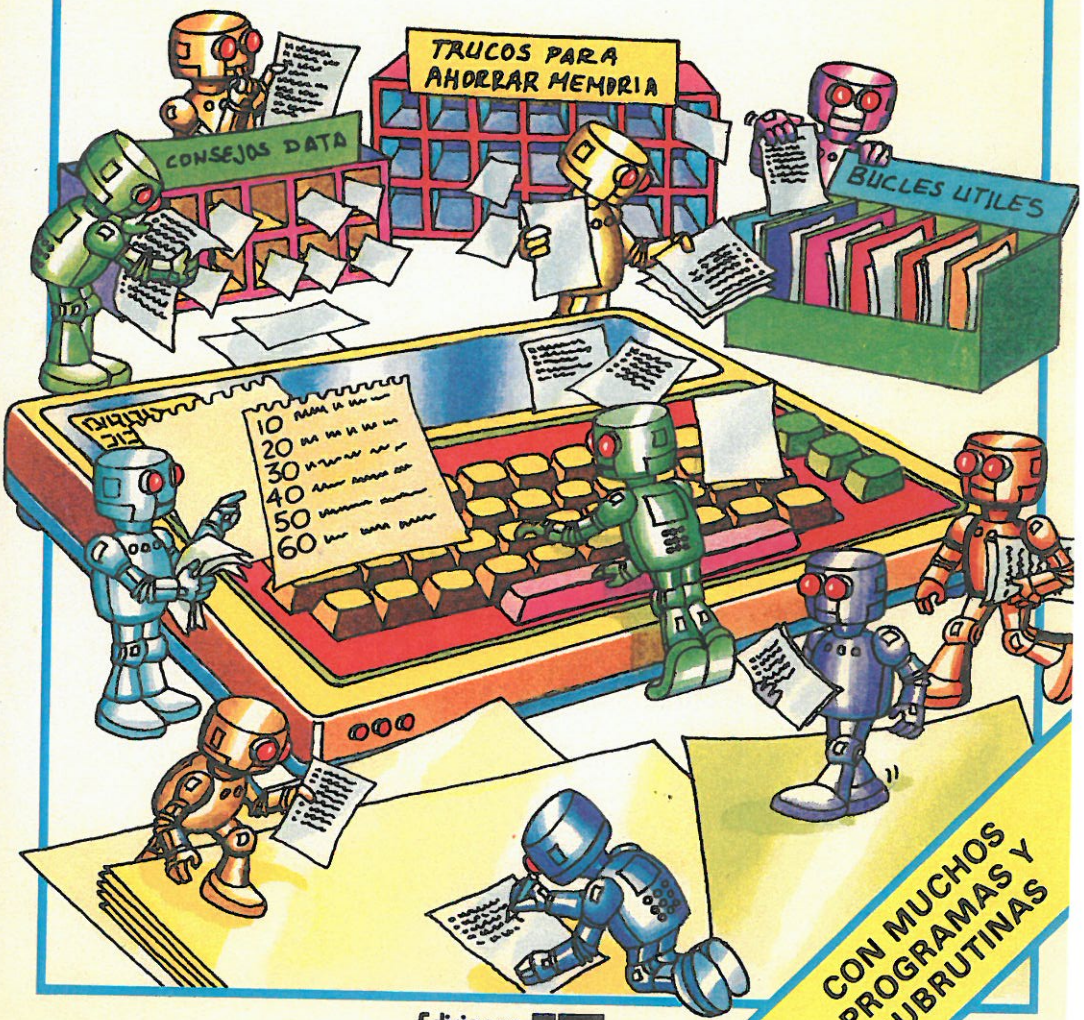
Colección Electrónica

TECNICAS Y TRUCOS DE PROGRAMACION

Métodos y consejos para una mejor programación

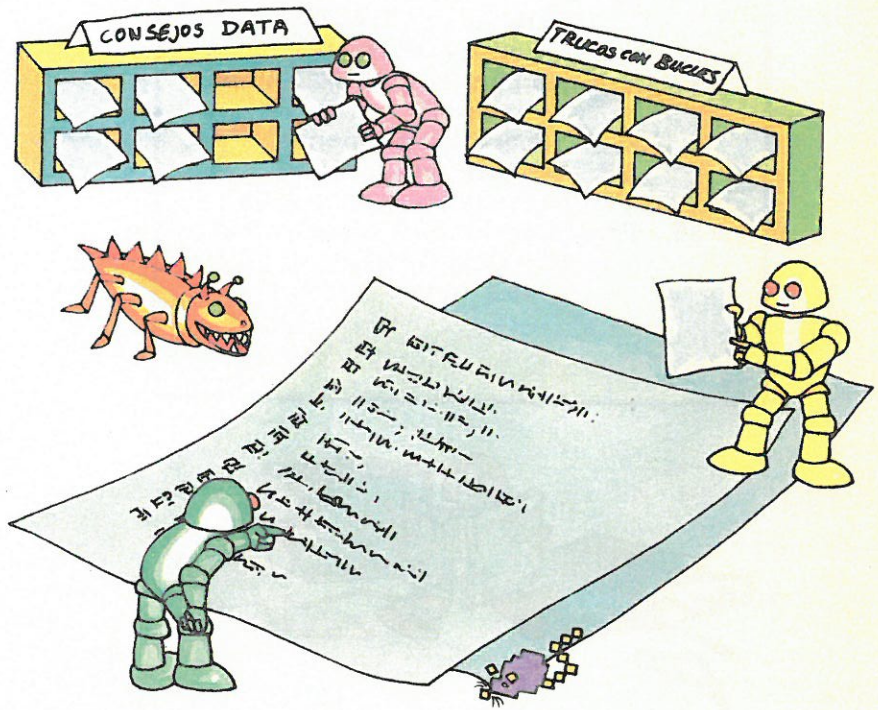


CON MUCHOS PROGRAMAS Y SUBRUTINAS

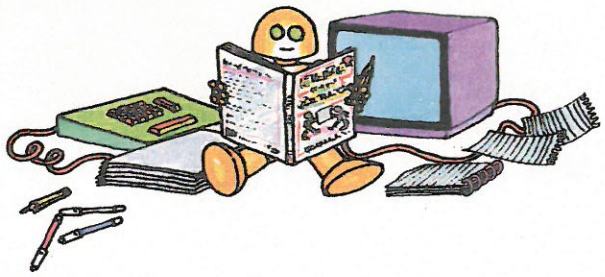
Ediciones Pleso sm

TECNICAS Y TRUCOS DE PROGRAMACION

Lisa Watts y Les Howarth



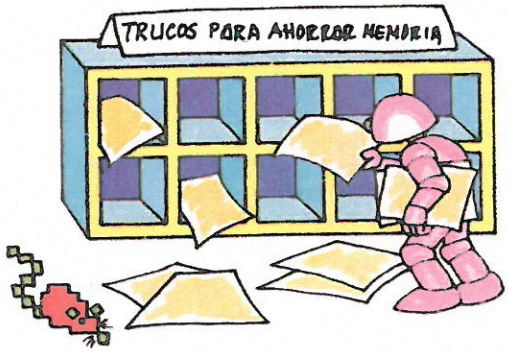
Contenido



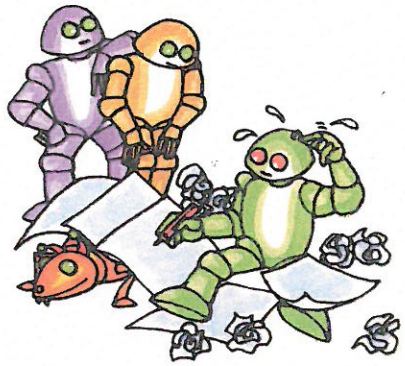
Impreso en España. Printed in Spain. MELSA - Pinto (Madrid)
I.S.B.N.: 84-7374-172-2 Depósito Legal: M-15970-1987

Sobre este libro

Este libro está repleto de trucos, técnicas y consejos que te ayudarán a escribir mejores programas en BASIC. Hay trucos para hacer que los programas se ejecuten a mayor velocidad, ideas para ahorrar espacio de memoria y formas de bloquear tus programas para que nadie pueda ejecutarlos.

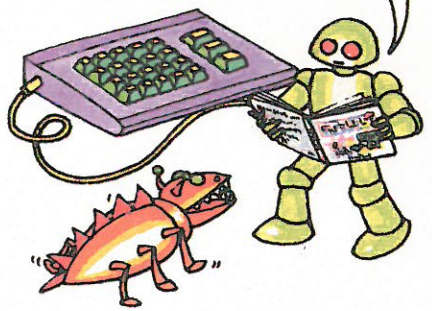


La primera parte del libro te enseña cómo planear y escribir claramente programas organizados que sean fáciles de corregir. A continuación tienes instrucciones detalladas con las que aprenderás a usar el BASIC para crear técnicas de programación realmente complejas.

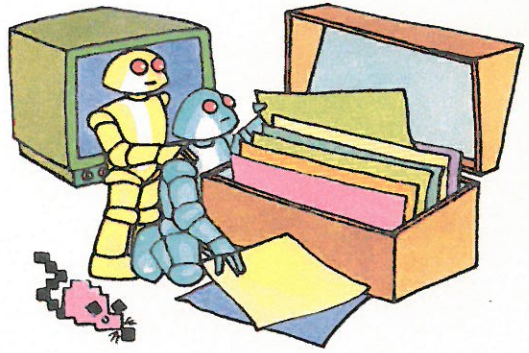


El libro contiene también multitud de rutinas que puedes añadir a tus propios programas. Hay rutinas para decorar la pantalla y para crear menús que parezcan profesionales, y para otras muchas cosas como comprimir los datos en la memoria para ahorrar espacio.

Si eres principiante conviene que leas un libro para principiantes como la *Introducción Plesa o la Programación de computadoras* antes de comenzar con este libro.



En las páginas 35-42 tienes un largo programa denominado Archivo-Clasificador que utiliza muchos de los trucos y rutinas descritos en este libro. El archivo clasificador es un archivo de tarjetas que puede almacenar nombres y direcciones o cualquier otra información. Junto al programa encontrarás explicaciones detalladas de cómo funciona, para que puedas estudiar las técnicas de programación en acción.



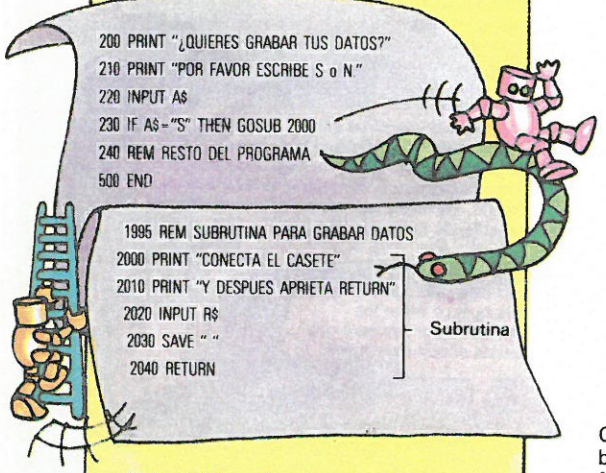
Todos los programas y rutinas de este libro están escritos en una forma standard de BASIC que, con mínimas variaciones, puede ejecutarse en casi todas las marcas principales de computadoras domésticas. Al final del libro hay una Guía BASIC con explicaciones del vocabulario BASIC más usado, y donde es necesario, los cambios que hay que hacer en las computadoras más conocidas.

