
380 S = S - Z
390 PRINT »YO TOMARE «;Z;» DE LAS QUE
    QUEDAN«
391 PRINT »Y SOLO NOS QUEDAN »; S
440 IF (S = 0) THEN 480
450 GOTO 200
460 PRINT »QUERIDO AMIGO, HA PERDIDO«
470 GOTO 490
480 PRINT »ES PROBABLE QUE HAYA GANADO«
485 PRINT »HABRA HECHO TRAMPAS?«
489 REM »FIN DE JUEGO«
490 PRINT »¿OTRA PARTIDITA?«
491 PRINT »SI = 1, NO = 0«
500 INPUT »SE ATREVE? »;E
510 IF (E = 1) THEN 30
511 PRINT »BUENO, PUES ENTONCES NADA«
512 STOP
799 REM »GENERADOR DE NUMEROS
    ALEATORIOS«
800 IF (Q < = 0,4) THEN LET Q = Q + 0,35
810 Q = Q*Q
820 R = INT(Q*100000)
830 U = INT(Q*10000)*10
840 N = R - U + 1
850 RETURN
```



Así se calcula de forma estratégica

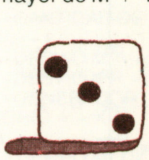
En nuestro programa, la estrategia se encuentra entre las líneas 320 y 380. Se divide S − 1 entre M + 1 y se multiplica la proporción m + 1. Si todo ello se resta de S − 1 nos queda la cantidad de cerillas que le falta al M + 1 restante. Con esta cifra se hace Z, el número a deducir. Si no hay resto de dicha división, es decir Z = 0, la Z toma el valor de un número



aleatorio entre 1 y M (o, si S es menor de M, entre 1 y S). Además: una vez se ejecute la instrucción de la línea 370 y no se cometan más fallos, se volverá siempre a la línea 370.

Comentarios sobre el programa

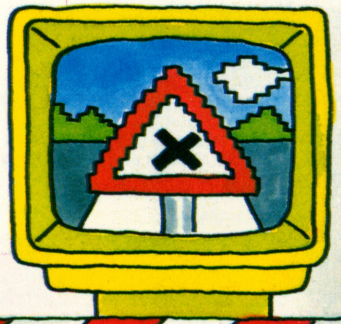
El programa tiene una estructura muy sencilla y es fácil de comprender si se ha captado la estrategia que rige el juego. Hasta la línea 14 se prepara el generador de números aleatorios. Entre 19 y 23 se encuentran las reglas del juego. Entonces, mediante introducción por el teclado, se indican cuántas cerillas tiene que haber. Este número lo llamamos S y se reducirá constantemente hacia cero. M es la cantidad de cerillas que pueden cogerse de una vez como máximo. En 130 se solicita dicha cifra. M = 1 no se acepta. El programa le reñirá también si S no es mayor de M + 1. Entre 199 y 310 se analiza lo que introduce el jugador. Si introduce un cero, bronca al canto, al igual que si pretende tomar más de lo permitido. Finalmente se hacen los cálculos pertinentes y se presentan en pantalla (260 a 262). Si se han tomado más cerillas de las que había, aterrizamos en la línea 290 y se anula la jugada.

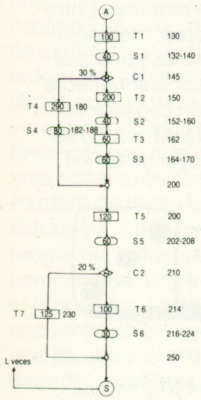


A partir de la línea 320 juega el ordenador. Entre 490 y 512 se pregunta si queremos continuar jugando, y a partir de 800 está el generador de números aleatorios.

Simulación por ordenador

Una de las aplicaciones del ordenador es su capacidad para la simulación de procesos reales. Aquí hay un ejemplo de este tipo ya programado. Se trata del tráfico urbano. Abajo a la izquierda puede ver el desarrollo. Se conduce a A hacia S. Las cajas rectangulares representan tramos de vía libre. Las cifras introducidas son los segundos que dura el trayecto. Las cajas redondeadas representan semáforos. Si en una de estas cajas hay un 40, es que el semáforo estará como máximo 40 segundos en rojo. Es posible cualquier tiempo de espera entre 0 y 40, según el momento en que se llegue al semáforo. Entonces hay todavía otros dos cruces. M y N indican el porcentaje de coches que giran. Cada trayecto a través de estas calles es distinto. En la simulación se determinan los tiempos de espera mediante un generador de números aleatorios. Los tiempos de espera y de trayecto se suman. En los cruces aparece un número aleatorio entre 1 y 100. Si es menor que M o que N giraremos; si es mayor seguiremos rectos.
Es importante calcular muchos casos para poder determinar una media.





T = Tramo recto (vía libre)
S = Semáforo
C = Cruce

Situación simulada de tráfico urbano

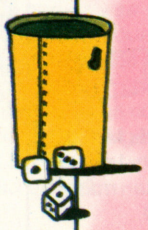
Nuestro ejemplo se compone de dos cruces, siete tramos con vía libre entre 60 y 290 segundos, y seis semáforos con tiempos de espera máximos entre 30 y 80 segundos. Los cruces son flexibles, es decir que puede indicarse el porcentaje de los casos en que se girará. Cada camino individual es una sana mezcla de tramos libres y tiempos de espera. El tiempo mínimo es de 580 segundos y el máximo de 810.

La casualidad viene impuesta por un generador de números aleatorios que produce, en cuanto se le solicita, un número entre 00 y 99, es decir, un número de dos cifras. Si debe ser menor de 40, se restará tantas veces como sea necesario hasta que se dé el número requerido. Por debajo de 20, el generador es algo parcial e incompleto, ya que al restar 40, los números entre 80 y 100 aportan un desequilibrio. Si ello nos molesta, podemos hacer que el generador ignore todos aquellos números mayores de 80. No obstante, dicho ejercicio significa mayor pérdida de tiempo, pero aquí sólo nos interesa el principio, aceptaremos este desequilibrio.

El programa de simulación

Nuestro programa no es complicado. Solicita las cifras para los porcentajes de desvío M y N, y pide información sobre el número de vueltas (1) que debe simular. También pide una cifra para el generador de números aleatorios, ya que en caso contrario realizaría siempre la misma serie de números aleatorios. Un bucle con el contador K ejecuta las instrucciones entre 106 y 252 L veces. En la figura de la izquierda se han indicado los números de las instrucciones que rigen cada uno de los puntos y cómo vuelven a juntarse los desvíos.

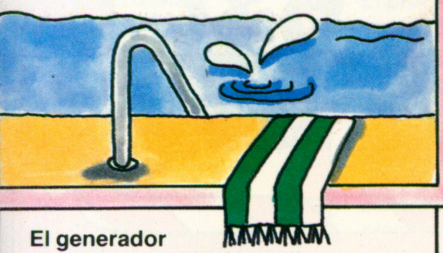
En un caso de simulación hay que saber con qué cifras se ha trabajado. Estas cifras se llaman, en terminología técnica, «parámetros». Algunos de estos parámetros están ya fijos en el programa. M, N, L y el generador de números aleatorios son los parámetros que deben introducirse. Por ello se imprimen o presentan en pantalla junto con la duración media del trayecto. De esta forma pueden compararse muchos casos y aclarar, mediante la simulación, si es mejor pasar, por ejemplo, por el tramo de vía libre o continuar recto.



El método heurístico

El matemático griego Arquímedes descubrió el principio de que un cuerpo sumergido en agua pierde tanto peso como el agua que desplaza. Esta idea se le ocurrió mientras se bañaba. Se entusiasmó tanto que salió a la calle totalmente desnudo gritando «¡Eureka, ya lo tengo!». Por este motivo, los métodos que muestran su solución de forma demostrativa se denominan hoy métodos heurísticos.

La simulación es un típico método heurístico. Como puede verse en nuestro ejemplo, se trata siempre de una combinación entre parámetros ya introducidos y fijos y otros que hay que darle al ordenador, todo ello junto a una configuración modélica básica. Si hacemos M y N igual a cero, el modelo no tendrá cruces. Si se desean más cruces habrá que programarlo en otro modelo.



El generador de números aleatorios

Nuestro generador de números aleatorios, con el valor base para Q de 0,314259621, nos suministra los siguientes primeros diez valores entre 00 y 99:

10, 57, 56, 4, 76, 37, 9, 21, 64 y 63

Si su versión de números

aleatorios bien introducida da también estos valores, puede calcular lo que saldrá para L = 1, es decir para la primera vuelta por nuestro circuito automovilístico. Para M = 30, N = 20 y L = 1, nuestro ordenador nos dio un tiempo de conducción de 635 segundos.

Simulación y tiempo de cálculo

Los modelos de simulación son típicos devoradores de tiempo de cálculo por tener programas largos y porque no siempre se contentan con tres pasadas, sino que a menudo necesitan miles. A ello hay que añadir variaciones en los parámetros. Por ello, los modelos de simulación suelen encontrar aplicación en los ordenadores más grandes.

Observaciones sobre el programa

El programa se divide en tres partes, al igual que la mayoría de los programas de simulación. En la primera parte se establecen todos los parámetros necesarios. Esto se realiza entre las líneas 010 y 054. A esta primera parte hay que añadir también la organización del bucle con la puesta a cero de los campos contadores. K cuenta las vueltas hasta L. T suma el tiempo total de conducción por vuelta, es decir que con cada vuelta se pone a cero. Y en S se almacenan los valores de T. La segunda parte del programa es el modelo en sí de las distintas vueltas. Se encuentra entre 120 y 250.

A partir de 270, en la tercera parte del programa, se realizan los cálculos. Se imprimen los valores de L, M, N y X, así como el tiempo promediado de conducción S/L. Entre 900 y 906 se encuentra, a modo de subrutina, el generador de números aleatorios. Puede que algunos se pregunten por qué no hemos ubicado el bucle principal (K/L) en un FOR-NEXT. El motivo está en que nuestro ordenador no pudo con él, debido, quizá, a que dentro de este bucle hay numerosos GOSUB. Por ello tememos que un bucle del tipo FOR-NEXT podría suponer alguna dificultad también en su computadora.

El propio modelo

En un modelo propio de circulación podría ocurrírsele que, incluso en la conducción en vías libres, el tiempo oscila ligeramente. Para poder hacer justicia a dicha condición le hará falta un generador adicional de números aleatorios.

Comprobación del simulador

Los programas de simulación suelen ser difíciles de comprobar. Debido, por un lado, a que trabajan con un generador de números aleatorios que se aplica con bastante frecuencia; y, por otro, a la intensidad y volumen de sus cálculos. Lo mejor es si podemos analizarlos instrucción por instrucción. Pruebe primero el generador de números aleatorios y anótese los diez primeros valores que produce.

En nuestro generador de números aleatorios obtenemos, para M = 0 y N = 0 (sin giro en los cruces) y con X = 0 en una vuelta (L = 1), un tiempo de trayecto de 635 segundos. Y para los valores X = 0, M = 100, N = 0 y L = 1 el valor 687. En el caso alternativo (M = 0, N = 100, L = 1 y X = 0) el valor es de 651.

El programa