¿Qué son los trucos de programación?

Un truco de programación es una forma clara y breve de realizar algo dentro de un programa. Muchos de los trucos de este libro son subrutinas que pueden usarse en gran cantidad de programas diferentes. En las próximas páginas encontrarás cómo escribir un programa utilizando una serie de subrutinas que desempeñen funciones diferentes. Esto permite escribir un programa muy largo en forma de una sucesión de programas cortos. A menudo puedes incluso usar una misma subrutina para que realice una determinada función en multitud de programas diferentes. Los profesionales que se dedican a escribir programas casi nunca escriben un nuevo programa sin acudir a subrutinas ya hechas.

¿Qué es una subrutina?

Una subrutina, también denominada rutina, es un grupo de líneas de programa que se encargan de realizar una determinada función. Puede situarse en cualquier lugar del programa y ejecutarse tantas veces como quieras. Cada vez que quieras que la computadora realice esa función usa la instrucción BASIC, GOSUB, con el número de la primera línea de la subrutina.

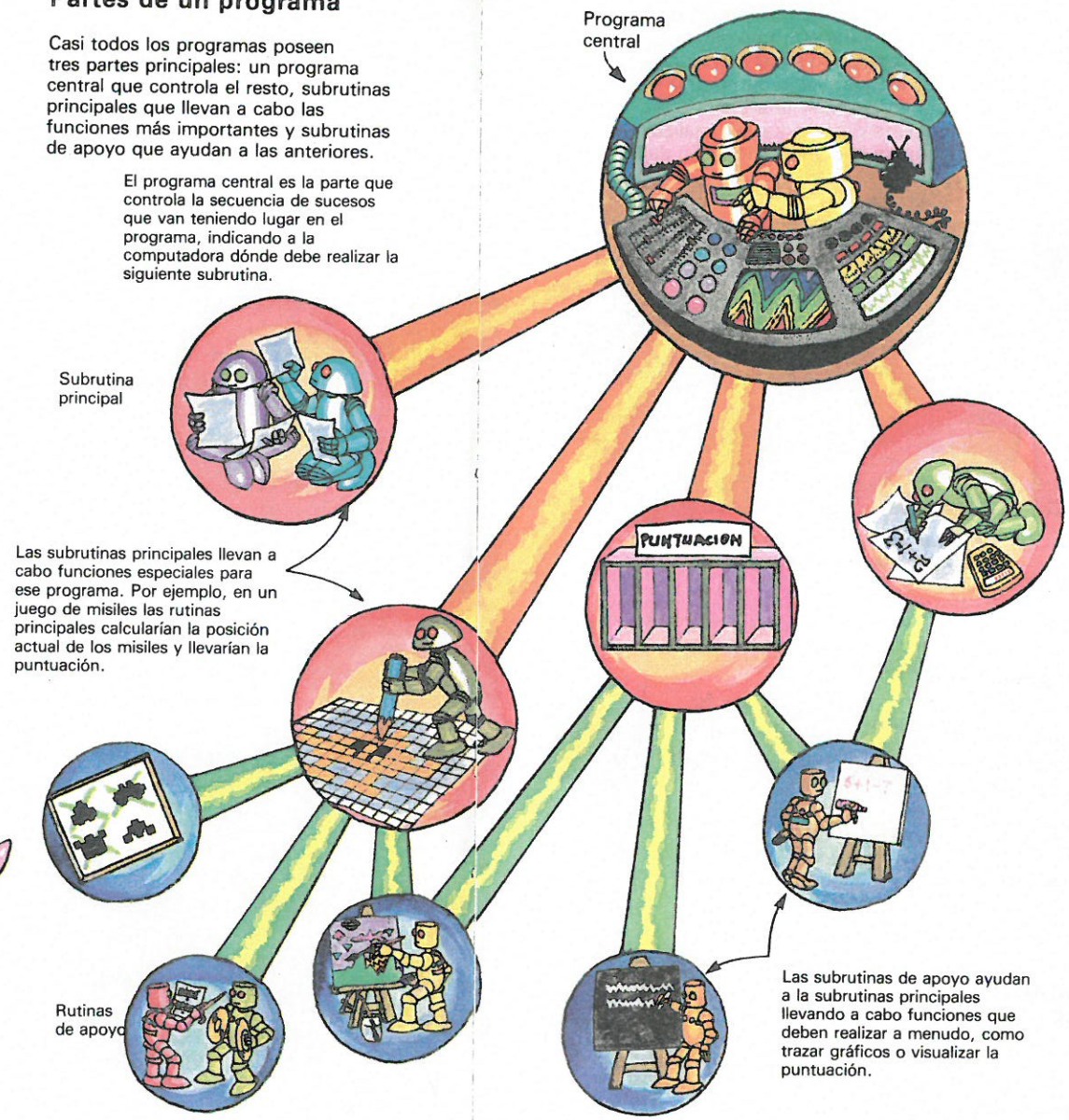


Al final de la subrutina necesitas la instrucción RETURN. Esta le dice a la computadora que vuelva a la instrucción que haya después de GOSUB.

Partes de un programa

Casi todos los programas poseen tres partes principales: un programa central que controla el resto, subrutinas principales que llevan a cabo las funciones más importantes y subrutinas de apoyo que ayudan a las anteriores.

El programa central es la parte que controla la secuencia de sucesos que van teniendo lugar en el programa, indicando a la computadora dónde debe realizar la siguiente subrutina.



Las subrutinas principales llevan a cabo funciones especiales para ese programa. Por ejemplo, en un juego de misiles las rutinas principales calcularían la posición actual de los misiles y llevarían la puntuación.

Las subrutinas de apoyo ayudan a la subrutinas principales llevando a cabo funciones que deben realizar a menudo, como trazar gráficos o visualizar la puntuación.

Cuando planees y escribas un programa busca funciones que deban realizarse frecuentemente en distintas etapas del programa. Por ejemplo, visualizar un mensaje en la pantalla o retardarlo. Colocar estas funciones en subrutinas de apoyo hará que tu programa sea más corto, rápido y fácil de entender. No importa la longitud de las rutinas —una subrutina para retardar puede hacerse en una simple línea que contenga un bucle. Para ver más funciones que conviene realizar con las rutinas de apoyo, estudia el programa Archivo-Clasificador del final de este libro.

Trucos REM

Para acordarte de para qué sirve cada subrutina ponle una etiqueta con un paso REM. La computadora ignora todo lo que se escribe detrás de un REM.

Cuando hayas terminado el programa y veas que funciona bien puede interesarte quitar los pasos REM para ahorrar memoria.

```
200 GOSUB 500

495 REM SUBRUTINA PARA RETARDAR
500 FOR J=1 TO 200:NEXT J
510 RETURN
```

Para hacerlo más fácil coloca el paso REM en la línea anterior a la que mande la instrucción GOSUB correspondiente; para que de esta forma no tengas que renumerar el programa.

```
200 GOSUB 500

500 FOR J=1 TO 200:REM DELAY
510 NEXT J
520 RETURN
```

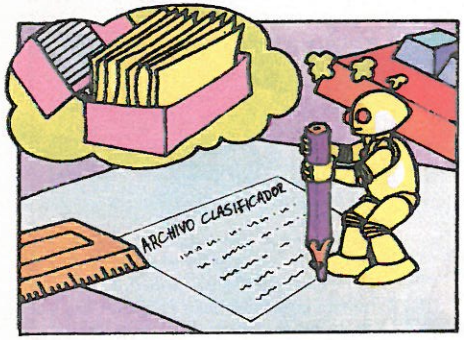
También puedes poner el REM como parte final de una línea con varias instrucciones, tal y como se muestra arriba.

Hacer una biblioteca de subrutinas

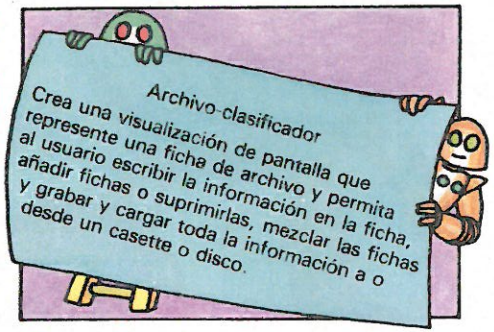
Es una buena idea conservar subrutinas útiles que puedas usar en programas diferentes. Además de escribir tus propias subrutinas puedes estudiar las que encuentres en programas ya hechos, libros o revistas y anotar aquellas que te parezcan útiles.

Técnicas para planear programas

Perder algo de tiempo al planear el programa puede ahorrarte más adelante muchas horas al intentar corregirlo. En los centros profesionales de software, una persona llamada analista de sistemas se encarga de planear el programa hasta el último detalle antes de pasárselo a un codificador que lo convierte en lenguaje de programación. Si estás escribiendo tu propio programa te será suficiente con escribir una breve lista «funciones de la computadora» de las funciones principales que tendrá que realizar la computadora, y a partir de ésta desarrollar un programa en la computadora. En estas dos páginas encontrarás algunas ideas que te pueden servir.



Antes de anotar las funciones tendrás que tener una idea clara de lo que quieres que haga el programa. Es una buena idea escribir



una breve descripción del programa Archivo-Clasificador que está listado al final del libro.

Funciones del *Archivo-Clasificador*

Una vez que tengas una idea clara de lo que hará el programa puedes escribir la lista de funciones. Esta debe mostrar las principales funciones que la computadora tendrá que realizar al ejecutar el programa. Arriba tienes la lista de funciones del programa Archivo-Clasificador. En esta etapa procura que sea sencilla; siempre podrás añadir cosas más adelante.

El esqueleto de un programa

El siguiente paso consiste en escribir un esqueleto de tu programa. Este tiene que ser una versión simplificada del programa, que muestre todas sus características pero que aún no sea capaz de llevar a cabo ninguna de las funciones. Esto lo puedes hacer usando «subrutinas sin sentido» como las que se muestran en la versión esqueleto del Archivo-Clasificador que hay abajo.

Una subrutina sin sentido es aquella en la que mediante una instrucción GOSUB mandas a la computadora a una determinada línea y en esa línea escribes un mensaje que aparecerá en la pantalla recordándote que es donde debería estar la subrutina. A continuación irá una instrucción RETURN. Usando subrutinas sin sentido puedes probar el programa para ver si la compturadora lleva a cabo las diferentes funciones en el orden correcto.

Esqueleto del *Archivo-Clasificador*

Programa central

```
10 PRINT "ARCHIVO-CLASIFICADOR"
20 PRINT "DESEAS"
30 PRINT "1.AÑADIR UNA FICHA"
40 PRINT "2.SUPRIMIR UNA FICHA"
50 PRINT "3.HACER CAMBIOS EN UNA FICHA"
60 PRINT "4.BUSCAR EN LAS FICHAS"
70 PRINT "5.CARGAR FICHAS"
80 PRINT "6.ARCHIVAR ESTAS FICHAS"
90 PRINT "INTRODUCE EL NUMERO DE TU ELECCION"
100 PRINT "A CONTINUACION APRIETA RETURN"
110 INPUT CH
120 ON CH GOSUB 300,400,500,600,700,800
130 PRINT "APRIETA RETURN PARA CONTINUAR"
140 INPUT K$
150 GOTO 10
```

Introduce el programa y pregunta al usuario lo que desea hacer. Esta parte del programa se denomina menú. En las páginas 30-31 tienes más cosas sobre menús.

Rutina input. La variable CH contiene la elección del usuario. La instrucción ON GOSUB manda a la computadora a la subrutina adecuada para ejecutar la elección del usuario. En la página 30 tienes más cosas sobre ON GOSUB*.

Hace que la computadora espere hasta que aprietes una tecla.

Repite el programa para que el usuario pueda hacer otra elección (salvo si eligió la opción grabar, ya que en este caso el programa termina en la línea 810).

Rutinas principales

```
295 REM SUBRUTINA PARA AÑADIR FICHAS
300 PRINT "SUBRUTINA AÑADIR"
310 RETURN
395 REM SUBRUTINA PARA SUPRIMIR FICHAS
400 PRINT "SUBRUTINA SUPRIMIR"
410 RETURN
495 REM SUBRUTINA PARA HACER CAMBIOS
500 PRINT "SUBRUTINA DE CAMBIOS"
510 RETURN
595 REM SUBRUTINA PARA BUSCAR EN LAS FICHAS
600 PRINT "SUBRUTINA BUSCAR"
610 RETURN
695 REM SUBRUTINA PARA CARGAR FICHAS
700 PRINT "SUBRUTINA CARGAR"
710 RETURN
795 REM SUBRUTINA PARA GRABAR FICHAS
800 PRINT "SUBRUTINA GRABAR"
810 END
```

Subrutinas principales sin sentido para cada una de las funciones. En cada subrutina la computadora escribe un mensaje en la pantalla para después volver a la línea 120.

Al final del programa central tienes que poner algo que evite que la computadora siga ejecutando las subrutinas. Puedes hacer esto haciendo que se ramifique de vuelta al menú, como en este ejemplo, o simplemente añadiendo la instrucción STOP o END.

Deja huecos entre los números de líneas para que puedas añadir más líneas.

Puedes escribir el esqueleto del programa en el teclado o en un trozo de papel. Una vez que tengas el esqueleto y que funcione bien, puedes ir incluyendo cada subrutina como unidades separadas para después incorporarlas al programa y comprobarlas. Según escribas las subrutinas principales busca funciones que puedan ser desempeñadas por rutinas de apoyo. Es perfectamente válido mandar a la computadora a una subrutina desde otra subrutina. Puedes usar también subrutinas sin sentido en lugar de las subrutinas de apoyo, hasta que estés preparado para completarlas.

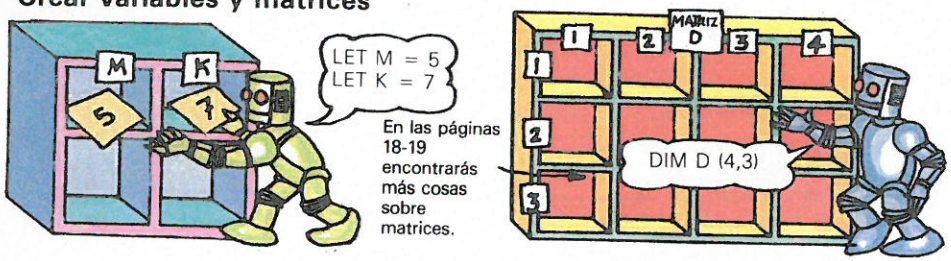
*El Spectrum no tiene la instrucción ON GOSUB. Mira en la Guía BASIC.

Para empezar

Antes de comenzar a escribir las subrutinas para completar el esqueleto del programa debes decidir dos cosas. Cómo vas a almacenar los datos para el programa y cómo quieres que aparezca la información en la pantalla*.

Los datos pueden almacenarse en variables o en matrices. Una variable es un espacio etiquetado de la memoria de la computadora donde puede almacenarse un solo dato. Una matriz es una forma de almacenar varios datos bajo un solo nombre de variable.

Crear variables y matrices



En las páginas 18-19 encontrarás más cosas sobre matrices.

Debes dar a la computadora valores iniciales para las variables. Si vas a usar una matriz debes indicar cuántos datos quieres que contenga. Esto se denomina dimensionar una matriz. El proceso de creación de las variables y de las matrices se denomina inicialización y es la primera función que debe realizar la computadora. No obstante es mejor colocar la rutina de inicialización al final del programa con una instrucción GOSUB al principio.

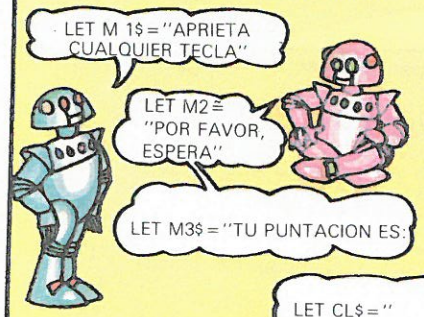
```
10 GOSUB 1500
20 REM RESTO DEL PROGRAMA
1495 REM SUBRUTINA DE INICIALIZACION
1500 LET K=10:LET T=100
1510 DIM D(4)
1520 FOR J=1 TO 4
1530 READ D(J)
1540 NEXT J
1550 DATA 25,15,39,22
1560 RETURN
```

Bucle para introducir los datos en la matriz.

Si la colocas al principio, el programa irá más lento por la forma en que la computadora busca las subrutinas (ver página 10). Si tienes que usar el módulo de gráficos conviene hacerlo al principio de la rutina de inicialización ya que puede afectar a la memoria disponible para almacenar datos.

Trucos con variables

Ahorrarás memoria y harás que los programas sean más rápidos si usas variables para cualquier dato que se use varias veces en el programa. Puedes hacer esto con las constantes, es decir, con los datos que no cambian, así como con los datos que varían durante el programa.



Por ejemplo puedes usar variables para mensajes que se repitan varias veces en el programa. Si tienes muchos mensajes puedes introducirlos todos en una matriz y acudir a ellos mediante su posición en la matriz. Por ejemplo, PRINT M$(5).

Una variable que contenga una sucesión de guiones puede ser útil para dibujar líneas en la pantalla o subrayar títulos.

Puedes usar PRINT TAB (o la palabra específica de tu computadora)** para dibujar una línea donde quieras. Para borrar líneas o mensajes usa una variable que contenga tantos espacios en blanco como espacios contenga la línea o como caracteres haya en el mensaje.

*En las páginas 28-29 tienes ideas para diseñar pantallas.

**Ver Guía BASIC.

Subrutinas de uso normal

Una vez que funciona el esqueleto de tu programa y has añadido la rutina de inicialización, puedes empezar a escribir las subrutinas principales. Según las escribas busca funciones que deban ser realizadas por varias subrutinas principales e intenta escribirlas en forma de rutina de apoyo.

Las rutinas de apoyo deben ser rutinas de uso normal que no contengan ningún dato específico para una parte del programa. Puedes hacer esto usando variables en las rutinas de apoyo y dando a estas variables valores específicos antes de mandar a la computadora a la subrutina.

Rutina sobre mensajes muy usados

```
200 LET M$="APRIETA CUALQUIER TECLA"
210 GOSUB 2000
.
.
530 LET M$="¿OTRO INTENTO?"
540 GOSUB 2000
550 END
1995 REM SUBRUTINA MENSAJE
2000 PRINT M$
2010 INPUT RESPUESTA$
2020 RETURN
```

En el Archivo-Clasificador existe una rutina de mensajes como ésta en las líneas 1135-1160.

Una subrutina mensaje como ésta puede usarse en cualquier parte del programa, o incluso en otros programas, ya que cada vez que las usas puedes introducir mensajes diferentes en la variable M$.

Rutina de gráficos muy usados

```
130 LET SIZE=10:LET X=12:LET Y=12
140 GOSUB 300
150 END
295 REM SUBRUTINA PARA DIBUJAR UN CUADRADO
300 PLOT X, Y
310 DRAW X+LADO, Y
320 DRAW X+LADO, Y+LADO
330 DRAW X, Y+LADO
340 DRAW X, Y
350 RETURN
```

Tendrás que adoptar las instrucciones de gráficos que correspondan a tu computadora.

Esta rutina dibuja un cuadrado con lados que miden LADO unidades, en posición de pantalla X, Y. Para dibujar cuadrados de diversos tamaños en lugares diferentes das a las variables LADO, X e Y diferentes valores cada vez que uses la subrutina.

Orden de las subrutinas

Según va creciendo el programa se hace más difícil ver para qué sirve cada parte. Por ello puede ayudarte el ordenar las subrutinas siguiendo un orden lógico. Estos son algunos consejos.

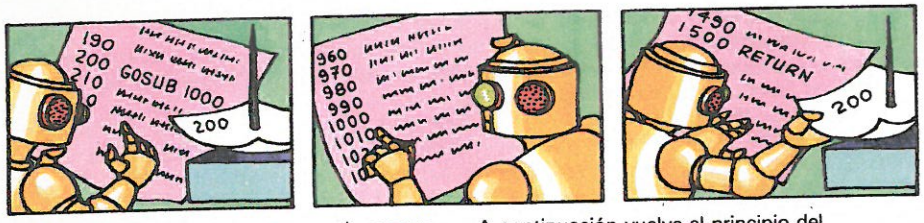
1. Pon en primer lugar todas las subrutinas principales y, a continuación, todas las de apoyo. Si las mezclas te será más difícil distinguir las partes principales del programa.

2. Dentro de estos dos grupos coloca juntas todas las subrutinas que realicen funciones similares, o bien en el orden en que sean usadas por la computadora.

Truco para corregir

Si tu programa se interrumpe con un mensaje de error y no puedes localizarlo usa PRINT como una instrucción directa (es decir, sin número de línea) para obtener el valor de las variables. Por el contenido de las variables podrás localizar lo que causó el error. Este truco para corregir es muy útil cuando haces tus propios programas.

Cómo localiza la computadora las subrutinas



Todas las líneas de un programa se almacenan en la memoria de la computadora que las va ejecutando una a una por orden. Cuando llega a una instrucción GOSUB hace una señal en una zona especial de la memoria denominada pila GOSUB, reteniendo la ubicación de memoria en la que está almacenada la instrucción GOSUB.

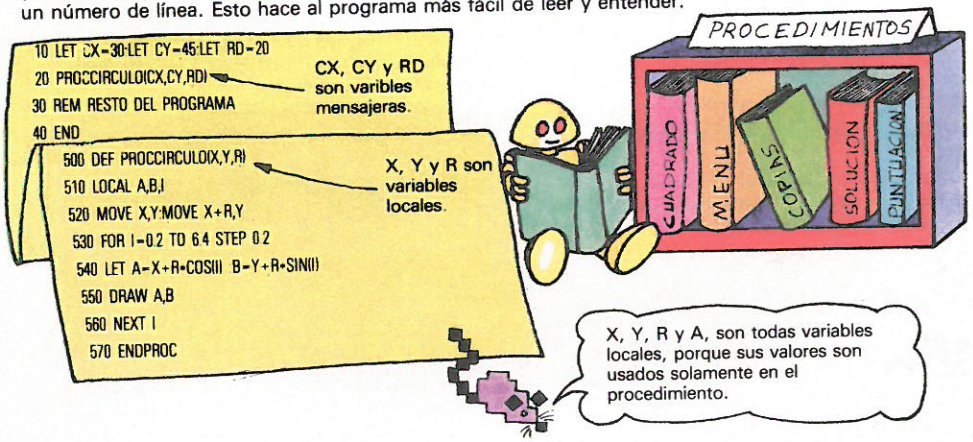
A continuación vuelve al principio del programa y lee cada número de línea hasta que encuentra la subrutina buscada. Lleva a cabo las instrucciones de la subrutina hasta que llega a una instrucción RETURN. En este momento lee la pila GOSUB para saber dónde debe volver y se dirige inmediatamene a esa parte del programa.