```
010 REM »SIMULACION
    TRAFICO«
020 INPUT »DESVIO M «;M
025 PRINT »M = «;M
030 INPUT »DESVIO N «;N
035 PRINT »N = «;N
040 INPUT »NUMERO DE
    VUELTAS «;L
045 PRINT »L = «;L
050 PRINT »UNA CIFRA ENTRE
    0 y 0,3«
051 INPUT »GENERADOR
    ALEATORIO «;X
052 PRINT »GENERADOR
    ALEATORIO; «;X
054 Q = 0,314159625 + X
100 S = 0
105 K = 0
106 K = K + 1
107 IF K > L GOTO 270
120 T = 0
130 T = T + 100
132 GOSUB 900
134 IF Z < = 40 GOTO 140
136 Z = Z - 40
138 GOTO 134
140 T = T + Z
145 GOSUB 900
147 IF Z < M GOTO 180
150 T = T + 200
152 GOSUB 900
154 IF Z < = 40 GOTO 160
156 Z = Z - 40
158 GOTO 154
160 T = T + Z
162 T = T + 60
164 GOSUB 900
166 IF Z < = 60 GOTO 170
168 Z = Z - 60
170 T = T + Z
172 GOTO 200
180 T = T + 290
182 GOSUB 900
184 IF Z < = 80 GOTO 188
186 Z = Z - 80
188 T = T + Z
200 T = T + 120
202 GOSUB 900
204 IF Z < = 60 GOTO 208
206 Z = Z - 60
208 T = T + Z
210 GOSUB 900
212 IF Z < N GOTO 230
214 T = T + 100
216 GOSUB 900
218 IF Z < = 30 GOTO 224
220 Z = Z - 30
222 GOTO 218
224 T = T + Z
226 GOTO 250
230 T = T + 125
232 GOTO 250
250 S = S + T
252 GOTO 106
270 PRINT »CANTIDAD DE
    TESTS = «;L
280 PRINT »DESVIO M = «;M
290 PRINT »DESVIO N = «;N
295 PRINT »GENERADOR
    ALEATORIO = «;X
300 U = S/L
310 PRINT »TIEMPO MEDIO DE
    CONDUCCION = «;U
312 PRINT »FIN DE TEST«
315 STOP
900 IF Q > = 0,4 GOTO 902
901 Q = Q + 0,35
902 Q = Q*Q
903 V = INT(Q*100000)
904 W = INT(Q*1000)*100
905 Z = V - W
906 RETURN
```

¿Pueden pasarse números por un colador?

Esta pregunta puede parecer algo cómica, y tiene naturalmente significado metafórico. Hay una historia sobre un malévolo rey que encerró a la mitad de sus súbditos en mazmorras. La otra mitad la utilizó como carceleros.
Con el tiempo, el rey vio que de esa forma no había quien reinara. Así ordenó al matemático de la Corte que ideara un sistema mediante el cual se liberaran algunos súbditos.
El matemático real puso manos a la obra y presentó al rey el siguiente proceso: cada prisionero tiene una celda. Estas celdas están numeradas de forma consecutiva. El primer carcelero abre todas las puertas. El segundo se encarga de cada segunda puerta —de la puerta 2, 4, 6, etc. Si la encuentra abierta la cierra, y si la encuentra cerrada la abre. El tercero comienza por la puerta número 3 y se ocupa de cada tercera puerta, etc. ¿Qué puertas quedarán al final abiertas, dejando en libertad a sus ocupantes? Si a todo este proceso lo llamamos «calador» (o tamiz, criba, cedazo, etc.), la pregunta sería: ¿Qué números se quedan en el colador?

El colador electrónico del ordenador

Si queremos aplicar la historia del colador de números al ordenador habrá que desarrollar primero un modelo. Necesitamos una representación apropiada para la cárcel y debemos estar en situación de imitar el proceso de las puertas.
Lo más sencillo será dar una variable a cada puerta. Allí almacenamos un cero para puerta cerrada y un uno para puerta abierta.
Mejor aún será trabajar con un «ámbito numérico». Este consiste en diferentes variables con un nombre conjunto. Si nos referimos a un número de variable en el ámbito A, escribiremos A(24), 24 es un «índice»; se dice que las variables son «indexadas».
Por lo tanto, abriremos en nuestro programa un ámbito numérico de este tipo, tras indicar su longitud con una orden INPUT. Si este número es B, nuestras mazmorras tendrán B celdas y B carceleros.
Seguidamente trabajaremos el ámbito numérico con un bucle FOR-NEXT, poniendo a cero los unos y a uno los ceros, tal como lo exige la idea del colador de números.
Al final pasaremos nuevamente por un bucle FOR-NEXT a través de todo el ámbito numérico. Si un valor determinado es cero, pasaremos sin hacer caso: la puerta está cerrada. Si el valor es uno, imprimiremos el valor del índice. ¿Qué números aparecerán entonces en la pantalla?

Un programa que «cuela» números

Ya que es posible que queramos probar varios coladores, dividiremos el programa en tres partes. En la primera se fija la longitud del colador B. Todas las cifras en el ámbito A, que debe representar el modelo de la cárcel, se ponen a cero. En la segunda parte se programa el colador en sí.
Si más tarde queremos programar otros tipos de colador y comprobarlos, sólo necesitaremos sustituir esta parte del programa.
En la tercera parte se realizan los cálculos. Se trata de un bucle que corre de 1 a B. Si A(B) = 0 no pasa nada. Si A(B) = 1, se imprime la B.
El programa tendrá una elevada intensidad de cálculo, pues debe pasar muchas veces por el ámbito numérico A.

El colador de Eratóstenes

El griego Eratóstenes (276-194 a.C.) inventó otro tipo de colador. El primer carcelero abre todas las puertas. El segundo deja su puerta abierta y cierra todas las segundas puertas, etc. Todo prisionero que encuentre su puerta abiera, la deja abierta, pero cierra todas aquellas cuyo número sea múltiplo del número de su puerta. Todo prisionero que encuentra su puerta cerrada podrá marcharse. Lo que resulta de ello puede casi adivinarse. Las puertas abiertas serán aquellas cuyo número sólo pueda dividirse por 1 y por sí mismo sin que quede resto. Pero estos números son los números primos. El colador de Eratóstenes sirve, así, para determinar los números primos.

¿Con o sin DIM?

El ordenador BASIC en el que hemos probado estos programas es un ordenador de bolsillo del tipo SHARP PC 1211. Posee una impresora y domina el BASIC original, aunque sin la práctica sentencia DIM (»dimensión»). Sólo puede indexarse la variable A y la cantidad dependerá del espacio que quede en la memoria después del programa. Utilizamos la A indexada a partir de la posición 11, ya que en el programa necesitamos algunas otras variables. Por ello, en las líneas de instrucciones 260 y algunas otras aparece la constante 10 en la zona indexada.
Si su ordenador es, no obstante, mayor y puede trabajar bien con memorias fijas RAM (junto a la sentencia DIM que proporciona espacio en la memoria dimensionándola previamente), podrá entonces indexar ámbitos numéricos mayores. En muchos ordenadores personales (PC) puede utilizarse este sistema libremente hasta el borde de la memoria, es decir, hasta que el programa y los datos hayan ocupado totalmente la zona de memoria RAM.
A menudo puede realizarse esta indexación sin problemas hasta el índice 10 aproximadamente. Pero entonces hay que decirle al ordenador el tamaño que tendrá el ámbito numérico y cuál es su nombre. En caso contrario avisará, a lo largo del programa y en momentos muy poco apropiados, que se ha superado el ámbito de memoria permisible y que se siente frustrado.

El programa para el colador

```
200 REM »EL COLADOR Q«
210 PRINT »INTRODUZCA LA
    LONGITUD DEL COLADOR«
220 INPUT »LONGITUD DE
    COLADOR = «; B
230 PRINT »LA LONGITUD ES«; B
250 FOR C = 1 TO B
260 A(C + 10) = 0
270 NEXT C
300 FOR C = 1 TO B
310 D = 1
315 E = D*C + 10
320 IF A(E) = 0 THEN LET
    A(E) = 1: GOTO 340
330 A(E) = 0
340 D = D + 1
350 IF ((D*C - 1) >B) THEN 370
360 GOTO 315
370 NEXT C
600 FOR C = 1 TO B
610 IF A(C + 10) = 0 THEN 630
620 PRINT C
630 NEXT C
640 PRINT »FIN EN«; B
650 STOP
```

El programa para el colador de Eratóstenes

```
200 REM »COLADOR DE
    ERATOSTENES«
210 PRINT »INTRODUCIR
    LONGITUD DEL
    COLADOR«
220 INPUT »LONGITUD DE
    COLADOR = «; B
230 PRINT »LA LONGITUD
    ES«; B
250 FOR C = 1 TO B
260 A(C + 10) = 1
270 NEXT C
300 FOR C = 2 TO B
310 IF A(C + 10) = 0 THEN 370
320 D = 2
330 A(C*D + 10) = 0
340 D = D + 1
350 IF ((D*C) >B) THEN 370
360 GOTO 330
370 NEXT C
600 FOR C = 1 TO B
610 IF A(C + 10) = 0 THEN 630
620 PRINT C
630 NEXT C
640 PRINT »FINAL EN«; B
650 STOP
```

El programa para el señor Eratóstenes

Juegue primero con el colador del malévolo rey. Entonces no tendrá problema en adaptar el segundo programa en el primero. Para variar se cargan ya en la fase de preparación todas las celdas con un uno. En la parte del colador comenzamos por la puerta 2. Por lo demás, los programas no se diferencian; ambos tienen una estructura relativamente sencilla.

Para los mayores

Hemos mantenido también este ejemplo conscientemente con una estructura muy sencilla y modesta desde el punto de vista de programación, pues nuestra intención es que funcione en el más pequeño ordenador BASIC (válido para todos los programas de este libro). Para aquellas personas con ordenadores mayores se les presenta aquí la gran oportunidad. Tan sólo la cuestión de si »¿Es N un número primo?« o la de »A ver cuántos números primos puedo conseguir con mi ordenador« tienen ya un gran atractivo y plantean todo un reto.
Lo mismo es válido para números elevados al cuadrado. Otro deporte para el intelecto, prácticamente insoluble incluso para un ordenador: ¿Cuál es el número primo más pequeño divisible por 13?

Eratóstenes sobre el banco de pruebas

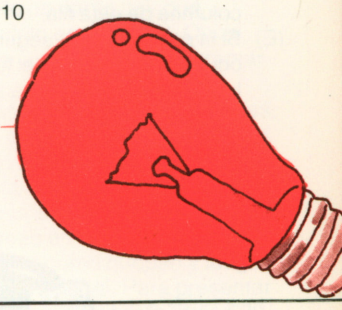
Los programas como el nuestro son más lógicos que calculadores. Por ello hay que comprobarlos con especial rigor. Lo mejor es tomar una longitud determinada de colador, por ejemplo, 10, y analizar paso a paso el desarrollo y funcionamiento del programa. Aquí hay algunos datos de comprobación.

Prisioneros (Pasos)	Puertas = Contenido de A									
	1	2	3	4	5	6	7	8	9	10
Comienzo	1	1	1	1	1	1	1	1	1	1
2	1	1	1	0	1	0	1	0	1	0
3	1	1	1	0	1	0	1	0	0	0
4	1	1	1	0	1	0	1	0	0	0
5	1	1	1	0	1	0	1	0	0	0
6	1	1	1	0	1	0	1	0	0	0
7	1	1	1	0	1	0	1	0	0	0

De este caso en forma de test podemos comprobar que no hay más alteraciones como máximo a partir de la mitad del colador de números. Ya que el programa tiene una elevada intensidad de cálculo, vale la pena tenerlo en cuenta. Para gente avispada: puede pararse tras el número siguiente a la raíz cuadrada de la longitud del colador.

El secreto del rey

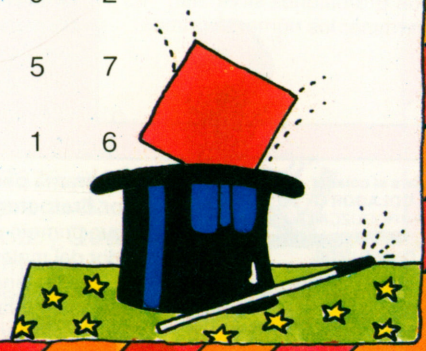
El colador del rey malo nos da los cuadrados de los números entre 1 y la longitud del colador. Y ese es, ni más ni menos, el secreto del rey. Para comprobar: entre 1 y 10 se encuentran los cuadrados 4 y 9. ¿Qué cuadrados se encuentran entre 1 y 100? Y, ¿puede su ordenador determinar si el 121 es un cuadrado?.



El cuadrado mágico

Un cuadrado mágico es una disposición cuadrada de números. Al sumar los números de cualquier línea, columna o diagonal nos sale siempre la misma cantidad. A la derecha tenemos un pequeño ejemplo: las columnas, líneas y diagonales suman todas 15. ¿Puede un ordenador componer un cuadrado mágico? Muchos dirían sin dudarlo un instante que no. Pero sí puede, aunque sólo sean cuadrados con un número impar de celdillas, es decir 9, 25, 49 ó 81. Nadie sabe quién inventó la forma de construir cuadrados mágicos, pero la información puede extraerse de uno de los libros del maestro del cálculo Adam Riese, que vivió de 1492 a 1559. El maestro Riese nos ha dado normas para crear cuadrados mágicos que pueden ser programadas.

4	9	2
3	5	7
8	1	6



Las reglas de Riese para el cuadrado

(A) La construcción (se trata de un cuadrado con un número impar de lado) comienza con el elemento justo debajo del elemento central.

(B) El elemento siguiente ocupará la casilla una posición inferior y una posición a la derecha —siempre que esta posición no esté ya ocupada por otro número. Esta regla se llama regla Sud-Este.

(C) Si el punto reace fuera de la matriz según la regla Sud-Este, nos encontraremos en la columna correcta, pero el número deberá ir en la primera fila de esta columna.

(D) Si el punto según la regla Sud-Este recae a la derecha de la matriz, nos encontraremos en la fila correcta, pero el número deberá ir en la primera columna de esta fila.

(E) Si la posición está, según la regla Sud-Este, ocupada, se partirá del lugar ocupado uno a la izquierda y uno hacia abajo.

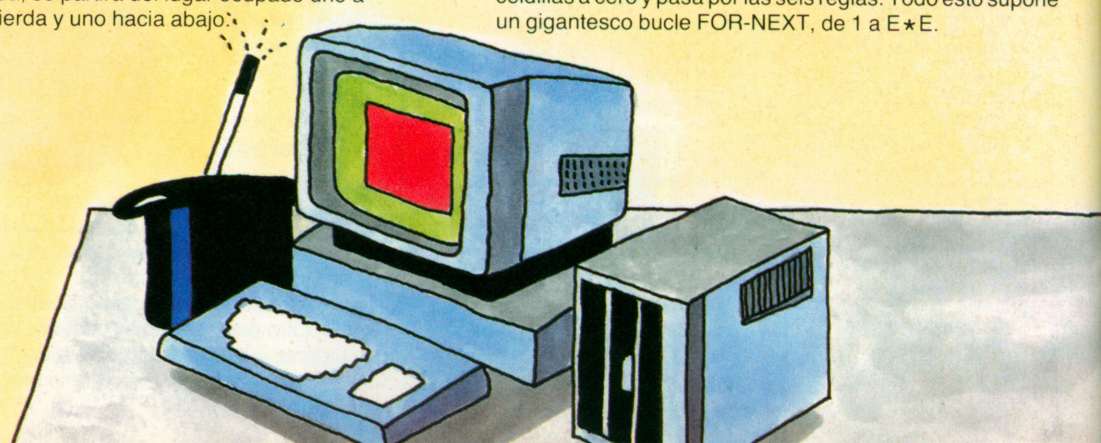
(F) Si nos dirigimos según B fuera de la matriz por la diagonal, la posición será en la última columna, segunda fila empezando por arriba.

Estas reglas proceden de Adam Riese y funcionan siempre.

El cuadrado en el ordenador

Para poder componer y almacenar un cuadrado de 5 de lado harán falta 25 celdillas en la memoria. La forma más sencilla es almacenar los 25 números uno detrás de otro. Entonces le daremos a este ámbito numérico (también llamado «vector») un nombre, por ejemplo A. Los diferentes números se obtienen por indexación. A(17) es el decimoséptimo número. Si N = 3, A(N) será el tercer número.

El programa comprueba si la longitud del lado (E) no es demasiado grande y si este número es par, pone todas las celdillas a cero y pasa por las seis reglas. Todo esto supone un gigantesco bucle FOR-NEXT, de 1 a E*E.



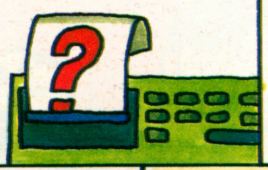
Más sobre el programa

Desde el inicio del programa hasta la línea 250 se comprueba que la longitud del lado, leída en 225, sea menor de 10 y sea impar. Entonces, entre 300 y 325 se rellena el ámbito numérico A con E*E ceros. La dirección de todas y cada una de las celdillas es el resultado del contador (I) para las filas y del contador de las columnas (c) calculado en la línea 310 como G. En A se almacena una línea del cuadrado tras otra. A(1) es la celdilla de arriba a la izquierda y A(E*E) es la posición de abajo a la derecha.

Entre 330 y 530 se ejecuta un gran bucle del tipo FOR-NEXT con el contador F. I cuenta nuevamente las líneas y C las columnas.

Las cinco reglas B a F son ejecutadas en las líneas de instrucciones siguientes: regla B: 415 a 440, regla C: 450, regla D: 490, regla E: 510 y 515, regla F: 460 y 470.

Entre las líneas 600 y 680 se expresa el cuadrado en forma numérica.



DIM y matriz

Un cuadrado de cifras es el típico ejemplo de un ordenamiento de datos conocido bajo el nombre de «matriz». Es bidimensional. Los grandes ordenadores BASIC están preparados para manejar matrices. Para ello habrá que dimensionar la matriz con una orden DIM. Si la matriz se llama A, habrá que escribir al inicio del programa