Truco para subrutinas

Si tienes una subrutina que deba ejecutarse muy rápidamente (por ejemplo el trazado de un gráfico en unos dibujos animados), colócalo al principio del programa ya que la computadora empieza a buscar desde el principio y lo encontrará antes. Necesitarás una instrucción GOTO delante de la subrutina para hacer que la computadora la salte al empezar a ejecutar el programa.

Otro tipo de subrutinas

Algunas computadoras tienen otro tipo de subrutinas conocido como procedimiento*. Un procedimiento es como una subrutina salvo que se accede a él por un nombre en lugar de por un número de línea. Esto hace al programa más fácil de leer y entender.

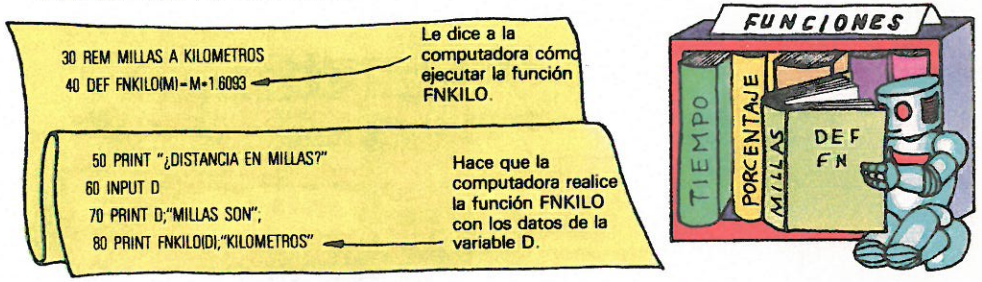


PROC es la abreviatura de «procedimiento dice a la computadora que lleve a cabo el procedimiento denominado CIRCULO usando los valores de las variables CX, CY y RD. DEF RROC es la abreviatura de «procedimiento definido» y las líneas 500-570 le dicen a la computadora que lo lleve a cabo. END PROC es como RETURN y le dice a la computadora que vuelva a donde abandonó el programa principal. Las variables CX, CY y RD se usan como mensajeras para llevar valores a las variables X, Y y R en el procedimiento. X, Y y R son variables locales, lo que significa que sus valores se usan sólo en el procedimiento. Puedes usar los mismos nombres de variables en cualquier otra parte del programa y la computadora no los confundirá. Este rasgo hace que los procedimientos estén como aislados del programa y sirvan como rutinas de uso general.

*El Commodore 64 con el BASIC Simon, y el BBC y el Lynx tienen procedimientos.

Definir tus propias funciones

Una función es un cálculo que la computadora puede realizar con datos numéricos o variables de cadena. Existen varias instrucciones BASIC con funciones. Por ejemplo, la función SQR halla la raíz cuadrada de un número y LEN el número de caracteres en una cadena. Muchas versiones del BASIC te permiten definir tus propias funciones y referirte a ellas por un nombre.



DEF FN es la abreviatura de «función definida» y PRINT FN nombre le dice a la computadora que ejecute la función. En el ejemplo superior la variable D es una variable mensajera. Lleva los datos a la variable M en la línea DEF FN. La variable M es una variable local. Puedes tener otra variable M en el programa y no se vería afectado por el valor de M en la función.

Funciones útiles

Estas son algunas funciones que pueden usarse en muchos programas.

Pon las líneas DEF FN en la inicialización del programa.

Fahrenheit a Centígrado

```
DEF FNCENT(F)=(F-32)/1.8
```

Convierte la temperatura Fahrenheit F a grados centígrados.

Centígrado a Fahrenheit

```
DEF FNFAHR (C)=C*1.8+32
```

Convierte la temperatura en grados centígrados C a Fahrenheit.

Kilómetros a millas

```
DEF FNMILES(K)=K/1.6093
```

Convierte kilómetros K en millas.

Porcentajes

```
DEF FNPERCENT (A,B)=A/B*100
```

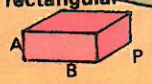
Calcula A como porcentaje de B.

Area de un rectángulo


```
DEF FNAREA (A,B)=A*B
```

Calcula el área de un rectángulo de base B y altura A.

Volumen de un sólido rectangular




```
DEF FNVOL (B,P,A)=B*P*A
```

Calcula el volumen de un sólido de base B, profundidad P y altura A.

Circunferencia de un círculo


```
DEF FNCIRC(R)=(R*2)*3.14159265
```

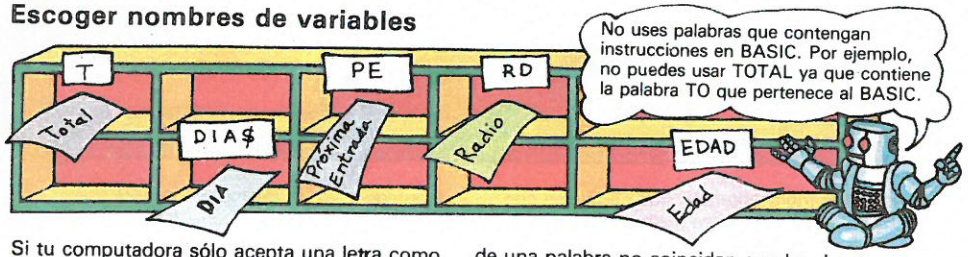
Calcula la circunferencia de un círculo de radio R.

En algunas computadoras el nombre de la función debe ser de una sola letra, y sólo puedes usar una variable después del nombre de la función.

Trucos con variables

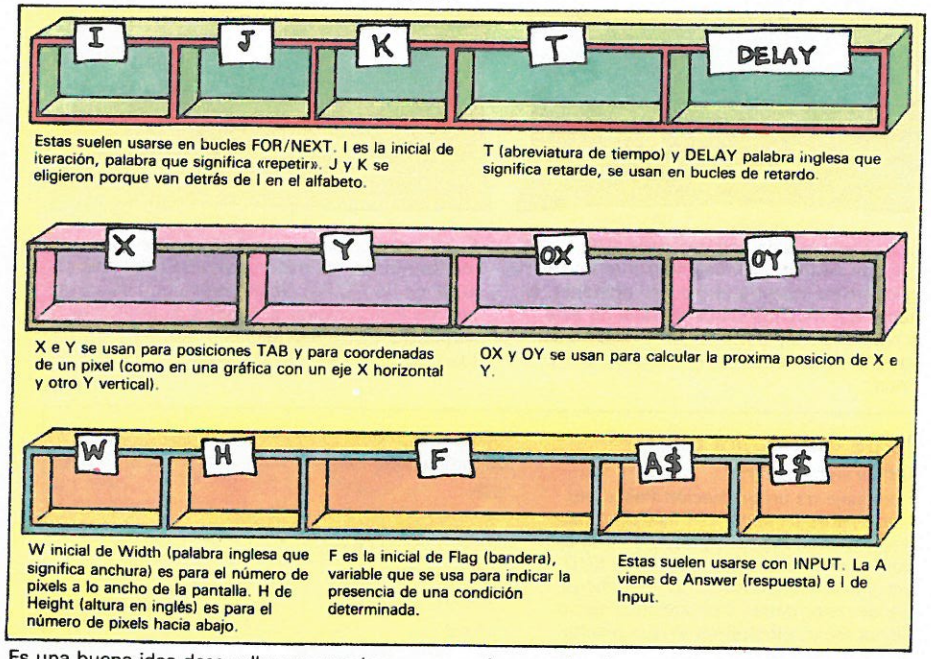
Cuando hagas tus programas te será útil usar palabras enteras como nombres de variables para que el programa resulte más fácil de leer y entender. No obstante si son demasiado largas ocuparán demasiada memoria y harán que el programa sea más lento, por lo que una vez terminado conviene que las reduzcas a una o dos letras. Algunas computadoras no te permiten usar palabras, o reconocen sólo las dos primeras letras*. Aquí encontrarás algunos consejos para elegir y usar variables.

Escoger nombres de variables



Si tu computadora sólo acepta una letra como nombre de variables, utiliza la primera letra de una palabra que describa la variable, por ejemplo T por «total». Si puedes usar palabras hazlo pero con palabras cortas. Sin embargo tienes que tener cuidado en algunas computadoras de que las dos primeras letras de una palabra no coincidan con las de otra. De todas maneras esto siempre te será útil ya que permite abreviar fácilmente los nombres una vez que has finalizado el programa. También puedes usar nombres mnemotécnicos, es decir, abreviaturas que te recuerden la misión de la variable, por ejemplo LGT por longitud.

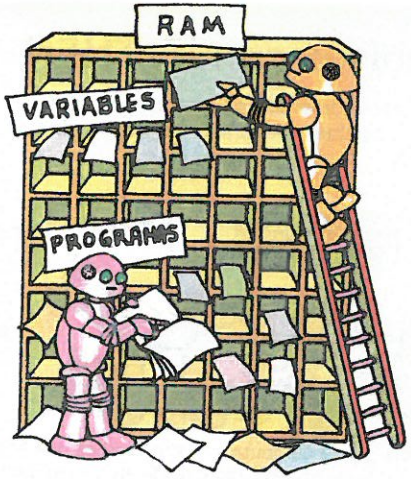
Nombres habituales de variables



Es una buena idea desarrollar tus propias variables y usarlas siempre para las mismas funciones. De esta manera las reconocerás inmediatamente y te será más fácil usar las mismas subrutinas en programas diferentes. Existen ciertas letras que la gente suele usar como nombres de variables. Estas te las mostramos en el dibujo superior.

*Comprueba las reglas especiales de tu computadora.

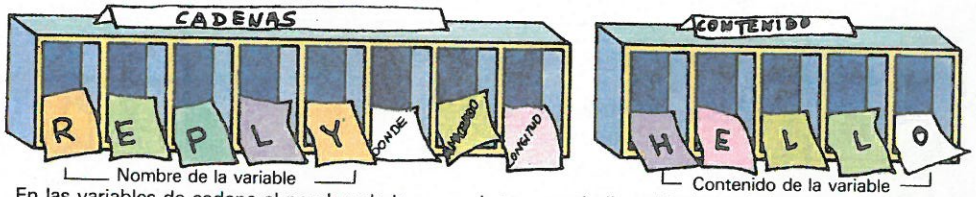
Dentro de la memoria de la computadora



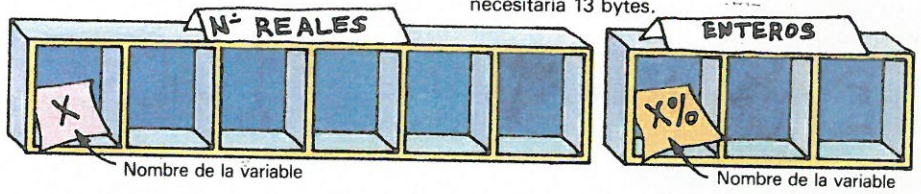
Las variables se almacenan en una zona especial del RAM (memoria de acceso directo) separados del programa. Esto permite que mientras ejecuta el programa la computadora puede mirar rápidamente el valor de una variable. Esta es la única razón por la que no puedes usar todo el RAM para almacenar tu programa. Por ejemplo, si tienes 16K de RAM, sólo unos 12K pueden estar disponibles para tu uso. El resto se usa para almacenar variables y como zona de trabajo en la que la computadora ejecuta el programa.

Cómo se almacenan las variables

La memoria de la computadora es como muchas cajas pequeñas numeradas. Las cajas se denominan ubicaciones y los números direcciones. Cada caja puede almacenar un byte en código de máquina, es decir, el código para una letra, número o símbolo. El tamaño de la memoria se mide en bytes y 1024 bytes hacen 1K (K quiere decir Kilobyte).



En las variables de cadena el nombre de la variable se almacena en un lugar diferente al contenido. Cada letra del nombre ocupa un byte. Además la computadora necesita dos bytes para indicar dónde está almacenada la cadena, y otro para anotar su longitud. La cadena en sí necesita un byte para cada carácter. La variable REPLY$ superior necesitaría 13 bytes.

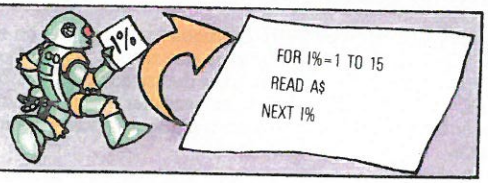


La computadora almacena números con una precisión de nueve decimales. Usa un sistema especial de codificación que en muchas computadoras ocupa cinco bytes, más uno por cada carácter del nombre. Usa cinco bytes para el número aunque éste sea un número entero de una cifra, es decir, un número sin decimales. Los números sin decimales se denominan números enteros y los que llevan decimales números reales.

Algunas computadoras tienen unas variables especiales para números enteros, que se denominan variables enteras*. Estas variables llevan un signo % detrás del nombre y ocupan sólo dos bytes. Si tu computadora tiene variables enteras úsalas siempre que puedas ya que ahorran memoria y hacen más rápido el programa.

Truco con bucles

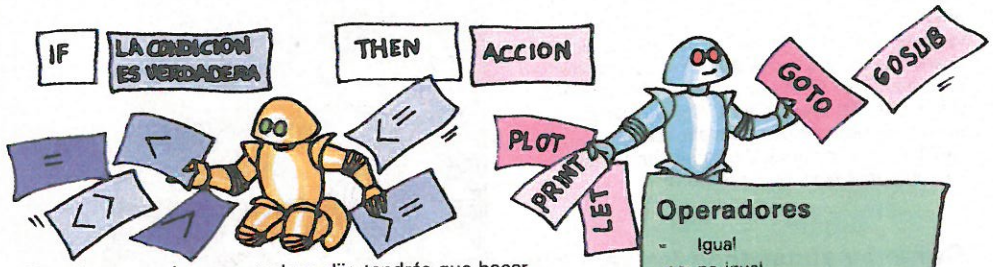
Si tu computadora lo permite, usa variables enteras en los bucles FOR/NEXT. Ocupan menos memoria y hacen más rápido el programa.



*El Vic-20, Commodore 64, BBC, Electron y Apple tienen variables enteras.

Técnicas con IF/THEN

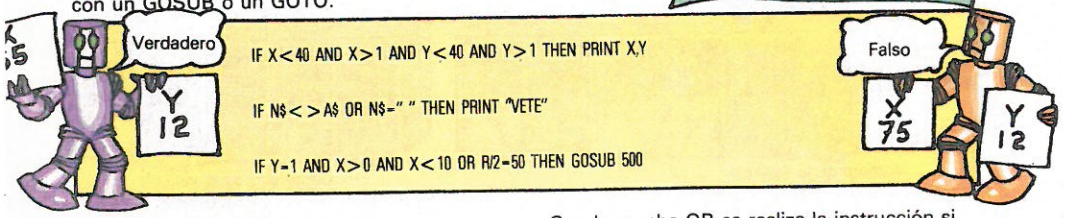
En algunos momentos del programa necesitarás que la computadora escoja entre varias acciones posibles, dependiendo de ciertas condiciones. Esto se hace usando las palabras IF/THEN.



Operadores

- = igual
- < > no igual
- > mayor que
- < menor que
- > = mayor o igual que
- < = menor o igual que

Para hacer que la computadora elija tendrás que hacer comprobaciones con los datos mediante el uso de los signos de la derecha. Estos signos se denominan operadores. Si la prueba es verdadera, la computadora realizará la instrucción que halla detrás de THEN. Cualquier instrucción BASIC puede ir detrás del THEN o también puedes mandar a la computadora a otra línea con un GOSUB o un GOTO.




```
IF X<40 AND X>1 AND Y<40 AND Y>1 THEN PRINT X,Y

IF N$<>A$ OR N$="" THEN PRINT "VETE"

IF Y=1 AND X>0 AND X<10 OR R/2=50 THEN GOSUB 500
```

Puedes crear comprobaciones más complicadas usando las palabras AND y OR tal y como se ve arriba. Con AND, la instrucción THEN sólo se ejecutará si ambas pruebas son verdaderas.

Con la prueba OR se realiza la instrucción si cualquiera de las condiciones es verdadera. Recuerda que tienes que repetir el nombre del dato detrás de cada OR o AND.

```
50 IF A=10 THEN PRINT "CORRECTO" ELSE PRINT "MAL"

80 IF A=10 THEN PRINT "CORRECTO"
90 IF A<>10 THEN PRINT "MAL"
```

La palabra ELSE te permite decir a la computadora sin necesidad de repetir una línea IF/THEN lo que debe hacer si la prueba no es verdadera. Por ejemplo, aquí la línea 50 realiza la misma función que la 80 y la 90 pero es más corta. (No todas las computadoras tienen esta instrucción).

```
IF X<1 THEN LET X=ABS(X) ELSE IF X>100
THEN LET X=100 ELSE IF X=Y THEN LET Y=Y+2
```

```
IF X=5 AND (Y=2 OR Y=4) THEN PRINT "VERDADERO"
IF X=5 AND Y=2 OR Y=4 THEN PRINT "VERDADERO"
```

Puedes unir muchos IF/THEN mediante el uso de ELSE pero cuida de no hacer que resulte demasiado difícil de leer. Si usas demasiados AND y OR en una línea puedes necesitar paréntesis para indicar a la computadora cómo leerla.

En el primer ejemplo superior la computadora sólo realizará la instrucción THEN si X es 5 e Y es ó 2 ó 4. En el segundo ejemplo será verdadero cuando X sea 5 e Y sea 2, o siempre que Y valga 4.

Seguir el proceso de la acción

Uno de los problemas del BASIC es que no existe una forma clara de dar a la computadora una serie de instrucciones para realizar si la prueba IF/THEN es verdadera. La única manera de hacer esto es con una línea multisentencial (de varias instrucciones) o mandando a la computadora una subrutina, o bien saltando un grupo de instrucciones con un GOTO. En otros lenguajes como el Pascal no existe ningún límite en el número de instrucciones que puede realizar la computadora en función del resultado de una prueba.

1 Líneas multisentenciales

```
100 IF X<10 THEN LET Y=1:LET R=R+X:INPUT K:FOR J=1 TO 6: LET
X=X+K:NEXT J
```

Comprueba el número de caracteres que tu computadora acepta en una línea.

En una línea multisentencial que comience con IF, todas las sentencias se realizan sólo si la prueba IF es verdadera. Si la prueba no es verdadera no se ejecuta ninguna de las sentencias y la computadora pasa a la siguiente línea. Esta es una manera de dar a la computadora varias instrucciones para que realice después de una prueba. Sin embargo, está limitada por el número de caracteres que tu computadora acepta por línea de programa (en muchas computadoras son 255)*.

2 GOTO

```
100 IF X>1 THEN GOTO 150
110 REM SOLO SI X<1
120 REM INSTRUCCIONES
130 REM INSTRUCCIONES
140 GOTO 180
150 REM SOLO SI X>1
160 REM INSTRUCCIONES
170 REM INSTRUCCIONES
180 REM EL PROGRAMA CONTINUA
```

Va a la línea 150 si la prueba es verdadera.

Instrucción a ejecutar si la prueba no es verdadera. La línea 140 hace saltar a la computadora al siguiente grupo de instrucciones.

3 GOSUB

```
110 IF X>1 THEN GOSUB 500 ELSE GOSUB 550
120 REM EL PROGRAMA CONTINUA
400 END
500 REM SOLO SI X>1
510 REM INSTRUCCIONES
520 RETURN
550 REM SOLO SI X<1
560 REM INSTRUCCIONES
570 RETURN
```

Instrucciones para realizar si X > 1.

Instrucciones para realizar si X < 1.

Puedes hacer que la computadora salte sobre un bloque de instrucciones mediante un GOTO tal y como se muestra arriba. No obstante, si la haces dar muchos saltos de rana de este tipo resultará muy difícil seguir la acción del programa y por tanto complicado de leer y corregir.

Si las instrucciones no caben en una línea multisentencial es mejor usar una subrutina en lugar de un GOTO. Necesitarás una subrutina con las instrucciones a seguir si la prueba es verdadera y otra con las instrucciones para el caso de que la prueba no sea verdadera.

Otros lenguajes de programación

El lenguaje de programación Pascal no usa números de línea. Las instrucciones se agrupan en bloques y cada bloque se etiqueta con las palabras BEGIN y END para que la computadora las identifique. Puedes tener tantas instrucciones como quieras en un bloque por lo que resulta más fácil dar a la computadora una serie completa de instrucciones para seguir después de una prueba.

El breve programa de la derecha es un programa para obtener tablas de multiplicar.

```
TABLAS DE MULTIPLICAR
VAR BASE, NUMERO, I:INTEGER;
BEGIN
  WRITELN ('¿QUE TABLA DE MULTIPLICAR?')
  WRITELN ('¿DESEAS VER?')
  INPUTLN (BASE)
  FOR I=1 TO 12 DO
  BEGIN
    NUMERO:=BASE*I
    WRITELN (BASE,'X',I,'=', NUMERO,
  END;
END;
```

Crea las variables BASE, NUMERO e I

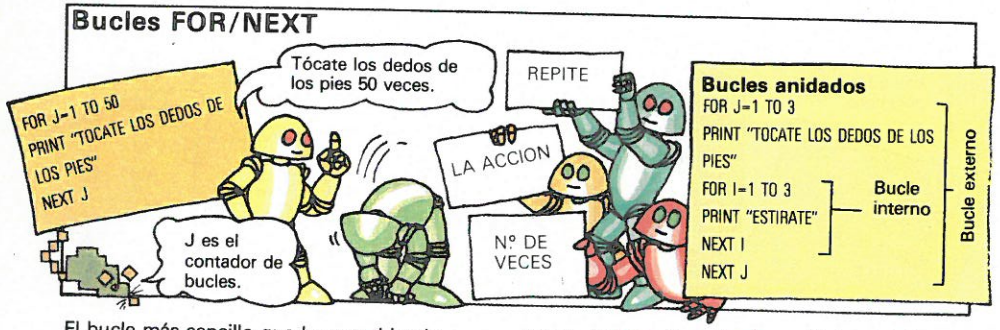
Bucle para escribir las tablas.