```
010 DIM A(5,8)
```

Con ello dispondrá de una matriz para 5*8 o, lo que es lo mismo, para 40 números distintos. Si en M tenemos las líneas y en N las columnas, y sabemos que M = 5 y N = 8, también podrá decirse

```
010 DIM A(M,N)
```

Si se desea el elemento de la fila tercera, columna quinta, escribiremos A(3,5).

Comprobación de Adam Riese

El programa, que trabaja nuevamente con más lógica que cálculo, deberá ser comprobado minuciosamente. Para ello utilizaremos el cuadrado de lado 5. En nuestro programa tiene el aspecto siguiente:

11	24	7	20	3
4	12	25	8	16
17	5	13	21	9
10	18	1	14	22
23	6	19	2	15

Si su programa produce este cuadrado para longitud de lado 5 es que funciona a la perfección.

El programa

```
 200 REM «CUADRADOS MAGICOS
 210 PRINT »LONGITUD DE LADO
     SOLO IMPAR Y MENOR DE 10«
 215 INPUT »LONGITUD DE LADO «;E
 220 IF E < 10 THEN 235
 225 PRINT »LADOS DEMASIADO
     LARGOS»; RUN
 230 STOP
 235 I = E - (INT (E/2))*2
 240 IF I < > 0 THEN 300
 245 PRINT »LONGITUD ES PAR« ;RUN
 300 FOR I = 1 TO E
 305 FOR C = 1 TO E
*310 G = ((I - 1)*E) + C + 10
 315 A(G) = 0
 320 NEXT C
 325 NEXT I
 330 I = (E/2) + 2
 331 C = INT(E/2) + 1
 350 FOR F = 1 TO (E*E)
*355 G = ((I - 1)*E) + C + 10
 360 IF I > E THEN 450
 400 IF C > E THEN 490
 410 IFA(G) < > 0 THEN 510
 415 A(G) = F
 420 I = I + 1
 430 C = C + 1
 440 GOTO 530
 450 IF C < = E THEN LET I = 1:
     GOTO 355
 460 C = E
 470 I = 2
 480 GOTO 355
 490 IF I < = E THEN LET C = 1:
     GOTO 355
 500 GOTO 460
 510 I = I + 1
 515 C = C + 1
 520 GOTO 355
 530 NEXT F
 600 FOR I = 1 TO E
 610 FOR C = 1 TO E
*620 G = (I - 1)*E + C + 10
 630 PRINT I;«»;C;«»;A(G)
 640 NEXT C
 650 PRINT
 660 NEXT I
 670 PRINT »FINAL«
 680 STOP
```

Para almacenarlo en mini-ordenadores

Nuestro programa ha sido diseñado para un ordenador de bolsillo SHARP PC 1211. Este ordenador puede almacenar hasta 1424 pasos de BASIC. Lo que no se necesita para el programa queda disponible como memoria para almacenar números. Cada ocho pasos representa una celdilla de memoria para datos, de este espacio podemos utilizar tanto como el programa deje vacío. Ya que en el programa necesitamos también variables, hemos reservado los nombres A a K. Por ello no disponemos del espacio entre A(1) y A(10) en la memoria. Por lo tanto, el almacenamiento de nuestro cuadrado se realiza a partir de A(11). Por ello, en el cálculo de las direcciones se suma 10. Estas instrucciones tienen un asterisco en el listado del programa.

De todas formas, nuestro enano BASIC dispone así de unas 1000 celdillas más de almacenamiento. Ello es suficiente para un cuadrado de longitud 9 con 81 números (ya que una variable requiere ocho celdillas).

¿Qué otras cosas pueden hacerse?

El programa de impresión entre 600 y 680 presenta la gran ventaja de poder imprimir cuadrados de cualquier tamaño de E. No obstante hay que componerse luego el cuadrado a mano o pegándolo en un papel. Quizá fuera mejor imprimir no línea a línea, sino columna a columna. Programar esta forma de impresión no es nada difícil.

No obstante, pueden tratarse las líneas como filas y las filas como columnas, obteniendo un cuadrado mágico igual al original, aunque en imagen inversa. Según el tipo de ordenador o de impresora que se tenga puede programarse para que imprima los cuadrados completos hasta longitudes de lado de aproximadamente 15. Quien sea aficionado a escribir programas de impresión puede hacer que los números aparezcan en cajoncitos. Otra versión sería imprimir sobre papel tantas columnas como quepan y luego componerse el cuadrado completo pegando tiras de papel. Para presentar pequeños cuadrados completos en pantalla, hay que sustituir la línea 630 por PRINT A(G);» «;

Las líneas

```
655 H = (E*E*E + E)/2
660 PRINT »EL NUMERO MAGICO ES «;H
```

calculan el valor mágico del cuadrado.

Sobre la vida y la muerte

Aquí nos encontramos con un problema de biología: la vida y la muerte de una especie animal; su evolución. Si por ejemplo viven N animales durante un determinado período y su índice de natalidad es del 25%, en cada período vendrán al mundo

$$G = 0{,}25 \star N$$

animales. La muerte es algo más complicada, pues los animales se dificultan mutuamente la vida al quitarse mutuamente el alimento de limitada presencia, se contagian enfermedades o incluso se comen entre sí. Por todo ello, el porcentaje de animales que mueren (S) resultará de la cantidad de animales vivos y de un factor de inhibición B:

$$S = 0{,}1 + B \star N$$

La cantidad de animales muertos en el período será de

$$T = S \star N = (0{,}1 + B \star N) \star N$$

Al final del año, la cantidad de animales es de

$$M = N \star G - T$$

Lo que se entienda bajo «período» dependerá de la especie animal. Podría tratarse de años, y en el caso de bacterias, de segundos.

El modelo animal

Nuestro modelo tiene en cuenta la cantidad de animales al inicio del primer período, el factor de inhibición B y la cantidad de períodos sobre los cuales debe realizarse el cálculo. El programa calcula con ello la cantidad M de animales al final del período y entra con M, tras imprimir el número de período y cantidad de animales, en el próximo período. Si N no se altera en dos períodos consecutivos, la población de la especie animal permanecerá constante. Entonces el programa se para por sí solo. Si la cifra de animales llega a valores inferiores al cero, el programa imprime el mensaje «especie extinguida». El programa no es complicado, pero no es fácil dominar el manejo de B. Cuando aún no existían ordenadores, la aplicación de estas fórmulas era muy complicada ya que todo el proceso de cálculo debía realizarse a mano. Con la aparición del ordenador, el experimento numérico se ha convertido en proceso usual incluso en las ciencias naturales.

El programa «Especie Animal»

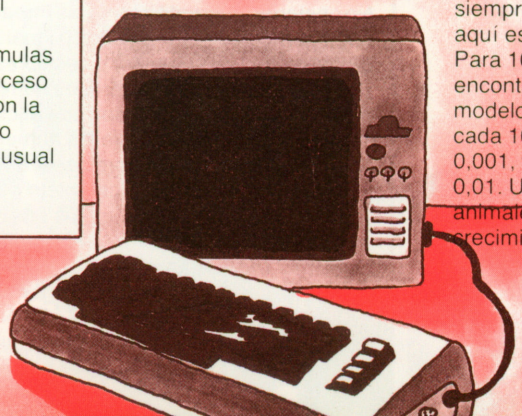
El programa apenas tiene algo complicado. Documenta cuidadosamente las magnitudes de N y B. Sin estos valores no puede predecirse el resultado de un cálculo. Y cuando deben realizarse muchos cálculos, estos valores son aún más importantes.

Para el comportamiento del programa (y por lo tanto del modelo de evolución de una especie animal que es simulado), los valores B y N son importantes. El porcentaje de fallecimientos se constata según la fórmula

$$S = 0{,}1 + B \star N$$

Cuando la especie animal deba mantenerse, S deberá ser en todo caso menor que 1, si no morirán más animales de los que hay. Para ello, $B \star N$ debe ser menor de 0,9; en el fondo considerablemente menor, para que no se mueran más animales que los que llegan al mundo. Los que nacen son $0{,}25 \star N$. Si $B \star N = 0{,}15$, morirán tantos animales como los que llegan al mundo. Este hecho sólo puede tenerse bajo control en el ordenador siempre que B y N no se alteren. Y aquí está el problema.

Para 1000 animales, B deberá encontrarse alrededor de 0,0001, si el modelo debe tener algún sentido. Por cada 100 animales, B debe valer 0,001, y para cada 10 animales será 0,01. Una B pequeña ante muchos animales significa un rápido crecimiento.



La especie estable

Una serie de números, tal como la calcula el programa, puede comportarse de forma muy distinta. En nuestro caso, los números no pueden ser ni menores ni mayores. En ese caso, la especie se expande, pero el índice de fallecimientos aumenta. Las cifras pueden también bajar. Entonces, el fallecimiento desciende también. En tercer lugar se da el caso de que la serie numérica se vuelva estable y oscile hacia un valor estable y fijo.

Nuestro modelo tiende a la estabilización. Por ello el programa se para al aparecer el mismo número de animales dos veces seguidas. También se acaba cuando ya no quedan más animales.

El programa