*El Commodore 64 admite sólo 80 caracteres y el VIC-20-88.

Repeticiones: Bucles

Además de tomar decisiones en ocasiones tendrás que hacer que la computadora repita un grupo de instrucciones un número determinado de veces. Existen varias formas de hacer con la computadora estos bucles.

Bucles FOR/NEXT



El bucle más sencillo que hay es el bucle incondicional en el que le dices a la computadora que repita una acción un número determinado de veces. Esto lo hacen con las palabras FOR, TO y NEXT como se muestra arriba. Puedes hacer bucles mucho más complejos colocando bucles dentro de bucles.

Estos se denominan bucles anidados. En los bucles anidados ambas partes del bucle interno han de encontrarse dentro del bucle externo. La variable que se encarga de contar las veces que se repite se denomina contador del bucle.

Bucles repetir/hasta

En ocasiones no sabes cuántas veces repetir el bucle y quieres que la computadora lo repita hasta que una determinada condición sea verdadera. Por ejemplo puedes querer repetir una línea input hasta que el usuario escriba ALTO, o bien hacer que la computadora busque en una matriz hasta que encuentre determinado dato. Algunas versiones del BASIC poseen las instrucciones REPEAT (Repetir) y UNTIL (hasta) pero si tu no las tienes puedes hacerlo escribiendo una rutina como las que ves abajo.

```
40 REM BUCLE REPETIR/HASTA
45 DIM D$(100)
50 LET I=0
60 LET I=I+1
70 PRINT "DATO";I
80 INPUT D$(I)
90 IF D$(I)<>"ALTO" THEN
   GOTO 60
```

```
40 REM NO HAGAS ESTO
45 DIM D$(100)
50 FOR I=1 TO 100
60 INPUT D$(I)
70 IF D$(I)="ALTO" THEN GOTO 90
80 NEXT I
90 REM RESTO DEL PROGRAMA
```

Esta rutina repite la línea input hasta que el usuario escriba ALTO. La variable I es el contador del bucle y es útil darle el valor 0 al principio del bucle no sea que se haya utilizado en alguna otra fase del programa*.

Nunca debes salir de un bucle con un GOTO. La computadora seguirá tus instrucciones pero en alguna parte de la memoria le quedará un contador de bucle a medio completar y esto puede más adelante conducir a errores.

*Para las computadoras Sinclair tienes que indicar en la línea 45 el número de caracteres del dato más largo. Ej. DIM D$ (100, 12).

Trucos para terminar bucles

```
10 REM BUCLE PARA BUSCAR EN UNA MATRIZ
20 FOR J=1 TO 40
30 IF D$(J)=R$ THEN LET K=J
40 NEXT J
50 PRINT D$(K)
```

```
10 REM VERSION ALTERNATIVA
20 FOR J=1 TO 40
30 IF D$(J)=R$ THEN LET K=J:LET J=40
40 NEXT J
50 PRINT D$(K)
```

Esto da al contador su valor final.

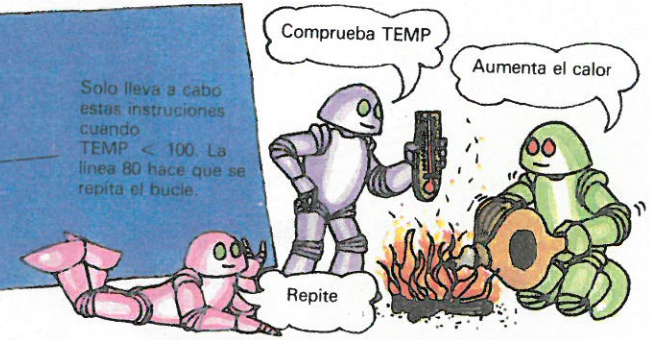
Estas dos rutinas muestran dos formas diferentes de terminar los bucles FOR/NEXT cuando la condición es verdadera. En el ejemplo de la izquierda la computadora busca en una matriz D$ hasta que encuentra un dato que coincide con R$. Graba su posición en la matriz asignando a otra variable K, el valor del contador en ese momento. Luego continúa con el resto del bucle. Si es importante la velocidad puedes terminar el bucle dando al contador su valor final, como en el ejemplo de la derecha*.

Bucles mientras

En un bucle «mientras», la computadora repite una acción mientras una determinada acción sea verdadera. Esto es parecido a un bucle repite/hasta pero la diferencia es que la computadora comprueba la condición antes de llevar a cabo la acción. Si la prueba no es verdadera no lleva a cabo las instrucciones.

```
35 REM BUCLE MIENTRAS
40 INPUT TEMP
50 IF TEMP>=100 THEN GOTO 90
60 PRINT "SUBE LA TEMPERATURA"
70 LET TEMP=TEMP+1
80 GOTO 50
90 PRINT "TEMPERATURA=";TEMP
```

Solo lleva a cabo estas instruciones cuando TEMP < 100. La línea 80 hace que se repita el bucle.

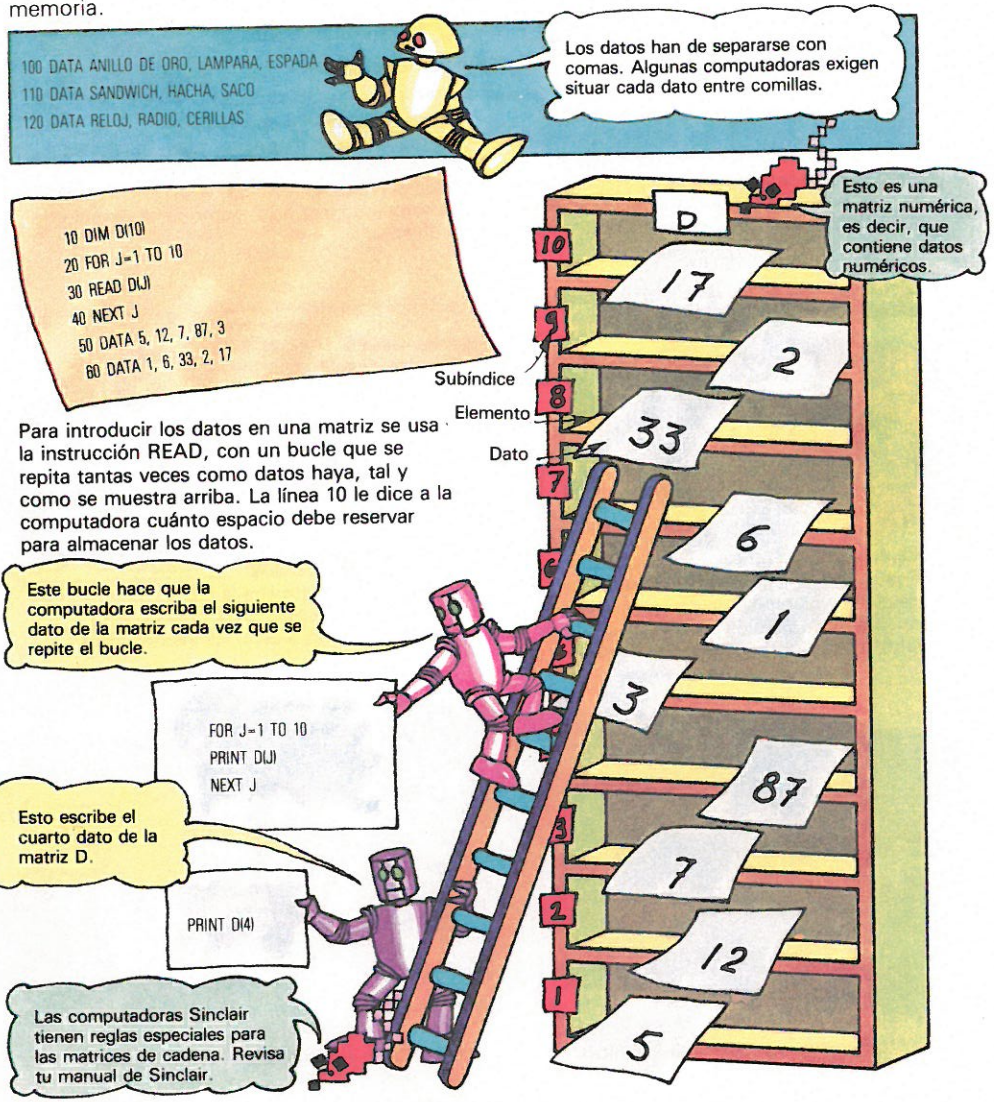


En esta rutina la computadora sólo llevará a cabo las líneas 60-80 cuando TEMP sea menor que 100. Cuando TEMP es 100 salta a la línea 90. Algunos lenguajes de programación (por ejemplo Forth y Pascal) tienen instrucciones para los bucles mientras, pero en el BASIC tienes que hacerlo mediante rutinas con dos GOTO como se muestra arriba. Esta es una de las razones por las que la gente critica el BASIC. En este sencillo ejemplo podrías evitar los dos GOTO con una línea multisentencial. 50 IF TEMP < 100 THEN PRINT «SUBE LA TEMPERATURA». LET TEMP = TEMP + 1:GOTO 50. Sin embargo en un programa más complicado en el que haya que realizar muchas instrucciones cuando TEMP sea menor que 100, necesitarás usar dos GOTO.

*Si tienes un Spectrum mira en la página 46.

Almacenar datos en matrices

La manera más habitual de introducir en la computadora largas listas de datos es mediante líneas DATA, tal y como se muestra abajo. Los datos tendrán que introducirse en variables o matrices antes de que puedan usarse en el programa. Estas dos páginas describen algunas formas en que puedes usar las matrices. En las páginas 20-26 tienes algunas ideas para almacenar datos de otras formas, así como trucos para ahorrar memoria.

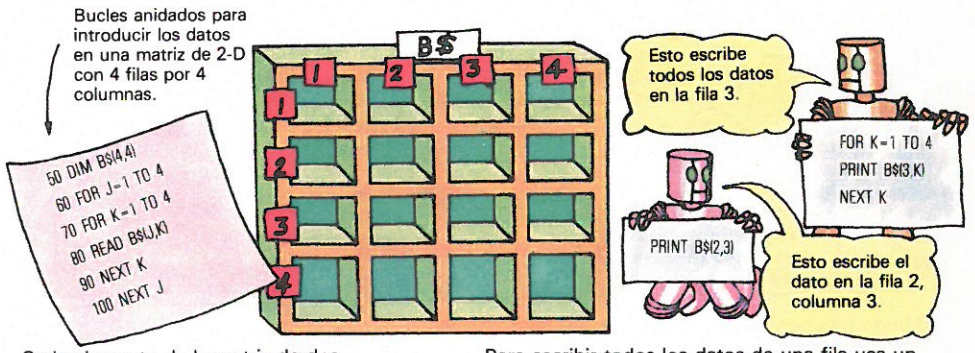


Para introducir los datos en una matriz se usa la instrucción READ, con un bucle que se repita tantas veces como datos haya, tal y como se muestra arriba. La línea 10 le dice a la computadora cuánto espacio debe reservar para almacenar los datos.