```
010 REM »EVOLUCION DE
    ESPECIE«
020 INPUT »CANTIDAD DE
    ANIMALES»;N
025 PRINT »LA CANTIDAD DE
    ANIMALES ES DE»;N
030 INPUT »INDICE DE
      INHIBICION«;B
035 INPUT »EL INDICE DE
    INHIBICION ES DE«;B
040 INPUT »CANTIDAD DE
    PERIODOS«;P
045 PRINT »LA CANTIDAD DE
    PERIODOS ES DE«;P
095 M = N
100 FOR K = 1 TO P
110 N = M
120 G = .25*N
130 S = .1 + B*N
140 T = S*N
150 M = INT (N + G - T)
160 IF M > 0 THEN 170
165 PRINT »ESPECIE ANIMAL
    EXTINGUIDA TRAS»;K;
    «PERIODOS«
168 GOTO 200
170 PRINT K;» «;M
180 IF N < > M THEN 190
185 PRINT »ESPECIE ANIMAL
    ESTABLE AL CABO DE »;K;
    «PERIODOS«
188 K = P
190 NEXT K
200 PRINT »FIN«
210 STOP
```

Datos de comprobación sobre la evolución de una especie

Para comprobar el programa lo mejor es darle un ejemplo para que lo calcule, una vez lo haya introducido cuidadosamente en el ordenador y haya verificado que no tiene erratas. Aquí le damos unos cuantos valores de prueba que llevan rápidamente a la estabilidad:

Cantidad de períodos 25

1.	131
2.	132
3.	133
4.	134
5.	135
6.	136
7.	136

Especie animal estable al cabo de 7 períodos.

¿Qué otras cosas pueden hacerse con este modelo?

Con nuestro modelo se puede variar, naturalmente, el índice de nacimientos y de muertes. En nuestro ejemplo es de 0,25. No cuesta nada «abrir» el programa allí donde haga falta el índice de natalidad para introducirlo por el teclado, pudiendo ver cómo reacciona el modelo con datos distintos. Para el aficionado a los ordenadores resulta especialmente atractiva representar los datos calculados para las alteraciones en el número de animales de forma gráfica. Sin embargo, para ello necesitaremos algo más que un ordenador de bolsillo. Aquí tenemos un ejemplo: diez animales con una inhibición de 0,00105 crecen en unos 45 pasos hasta 136 y se estabilizan allí. Pero pueden ser también 10 millones de bacterias que crecen en millones de pasos y que en 45 horas se estabilizan como cultivo estable de 136 millones.

Advertencia

El planteamiento del problema en estas dos páginas —así como el de los zorros y conejos— lo hemos extraído de un libro de texto con fines escolares que mencionamos gustosamente:

Ocker-Schöttle-Simon
Informatik
Oldenburg Verlag München
Wien. 1979

Cinco veces infinito

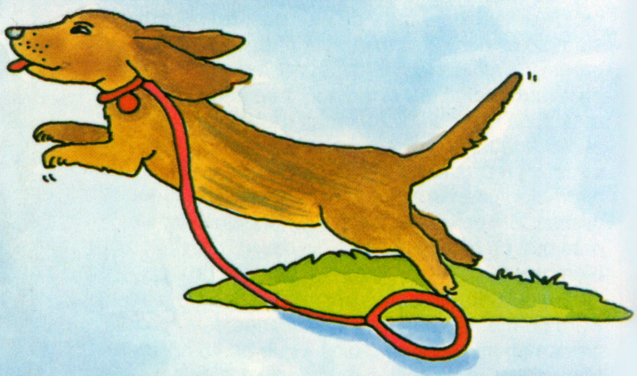
Bajo el concepto de serie infinita el matemático entiende una suma compuesta de infinitos sumandos. He aquí un ejemplo:
1 + 1/2 + 1/4 + 1/8 + 1/16...
¿Quién tiene tiempo suficiente para realizar esta suma aparentemente sin fin? Si hay alguien, será sin duda un ordenador. No tiene nada de complicado. La serie viene caracterizada por el hecho de que cada elemento se obtiene multiplicando su predecesor por 1/2. La segunda característica importante de esta serie es que sus elementos van haciéndose cada vez más pequeños. Podría ser que la aportación de los elementos siguientes sea tan pequeña que ya no juegue ningún papel digno de mención.

Este tipo de series se llaman convergentes. Las series divergentes están compuestas por elementos, que aunque también se vayan haciendo más pequeños, ejercen su influencia en el resultado total. En series divergentes pueden realizarse todas las sumas que se quiera sin obtener un resultado más o menos estable.

Diferentes series
En las series infinitas existen innumerables versiones. Con series divergentes podemos morirnos sumando. Otras series, no obstante, son rápidamente convergentes y no hace falta calcular mucho. Otras son muy lentas, pues requieren miles y miles de elementos para obtener un resultado.
La teoría de las series es difícil aunque muy interesante. Aquí tratamos cinco series programadas. Todas ellas proporcionan gran cantidad de cálculos al ordenador, pero todas pueden ejecutarse en una computadora BASIC de bolsillo.

Así se llaman y ejecutan los cinco programas
GEO 1 (a partir de RUN 5) — Serie geométrica en la que hay que introducir el número de elementos N.
GEO 2 (a partir de RUN 105) — Serie geométrica en la que hay que introducir una barrera de exactitud.
GEO 3 (a partir de RUN 205) — Serie armónica divergente; con la cantidad de elementos a sumar.
GEO 4 (a partir de RUN 305) — Serie armónica alternante.
GEO 5 (a partir de RUN 405) — Cálculo del número Pi.

La serie geométrica
La Serie S1 = 1 + 1/2 + 1/4 + 1/8 + 1/16... presenta una convergencia relativamente buena. Se llama «geométrica» por desempeñar un papel en el campo de la geometría.
El programa GEO 1 calcula la suma de tantos elementos como se introduzca en N. Un bucle FOR-NEXT trabajará de 1 a N. El valor límite de esta serie es 2. Pertenece al problema del perro del guardabosques: mientras su amo camina de regreso a casa, el perro corre al doble de velocidad hasta la casa y de vuelta hasta su amo.
El programa GEO 2 trata la misma serie. Pero ahora no se introduce cuántos elementos ha de tener la suma, sino que se indica una barrera E. Si la aportación de los sumandos es menor de E, el programa se para:



La serie armónica

El programa GEO 3 trata una serie que por su composición se denomina «armónica»:
S2 = 1 + 1/2 + 1/3 + 1/4 + 1/5...
Puede ejecutarse con RUN 205 para N miembros. Esta serie es un mal bicho, pues es divergente: cuantos más sumandos tenga, más cantidad resulta. Todos los programas imprimen gran cantidad de datos; si no le gusta convierta la línea 235 en 242. La cifra 235 borrará la orden de impresión y entonces escriba:

242 PRINT K; » «;S

De esta forma sólo se imprimirá el resultado final tras N vueltas.

Calculamos el número Pi

Cuando se conoce una serie infinita, es decir, su valor límite aproximado, puede calcularse éste con ayuda de la serie hasta la exactitud deseada. Esta es precisamente la finalidad de las series. No obstante, con los ordenadores normales y su BASIC, los números no pueden tener más de 8 ó 16 cifras. Esta es la serie que nos lleva al número Pi:
PI = 4*(1−1/3+1/5−1/7+1/9 −1/11...).
Esta serie, aunque de forma muy débil, es convergente.

El paquete de programas

```
005 PRINT »RUN GEO 1«
010 REM »GEO 1«
015 INPUT »SUMANDOS ?«;N
020 S = 1
025 G = 1
030 FOR K = 2 TO N
035 PRINT K; » «;S
040 G = G*0,5
045 S = S + G
050 NEXT K
055 PRINT »FINAL«
060 STOP