Una vez introducidos los datos en la matriz puedes leerlos con el uso de un bucle. Para sacar un solo dato usa el número de su posición en la matriz para identificarlo. Este número se denomina subíndice y lo que hay en cada posición de la matriz se denomina elemento. En este ejemplo la matriz D es una matriz unidimensional. Es decir, que es como una lista donde cada dato tiene un sólo subíndice.

Matrices de dos dimensiones

Si tienes grupos de información relacionada entre sí, como pueden ser direcciones, teléfonos y cumpleaños de varias personas, puedes usar una matriz de dos dimensiones y colocar toda la información sobre una persona en una fila. Esto hace que sea mucho más fácil encontrar toda la información sobre una persona o un simple dato como un cumpleaños.

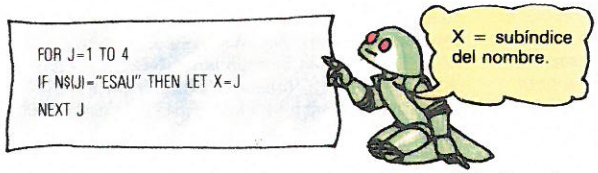


Cada elemento de la matriz de dos dimensiones tiene dos subíndices: los números correspondientes a la fila y a la columna. Para introducir los datos en la matriz necesitas bucles anidados —uno que cuente las filas y otro las columnas tal y como se muestra arriba.

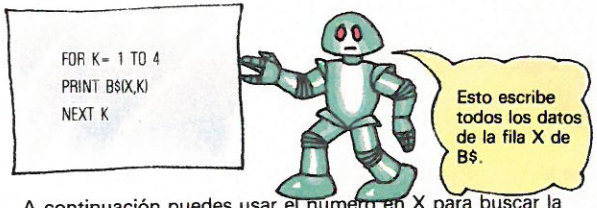
Para escribir todos los datos de una fila usa un bucle como el de arriba a la derecha. Para encontrar un dato particular sólo necesitas los números de fila y de columna.

Matrices de referencia

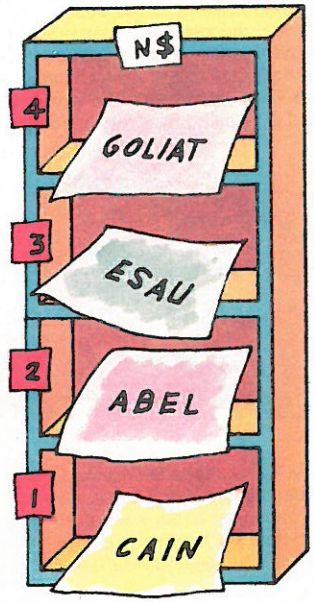
Cuando uses matrices de dos dimensiones puedes necesitar una segunda matriz para que sirva de contador de la primera. En el ejemplo superior podrías poner los nombres de las personas en una matriz y sus datos en la otra tal y como se indica arriba. Los nombres estarán en el mismo orden que las filas que contienen sus datos en la matriz de dos dimensiones. Por ejemplo, los datos para la persona en N$ (3) estarán en la fila tres de B$.



Este bucle hace que la computadora busque en N$. Cuando encuentra el nombre correcto graba en otra variable X el valor del contador del bucle, es decir, el subíndice del nombre. Esto es para evitar salirse del bucle como dijimos en la página 17.



A continuación puedes usar el número en X para buscar la fila para esa persona en la matriz de dos dimensiones y escribir todos sus datos. Para hacer esto usa un bucle como el que se muestra arriba.

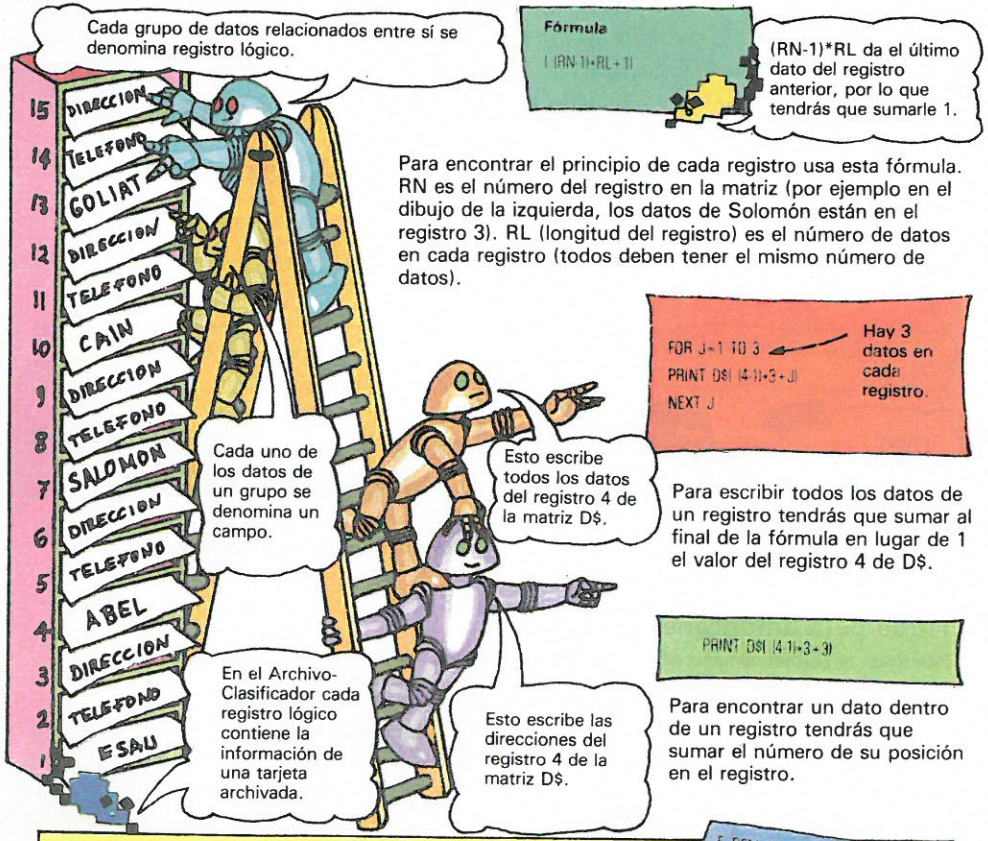


Trucos con Data

Cuando introduces datos en una matriz, los datos se almacenan dos veces en la memoria —una vez como líneas DATA en la zona donde se almacena el programa y otra en una matriz en la zona donde se almacenan las matrices y las variables. Este es el motivo por el que los programas que poseen muchos datos, como los juegos de aventuras, ocupan mucha memoria. En las siguientes páginas encontrarás algunas ideas para almacenar datos ahorrando el máximo de espacio.

Truco con las matrices para ahorrar espacio de memoria

Una matriz de dos dimensiones ocupa mucho más espacio que una matriz de una dimensión con el mismo número de elementos. Para ahorrar espacio puedes usar una matriz de una dimensión y organizar los datos de la misma manera que lo harías en una matriz 2-D, es decir, poniendo grupos de datos relacionados juntos. Para localizar cualquier dato tendrás que usar una fórmula para calcular su posición en la matriz. El programa Archivo-Clasificador al final del libro usa este método.



Para encontrar el principio de cada registro usa esta fórmula. RN es el número del registro en la matriz (por ejemplo en el dibujo de la izquierda, los datos de Solomón están en el registro 3). RL (longitud del registro) es el número de datos en cada registro (todos deben tener el mismo número de datos).

```
FOR J=1 TO 3          ← Hay 3 datos en cada registro.
PRINT D$( (4-1)*3+J)
NEXT J
```

Esto escribe todos los datos del registro 4 de la matriz D$.

Para escribir todos los datos de un registro tendrás que sumar al final de la fórmula en lugar de 1 el valor del registro 4 de D$.

```
PRINT D$( (4-1)*3+3)
```

Esto escribe las direcciones del registro 4 de la matriz D$.

Para encontrar un dato dentro de un registro tendrás que sumar el número de su posición en el registro.

Usar el elemento cero

Suele ser más fácil imaginar matrices que comiencen con el elemento 1, pero lo cierto es que casi todas las computadoras empiezan con el cero. Por ello si estás justo de memoria puedes usar el elemento 0. Para hacer esto haz que tus bucles para introducir o buscar datos comiencen con 0, y terminen con uno menos que el número de datos que haya, tal y como se muestra a la derecha.

```
5 REM 20 DATOS
10 DIM D$(19)
20 FOR J=0 TO 19      ← Bucle repetido 20 veces.
30 READ D$(J)
40 NEXT J
```

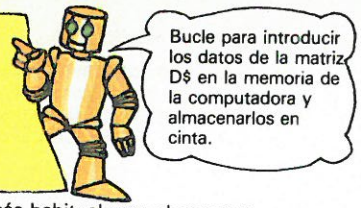
Si usas el elemento 0 en la fórmula superior no hace falta que sumes un uno.

Trucos con archivos de datos

En lugar de poner los datos en líneas DATA puedes almacenarlos en la cinta o disco y leerlos directamente con la matriz cuando ejecutes el programa. Esto ahorra mucha memoria. Una lista de datos almacenados en cinta o disco se denomina un archivo.

Tendrás que mirar las instrucciones para cargar y grabar datos en tu computadora. Estas son diferentes en cada computadora aunque todas suelen seguir un esquema parecido al que ves a la derecha. Los más habituales están listados en la Guía BASIC del final del libro.

```
OPEN nombre del archivo
FOR I=1 TO número de datos
PRINT #, D$(I)
NEXT I
CLOSE archivo
```



PRINT # es la instrucción más habitual para almacenar datos en cinta. Antes de almacenar los datos tendrás que abrir (open) un archivo y una vez finalizado el proceso, cerrarlo (close).

```
OPEN nombre del archivo
FOR I=1 TO número de datos
INPUT #,D$(I)
NEXT I
CLOSE archivo
```

INPUT # es la instrucción más habitual para cargar datos desde la cinta o disco. Igual que al grabar, debes abrir el archivo antes de empezar y cerrarlo una vez terminado el proceso.

Cómo crear un archivo de datos

1. En la rutina de inicialización dimensiona una matriz y añade las líneas necesarias para leer los datos del archivo X. Graba el programa.
2. Escribe NEW para borrar el programa de la memoria de la computadora.
3. Escribe una rutina como la que se muestra a la derecha para almacenar los datos en el archivo X.
4. Dimensiona la matriz para contener los datos.
5. Introduce los datos en la matriz.
6. Graba los datos en el archivo X.
7. Ejecuta esta rutina.

Estos son los pasos a seguir para almacenar los datos de un programa en cinta o disco en lugar de en líneas DATA en el programa. Después de seguir estos pasos tendrás una copia de tu programa en cinta o disco, así como un archivo con todos los datos. Cuando cargues y ejecutes el programa los datos se cargarán automáticamente en la matriz.