105 PRINT »RUN GEO 2«
110 REM »GEO 2«
115 INPUT »BARRERA «;E
120 PRINT »LA BARRERA ES «;E
125 S = 1
128 N = 1
130 G = 1
135 G = G*0,5
140 S = S + G
145 IF G < = E GOTO 165
150 N = N + 1
155 LPRINT N;» «;S
160 GOTO 135
165 PRINT »FINAL«
170 STOP

205 PRINT »RUN DIVERGENTE«
210 REM »DIVERGENTE«
215 INPUT »NUMERO SUMANDOS «;N
220 S = 0
225 FOR K = 1 TO N
230 S = S + 1/K
235 PRINT K;» «;S
240 NEXT K
245 PRINT »FINAL«
250 STOP

305 PRINT »RUN CONVERGENTE«
310 REM »CONVERGENTE«
315 S = 0
325 M = −1
330 F = −1
335 FOR K = 1 TO N
340 M = M*F
345 S = S + (M/K)
350 PRINT K; » «;S
355 NEXT K
360 PRINT »FINAL«
365 STOP

405 PRINT »RUN PI«
410 REM »PI«
415 INPUT »CANTIDAD ELEMENTOS «;N
420 I = 0
425 S = 0
430 M = −1
435 F = −1
440 FOR K = 1 TO (2*N) STEP 2
445 I = I + 1
450 IF I < = 1000 GOTO 465
455 PRINT ((K − 1)/2);» «;P
460 I = 1
465 F = F*M
470 G = F*(1/K)
475 S = S + G
480 P = 4*S
485 NEXT K
490 PRINT »FINAL«
495 STOP
```

Los datos de comprobación

Estos datos de comprobación le dirán si su programa, tal como lo ha introducido, funciona correctamente. Ya que los programas son de gran intensidad de cálculo, hay que estar seguros de que realizan los cálculos también de forma correcta.

Armónico con signo cambiante

A partir de RUN 305 llega al programa GEO 4. La serie
S3 = 1 − 1/2 + 1/3 − 1/4 + 1/5 − 1/6... se denomina, dentro del campo especializado de las matemáticas, «armónico alternante». Alternante significa que el signo cambia. Aunque los elementos de esta serie se reducen tan ligeramente que la serie resulta divergente, en las series alternativas ocurre que todo aquello que se suma en demasía, es restado por el signo cambiante. También aquí vale la pena alargar la serie. Pues quien tiene un ordenador, dispone también de todo el tiempo de cálculo que quiera.
En este programa, el problema se encuentra en que cada vez hay que cambiar el signo. Para ello se ponen dos variables M y F a − 1. En cada pasada se multiplica M por F. La consecuencia de esta multiplicación es que M alterna en el transcurso del programa, y la línea 345 se encarga de que cambie su signo.

Estos son los primeros diez valores del programa GEO 1 (a partir de RUN 5):

```
RUN GEO 1
 2        1
 3        1,5
 4        1,75
 5        1,875
 6        1,9375
 7        1,96875
 8        1,984375
 9        1,992188
10        1,996094
```

Si ponemos la barrera para el programa GEO 2 (a partir de RUN 1050) en E = 0,00001, se obtiene el siguiente resultado:

```
RUN GEO 2
La barrera es 0,00001
 2        1,5
 3        1,75
 4        1,875
 5        1,9375
 6        1,96875
 7        1,984375
 8        1,992188
 9        1,996094
10        1,998047
11        1,999024
12        1,999512
13        1,999756
14        1,999878
15        1,999939
16        1,9997
17        1,999985
```

Aquí están los primeros diez valores para el programa de la serie divergente armónica (a partir de RUN 205):

```
RUN DIVERGENTE
 1        1
 2        1,5
 3        1,833333
 4        2,083334
 5        2,283334
 6        2,45
 7        2,592857
 8        2,717857
 9        2,828969
10        2,928969
```

Los valores siguientes proceden de la serie armónica convergente (a partir de RUN 305):

```
RUN CONVERGENTE
 1        1
 2        0,5
 3        0,8333334
 4        0,5833334
 5        0,7833334
 6        0,6166667
 7        0,7595238
 8        0,6345238
 9        0,745635
10        0,6456349
```

El programa a partir de RUN 405 tiene una convergencia débil y suministra para cada 1000 pasos los valores siguientes:

```
RUN PI
1000      3,140579
2000      3,14106
```

Con tantas posiciones no puede garantizarse que cada computador llegue en su última cifra al mismo número. Ello dependerá del grado interno de precisión.

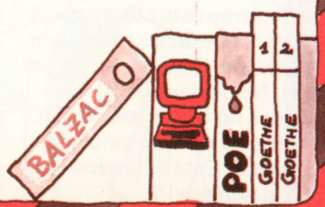
El ordenador poeta

La pregunta de si un ordenador puede escribir poesía no puede responderse a menos que se sepa cuál es el mecanismo de la poesía. El ordenador puede componer un texto; puede balbucear sin ton ni son. Pero resultará más divertido si dejamos que el ordenador componga sus «versos» de forma casual, sin que podamos predecir lo que vendrá después. Ya en los años cincuenta se hizo que un ordenador produjera versos al azar. La receta es sencilla: se le da al ordenador una muestra de frase gramaticalmente correcta y se le deja que, mediante un generador de números aleatorios, escoja las palabras de un almacén determinado. Aquí tenemos un programa de este tipo.

Los poemas que hace suenan en parte muy modernos. Veamos un ejemplo:

EL CIELO ES FIEL Y
TODA TIERRA ES BUENA Y
MI NIÑO ES ABIERTO PERO
CADA LUNA ES DULCE Y
EL JARDIN ES BELLO.

Excepto «es», todas las palabras han sido buscadas al azar. Sin embargo podemos observar como los artículos tienen el género correcto, al igual que TODA y BUENA, que son femenino por tratarse de la TIERRA.



La construcción de las frases

La construcción de las frases puede verse en el ejemplo
TODA TIERRA ES BUENA.
En el programa disponemos de diez sustantivos como HOMBRE, MUJER, ARBOL o TIERRA, al igual que diez adjetivos, algunos sin género como FIEL, DULCE y otros con género como ABIERTO, ABIERTA. Todas estas palabras se buscan aleatoriamente y, entre ellas, imprime el verbo «ES». TIERRA es el sujeto de la frase y ES BUENA el predicado. Delante del sustantivo puede imprimirse, con la concordancia correcta del género, una de las siguientes palabras:

1 EL, LA
2 CADA, MI
3 TODO, TODA
4 NUESTRO, NUESTRA.

Así está configurado el programa

El programa necesita un número H entre 0 y 0,3 para el generador de números aleatorios (ver página 22/23). También quiere saber cuántas líneas debe tener la «poesía». Todo ello se establece en la primera parte del programa.
Para almacenar nuestro diccionario utilizamos un ámbito numérico, de nombre A$, ya que debe almacenar datos alfanuméricos. Comienza con A$(20), ya que necesitamos cubrir otras variables. (A$(1) es A$, A$(2) es B$ y A$(19) es T o T$, según se requiera). Este ámbito numérico finaliza en A$(59) y contiene todo el vocabulario de nuestro poeta. Entre las líneas 100 y 139 se introduce en el ordenador un determinado número (40) de palabras. Entonces viene un gran bucle de 1 a P, que abarca las P líneas. En este bucle se fabrican cinco números aleatorios que son convertidos en direcciones entre 20 y 59. A las líneas con los sustantivos como HOMBRE precede una dirección con el género especificado en el artículo determinado EL. Si debe imprimirse otra palabra en lugar de este artículo, se comprueba con éste primero el género y luego se imprime la frase.
Finalmente encontramos un generador de números aleatorios. Este generador puede ser eliminado si su ordenador posee la función RND. Escribiremos pues:

```
080 J = RND
```

con lo que obtendremos un número J aleatorio, mayor que cero y menor que uno y que tendrá tras el punto decimal tantas cifras como sea usual en su BASIC. Si al comienzo del programa escribimos

```
015 RANDOMIZE
```

haremos algo similar que con nuestro número H para el generador. Esta es una diferencia dialectal del BASIC y habrá que consultar el manual.

El programa

```
010 REM «VERSOS
020 INPUT »GENERADOR ALEATORIO
    «;H
030 PRINT »GENERADOR
    ALEATORIO = »;H
040 Q = H + 0,314159625
050 INPUT »CUANTAS LINEAS? »;P
060 PRINT P;»LINEAS «
100 A$(20) = »EL«
101 A$(21) = »HOMBRE«
102 A$(22) = »LA«
103 A$(23) = »MUJER«
104 A$(24) = »EL«
105 A$(25) = »NIÑO«
106 A$(26) = »LA«
107 A$(27) = »NIÑA«
108 A$(28) = »EL«
109 A$(29) = «CIELO«
110 A$(30) = »LA«
111 A$(31) = »CASA«
112 A$(32) = »EL«
113 A$(33) = »JARDIN«
114 A$(34) = »LA«
115 A$(35) = »TIERRA«
116 A$(36) = »EL«
117 A$(37) = »ARBOL«
118 A$(38) = »LA«
119 A$(39) = »LUNA«
120 A$(40) = »BUENO«
121 A$(41) = »BUENA«
122 A$(42) = »BELLO«
123 A$(43) = »BELLA«
124 A$(44) = »ABIERTO«
125 A$(45) = »ABIERTA«
126 A$(46) = »FIEL«
127 A$(47) = »DULCE«
128 A$(48) = »FUERTE«
129 A$(49) = »HERMOSA«
130 A$(50) = »CADA«
131 A$(51) = »MI«
132 A$(52) = »TODO«
133 A$(53) = »TODA«
134 A$(54) = »NUESTRO«
135 A$(55) = »NUESTRA«
136 A$(56) = »Y«
137 A$(57) = »PERO«
138 A$(58) = »AUNQUE«
139 A$(59) = ».«
400 FOR K = 1 TO P
410 GOSUB 900
415 IF J > 9 THEN 410
420 B = J*2 + 20
430 GOSUB 900
435 IF J > 4 THEN 430
440 C = J*2 + 40
445 IF A$(B) = »EL« THEN 455
450 C = C + 1
455 GOSUB 900
460 IF J > 1 THEN LET D = B; GOTO 500
465 D = J + 50
470 GOTO 550
500 GOSUB 900
510 IF J > 1 THEN 550
520 E = J*2 + 52
530 IF A$(B) = »EL« THEN 550
540 E = E + 1
545 D = E
546 GOTO 600
550 GOSUB 900
555 IF J > 3 THEN 550
560 F = J + 56
600 B = B + 1
605 IF K = P THEN LET F = 59
610 PRINT A$(D) + A$(B) + »ES« +
    A$(C) + A$(F)
620 NEXT K
630 STOP
900 IF Q > = 0,4 THEN 902
901 Q = Q + 0,35
902 Q = Q*Q
903 R = INT(Q*100000)
904 S = INT(Q*10000)*10
905 J = R - S
906 RETURN
```

La gramática del ordenador

La gramática que utiliza el ordenador es extremadamente sencilla. Quien quiera programar más gramática —y falta que le hace a nuestro modelo— se planteará un problema que gustosamente dejamos a merced suya. Si, por ejemplo, queremos convertir alguna de las frases en pregunta (con o sin el POR QUE), habrá que poner la partícula verbal «ES» al comienzo de la frase:

CADA JARDIN ES ABIERTO PERO
(POR QUE) ES LA LUNA FIEL?

Aquí se dispone de un campo inmenso de trabajo para hacer hablar al ordenador, el cual a su vez deberá disponer también de mayor capacidad.

Comentarios sobre el programa poético

El programa tiene una estructura relativamente sencilla. Tan sólo el cálculo de las direcciones en el ámbito alfanumérico A$ puede resultar algo oscuro. Aquí habrá que estudiar el montaje.

El programa se divide en cuatro grandes partes:

1) En la parte introductoria, entre 00 y 060 se gradua el generador de números aleatorios (H y Q). Se pregunta también por el número de líneas a producir (P).
2) Entre 100 y 139 se carga el diccionario en el ámbito alfanumércio A$.
3) Entre 400 y 630 se convierten los números aleatorios del generador en direcciones para las palabras a utilizar en el diccionario A$. Entonces, estas palabras se imprimen en el orden y con la estructura marcados en la línea 610.
4) Entre 900 y 906 se encuentra el generador de números aleatorios como subrutina a la que se accede mediante GOSUB.

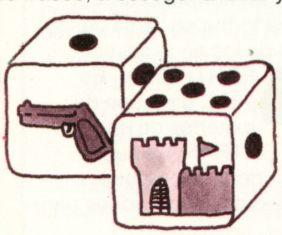
Comprobando el programa

La comprobación con números aleatorios no es fácil sobre todo por el hecho de que no siempre necesitamos números del mismo tamaño. Los sustantivos de nuestros poemas requieren un número entre 0 y 9, a veces necesitamos una cifra sólo entre 0 y 4, pero este número debe ser de una sola cifra.

El programa poético está montado de tal forma que con GOSUB 900 se exije un número aleatorio. Con IF se comprueba si es aceptable dentro del margen. Si no lo es se pide otro número, es decir, se vuelve al GOSUB. Lo mejor es que imprima, con un pequeño programa de print, los primeros 30 números aleatorios, poniendo luego el generador en posición inicial y comprobando lo que ocurre con estas cifras, es decir, si el programa sabe buscar las direcciones correctas en el diccionario para no hacer faltas gramaticales en la concordancia del género.

Así mejoraremos nuestro programa

Ya hemos hablado sobre la ampliación de la gramática anteriormente. Si su ordenador es lo suficientemente grande, puede programar también otro tipo de frases, a escoger al azar y rellenándolas correctamente con palabras del diccionario interno. Esto puede resultar muy divertido. También puede ser interesante elegir palabras que sean típicas de un poeta. Se asombrará del grado en que este tipo de palabras influyen en el carácter de las poesías casuales. Habrá una gran diferencia entre buscar palabras de una novela negra y buscarlas en un cuento de hadas.

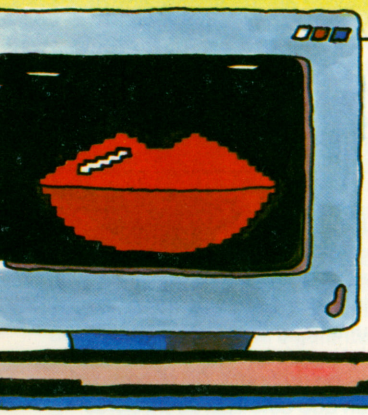


Si su ordenador puede con ello, amplíe todo lo posible su vocabulario. Si su ordenador dispone de los recursos de RND y/o DIM, no dude en utilizarlos. Con DIM puede que sea incluso necesario. Ello sin embargo alterará también la parte del programa dentro del bucle FOR-NEXT.

Así se manejan los ordenadores

Si de la docenita de programas que contiene este libro ha puesto en marcha alguno o incluso lo ha mejorado o ampliado, ya habrá enriquecido algo sus conocimientos. Uno de estos conocimientos podría ser que no hay nada mejor que un programa escrito por uno mismo. En esos casos se sabe exactamente lo que se hace, por qué y cómo se hace, siendo conscientes también de a quién hay que tirarle de los pelos en caso de que haya problemas. Cuando se escriben programas largos y se ejecutan, se llega casi siempre a un punto en el que se piensa que lo mejor sería tirarlo todo a la basura y empezar de nuevo. Pero ¿quién nos garantizará que a la segunda irá la vencida?

Además habremos perdido ya demasiado tiempo metiendo bastantes instrucciones, por lo que es mejor abrirse camino, aunque sea a mordiscos. Por este motivo, casi todo programa tiene sus pequeñas cicatrices, en las que el profesional nota que aquí se han realizado malabarismos. A veces será una curiosa forma de numerar las líneas que rompe el esquema, a veces hay variables con nombres muy curiosos, etc. En nuestros programas hemos evitado, con toda la intención, proceder a «pulirlos» y a darlos los últimos toques.



El BASIC y sus dialectos

A mediados de los años cincuenta se inventó el primer lenguaje de programación, orientado más hacia el problema que hacia el ordenador. Este lenguaje se llama FORTRAN y aún existe. No obstante, este lenguaje ha sido varias veces mejorado, ampliado y, finalmente, normalizado. En esta nueva forma se llama desde mediados de los años sesenta FORTRAN IV. En 1965, un grupo de expertos de una universidad americana creó, partiendo del FORTRAN, un nuevo lenguaje. Recibió el nombre de BASIC (Beginners All purpose Symbolic Instruction Code = Código simbólico de instrucciones para todo uso destinado a principiantes).

En su configuración básica, el BASIC está compuesto de dos docenas de órdenes, procedentes casi todas del FORTRAN. Es un lenguaje comprendido también por ordenadores relativamente pequeños. Entre tanto se ha producido un crecimiento salvaje de ampliaciones que se diferencian a menudo sólo por pequeñeces. Quien quiera introducirse un poco más a fondo en el mundo de los ordenadores, sería conveniente que junto al BASIC aprendiera algún otro lenguaje de programación, para no quedarse trabado en las desventajas y debilidades del BASIC.

En busca de diferencias de BASIC

Nuestros programas están escritos en un BASIC elemental y primario. Pero algún que otro dialecto requiere punto y coma donde otros esperan dos puntos. Una vez haya introducido su programa y el procesador no cumpla alguna instrucción, éste enviará un mensaje de error, anotando el número de la línea que no le guste. Algunos lo muestran en pantalla, otros lo imprimen en papel.

En estos casos se comprueba la línea del programa, signo a signo, en busca de fallos mecanográficos. Si no hay, habrá que consultar el propio manual de BASIC para descubrir dónde puede estar el error. Si encontramos una desviación dialectal se comprobarán también todas las demás instrucciones similares del programa, para ver si allí también aparece tal desviación.

IF

Hay casos en los que el IF tiene una estructura diferente a la nuestra. Nuestro IF presenta la fórmula siguiente:

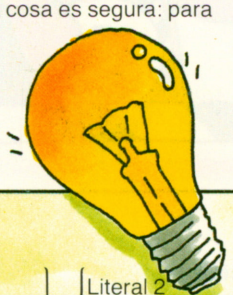
1) IF expresión 1: comparación lógica, expresión 2: orden. Es decir, algo así como:

IF B + 3 = C GOTO 50 ó
IF B + 3 = C THEN 50 ó
IF A = B LET B = B + 1
(aquí LET es obligatorio)

Comprobación de programas

Si su ordenador dispone de ayudas de comprobación, como DEBUG o TRACE, debería trabajar con ellas. «Ayuda de comprobación» significa principalmente que se pasa a través del programa instrucción a instrucción viendo lo que sale después de cada una de ellas, comparando los resultados con los datos de comprobación. Si el programa tiene INPUT, debe introducir los mismos valores que nosotros. Esto se llama DEBUG, es una palabra americana y significa «despiojar». El ordenador tiene una tecla, con la que puede verse el programa línea a línea. Ahora sirve para ejecutar las instrucciones a mano. Una vez pulsada aparece un número de línea; podemos, por ejemplo, observar el contenido de las variables. (Si pulsamos DEBUG 100, el funcionamiento paso a paso comienza en la línea 100).

TRACE significa «rastro». Cuando un computador dispone de este sistema imprime tras cada instrucción lo que se ha alterado —como por ejemplo el contenido de una variable. TRACE puede utilizarse también insertando en algunos lugares estratégicos sentencias PRINT, para ver los valores más importantes. En cuanto el programa funciona se eliminan estas líneas de él. Una cosa es segura: para comprobar hace falta tener la cabeza despejada.



2)

IF { Literal 1 / Variable de texto 1 } = { Literal 2 / Variable de texto 2 }

Algo así como:

IF A$ = »ABC«, ó

IF A$ = B$

Si a su ordenador no le gusta esta versión de IF, tendrá que alterar, bajo ciertas condiciones, el IF y su entorno.

LET

Algunos procesadores exijen que delante de una asignación (=) se encuentre la palabra LET. En nuestro caso podemos eliminarla si queremos ahorrar espacio de memoria, pero no es imprescindible. Pero i la asignación está dentro de una sentencia IF, debe constar LET.

DIM

Nuestro BASIC no conoce la instrucción DIM. Con él podemos indexar la variable A sin utilizar DIM. El tamaño del índice dependerá sólo del espacio libre en la memoria. La mayoría de BASICs exijen DIM en cuanto una variable indexada tenga más de diez valores.

El programa es demasiado grande

Si su programa no le cabe en la memoria el asunto es peliagudo. Los programas de este libro están preparados para memorias del tipo siguiente:

26 celdas básicas almacenan valores para los nombres de variable A – Z. En el programa se llaman A a Z o A$ a Z$ y/o A(1) a A(26), pero también A$(1) a A$(26). La última celdilla de memoria básica podrá tener, por lo tanto, uno de los cuatro nombres siguientes: Z, Z$, A(26) o A$(26).

El ordenador puede almacenar, además 1424 pasos de programación. Un paso de programación es una orden. En instrucciones complicadas hacen falta más pasos por sentencia. En lugar de sentencias pueden almacenarse también datos —una variable en lugar de ocho órdenes—. Estas memorias flexibles complementan la memoria básica con los nombres A(27) a A(204) o A$(27) a A$(204). (El número 204 es sólo teórico, pues en ese caso no quedaría sitio para ningún programa).

La disposición de este rellenado de la memoria la realiza el procesador por sí solo.

Si el programa es demasiado largo habrá que renunciar a la solución o descuartizar sin piedad el programa. Primero se tirarán a la papelera todos los REM. Entonces se sustituyen los comentarios y literales, en los que sea posible entenderse también con palabras código. Si todo eso aún no es suficiente, habrá que mirar si puede abreviarse un poco el programa. Si tampoco así funciona, podemos estudiar la posibilidad de dividir en partes el programa, de forma que los resultados de una parte puedan ser los valores iniciales de otra.

La posibilidad de intercambio entre programa y datos, antes mencionada, se soluciona en los PC generalmente con un procesador que se encarga de la disposición. Almacena el programa de delante hacia atrás y los datos de atrás hacia delante, o al revés. Para saber cómo estamos, en nuestro ordenador podemos preguntar en cualquier momento el estado de la memoria con la sentencia MEM («Memory» o «memoria»).

El ordenador nos dice en este caso:

384 STEPS 48 MEMORIES

Steps son pasos de sentencia, y memories es la memoria flexible. El desarrollo tiende a que en los ordenadores pequeños haya cada vez más capacidad de memoria. Muchos ordenadores pueden incluso ampliarse mediante módulos de expansión que compra e instala el mismo usuario.

El programa es demasiado lento

Es difícil acortar programas. Pero aún más difícil es acelerarlos. Si un programa es muy lento, no tendremos nunca la certeza de si el fallo está dentro del programa en un bucle del que no sale. Aquí, por ejemplo, tenemos un programa que funcionará hasta que alguien tenga la bondad de romperle el cuello:

```
010 REM »MARATON«
020 FOR K = 1 TO 10
030 K = K – 1
040 NEXT K
```

Cuando los programas correctos funcionan muy lentamente, es recomendable buscar las partes que lo frenan. Esto se hace con PRINTs esporádicos que nos dirán lo que el programa hace en ese momento y con un cronómetro para controlar la duración. Por ejemplo, en un bucle FOR-NEXT sólo debe realizarse aquello que sea imprescindible para cada vuelta.

Para estar seguros de que un programa lento no queda colgado en sus propios fallos podemos listar el lugar en el que se encuentra. Esto se hace con bucles FOR-NEXT, que imprimen valores intermedios importantes cada 100 ó 1000 pasos.

Una buena idea

Es posible que conozca a gente, sea en el trabajo, en la vecindad o entre sus familiares, a quien le guste trabajar con ordenadores. Gente a la que en lugar de una cajita de bombones se les pueda regalar también programas para ordenador. Sería naturalmente de ayuda que les regalara nuestro folleto. Pero nos parece casi mejor aún que se gastara los duros en un diskette o en una cinta, en la que estén almacenados nuestros programas como biblioteca. Deberá naturalmente documentar qué aspecto tienen los programas y qué hacen, una vez los haya comprobado en su ordenador. (Desde nuestro punto de vista se da por entendido que el librito que está leyendo en estos momentos, y otros de esta serie, pertenecerán naturalmente al paquete de regalo.)

Bibliotecas para programas

Con este pequeño libro hemos tenido también un poco la intención de convencerle de que no todo hay que programárselo uno mismo. Hay otras personas que tampoco lo hacen nada mal.

Los programas también se pueden comprar. A menudo son muy caros y profesionales. Es como si nos comprásemos un diccionario por tomos para consultar una sola palabra. Este librito le suministra doce palabras, y quizás el estímulo a otras más.

Quien posea muchos programas y el ordenador adecuado, se debe montar una biblioteca. Se trata de discos o cintas de cassette, en los que se encuentran programas correctos y comprobados, listos para su uso. Cómo debe hacerse la biblioteca dependerá de su ordenador. En todo caso sea nuestra recomendación la de incluir los programas que le gusten de este libro en su biblioteca.

Una biblioteca de programas comprobados ahorra la búsqueda de erratas y le suministra, siempre que lo desee, un programa apto para su uso en el ordenador.

La documentación

Una biblioteca de programas no debe contener ningún programa que no esté correctamente comprobado. Si esta condición no puede cumplirse, todo aquél que utilice la biblioteca deberá saberlo. A cada programa le pertenecen anotaciones en las que se dice lo que dicho programa hace y cómo puede comprobarse.

Este tipo de anotaciones se llama «documentación». Los programas no documentados no sirven para nada. Una buena documentación se compone de los siguientes datos:

+ Un listado correcto y actualizado de las instrucciones, tal como se obtiene con la orden LIST.
+ Un comentario del programa sobre sus diferentes apartados.
+ Instrucciones y datos para la comprobación, y
+ Datos y comentarios sobre lo que el programa hace o debe hacer.

Hemos intentado configurar este libreto de tal forma que represente la documentación útil para los programas que le da a conocer.

Índice alfabético

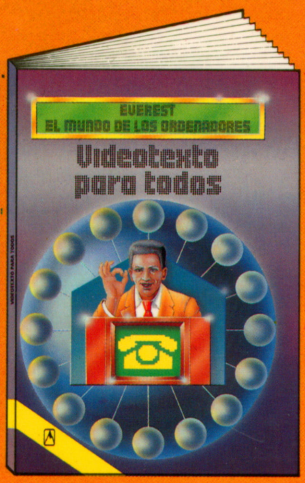
Índice alfabético

Indice

Indice

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.



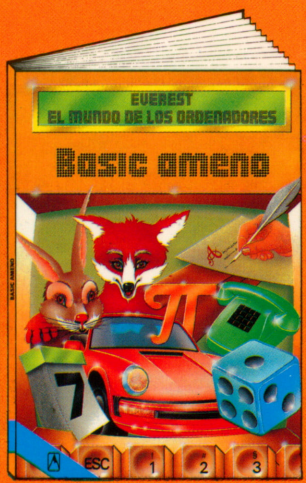
Videotexto para todos
Con la aparición del Videotexto (Vtx) se abre un inmenso campo en el mundo tecnológico. Este libro le proporciona toda la información sobre el Videotexto.

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.