Proteger tus archivos

```
LET PW$="CLAVE"
Abrir archivo
PRINT # archivo, PW$]      ← Graba la clave
Graba datos en el mismo archivo
Cerrar archivo
```

```
PRINT "CLAVE POR FAVOR"
INPUT P$
Abrir archivo
INPUT #,PW$
IF P$<>PW$ THEN LET PW$="":Cerrar archivo:NEW
```

Si tienes una computadora Spectrum mira en la página 46.

Cuando hagas tu archivo de datos puedes incluir una clave para que sólo las personas autorizadas que conozcan la clave puedan sacar los datos. Necesitarás una rutina como la de arriba a la izquierda para grabar una clave junto con los datos. También necesitarás añadir una subrutina de comprobación de la clave al programa principal justo antes de la rutina que carga los datos. Si alguien, ejecutando el programa, introduce la clave errónea la rutina de comprobación borrará PW$, cerrará el archivo y destruirá el programa con la instrucción NEW*.

*Si tu computadora no acepta la instrucción NEW como línea de programa usa un bucle para introducir (poke) números menores que 255 en varias ubicaciones con direcciones entre 0 y 255.

Truco con Restore

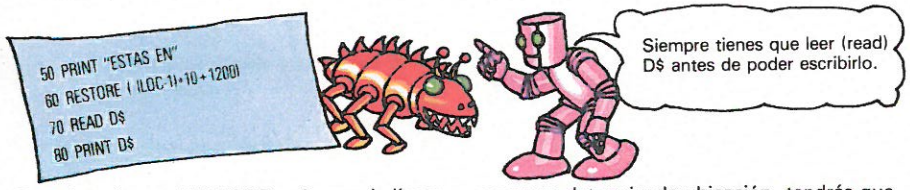
Este es otro truco para ahorrar espacio de memoria. La instrucción RESTORE hace que la computadora vaya a la primera línea de datos del programa. Este truco permite usar los datos listados en líneas DATA sin necesidad de tener que introducirlos antes en la matriz. Normalmente tienes que poner los datos en la matriz antes de usarlos.

Por ejemplo, podrías usar esta rutina en un juego de aventuras para obtener la descripción de la ubicación número X. Las descripciones están almacenadas en líneas DATA. Cada vez que se ejecute la rutina, RESTORE manda a la computadora a la primera línea de datos. El bucle hace que lea los datos hasta que J = X. Luego escribe el último dato que haya en D$ - la descripción de la ubicación X.

Este método es lento pero si no borras la pantalla para escribir D$ hasta que la computadora ha encontrado los datos, el jugador no lo apreciará.

Truco con el número de la línea RESTORE

La instrucción RESTORE «número de línea» le dice a la computadora que vuelva a una determinada línea de datos. Si tienes esta instrucción te será mucho más fácil y rápido encontrar un dato específico si usas este truco con RESTORE.



Este ejemplo usa RESTORE «número de línea» para encontrar los datos de la ubicación LOC. El ejemplo presume que cada línea contiene los datos para una ubicación. Las líneas DATA comienzan en la línea 1200 incrementándose los números de la línea de diez en diez. Para encontrar el número de la línea con los datos para una determinada ubicación, tendrás que restar uno al número de la ubicación y multiplicar el resultado por diez (ya que los números de línea van de diez en diez). Luego suma 1200, el número de la primera línea de datos.

```
45 REM CUATRO DATOS POR LINEA
50 PRINT "ESTAS EN"
60 LET DL=INT( (LOC-1)/4)
70 LET DATO=LOC-(DL*4)
80 RESTORE (1200+(DL*10))
90 FOR K=1 TO DATO
100 READ D$
110 NEXT K
120 PRINT D$
```

DL (Línea de datos) = (LOC-1)/4 ya que hay cuatro elementos por línea.

Encuentra la posición del dato en la línea restando DL*4 (el número de datos hasta la línea que contiene el dato que deseas).

Multiplica DL por diez ya que los números de línea van de diez en diez.

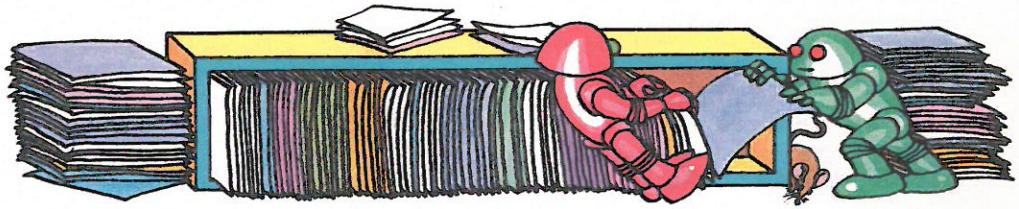
Bucle para leer cada dato de la línea hasta DATO, incluyendo éste.

Escribe el último dato leído.

Si pones varios datos en cada línea necesitarás una fórmula mucho más complicada que la superior. Puedes usar este truco en lugar del que describimos en la página 20 sobre matrices. Cada línea de datos contiene todos los datos para un registro lógico.

Amontonar datos

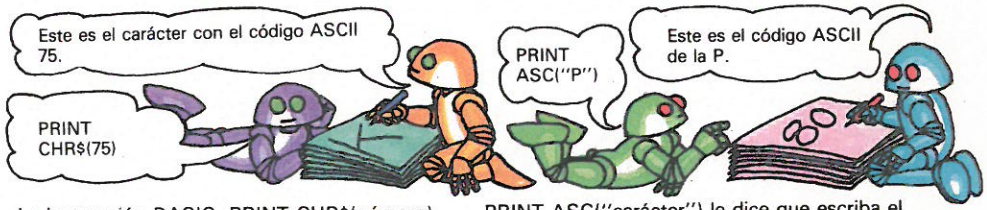
En la siguiente página hay dos métodos para ahorrar espacio cuando tienes que almacenar grandes cantidades de números como, por ejemplo, para las coordenadas de visualización de un laberinto. Sólo puedes usar estre truco con números hasta 255 ya que cada número se almacena como el carácter que dicho número representa en el código ASCII. Este es el código que la computadora usa para representar caracteres y sólo existen 256 códigos ASCII diferentes (0-255). Cada carácter ocupa un solo byte de memoria, por lo que ahorras cuatro bytes por cada número almacenado (Los números suelen ocupar cinco bytes) se denomina amontonar datos ya que los estás comprimiendo mucho en bastante menos espacio del que habitualmente ocuparían.



Código ASCII

Dentro de la computadora las letras, números y símbolos están representados por números codificados en binario. ASCII (Código standard Americano para el Intercambio de Información) es un código internacional en el que cada número representa un carácter.

Los 256 números del código ASCII se usan para representar todas las letras del alfabeto, los números 0-9, los signos de puntuación, los signos gráficos y las instrucciones especiales del teclado como RETURN o SHIFT.



La instrucción BASIC, PRINT CHR$(número) le dice a la computadora que escriba el carácter con el número de código.

PRINT ASC("carácter") le dice que escriba el código ASCII de un determinado carácter*.

Truco para amontonar números

Este truco te permite almacenar muchos números en una variable (P$). En muchas computadoras el máximo número de caracteres que puedes almacenar en una variable de cadena es 255, por lo que ésta sería la cantidad total de números que puedes almacenar.

1

```
10 DIM CANTIDAD (nº de números)
20 FOR J=1 TO nº de números
30 READ CANTIDAD (J)
40 NEXT J
50 DATA todos los números
```

2

```
60 LET P$="**********etc"
```

Tantas estrellas como números quieres almacenar.

En principio tienes que crear una variable (cantidad) e introducir todos los números en ella. Más adelante cuando todos los números estén amontonados en la memoria podrás borrar todas estas líneas.

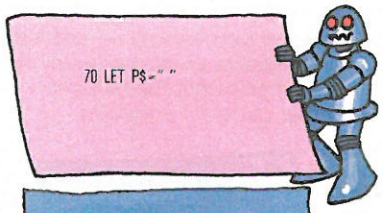
A continuación crea una variable de cadena (P$) con tantas estrellas como números tengas (máximo de 255). Esto es para reservar espacio en la memoria para los números. Necesitas una variable de cadena ya que los números estarán amontonados como caracteres.

*Las computadoras Sinclair usan las instrucciones CODE en lugar de ASC. Si tienes un Spectrum o TBS 80 mira en la página 46

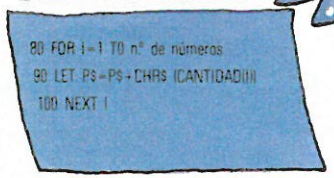
Continúa en la página siguiente

Continuación del truco para amontonar números

3



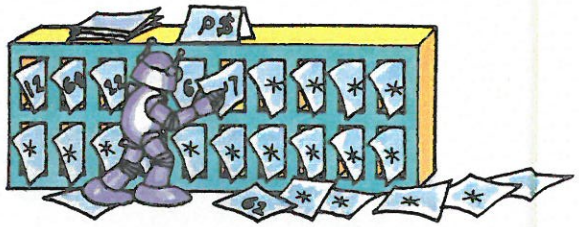
4



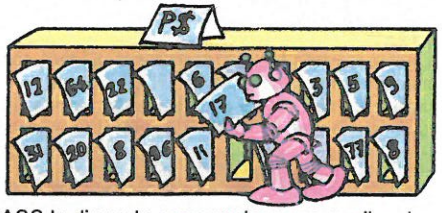
Este bucle sustituye las estrellas en P$ por los números almacenados en la matriz CANTIDAD. La instrucción CHR$ le dice a la computadora que trate los números como códigos ASCII y almacene cada número como

Ahora los números están amontonados en la memoria. Para sacar cualquier número tendrás que usar las instrucciones MID$ y ASC. En el ejemplo superior MID$ (P$, 21, 1) le dice a la computadora que tome un carácter de P$, comenzando con el carácter 21.

◀Luego «colapsa» la variable con esta instrucción que hará que la computadora piense que P$ está vacía aunque en la zona de memoria para las variables sigan conservándose las estrellitas.

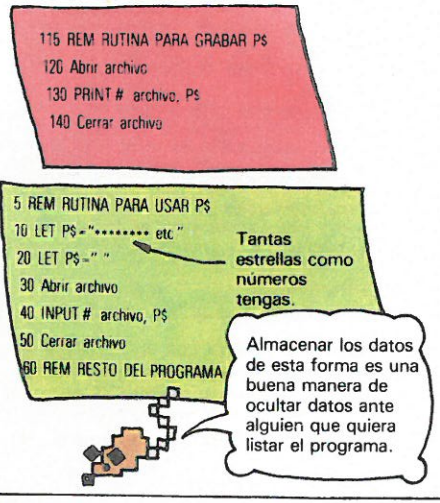


el carácter que representa. Dado que la computadora piensa que P$ está vacío almacena cada carácter en el espacio reservado por una estrella.



ASC le dice a la computadora que escriba el código ASCII que corresponde a ese carácter, tu número. Si no usas ASC la computadora puede bloquearse ya que alguno de tus números probablemente sean códigos de efectos especiales que no aparecen en la pantalla como caracteres.

Usar la rutina



Antes de poder usar este método para almacenar los datos de un programa tienes que grabar P$ como un archivo de datos tal y como se describe en la página 21 y se muestra a la izquierda.

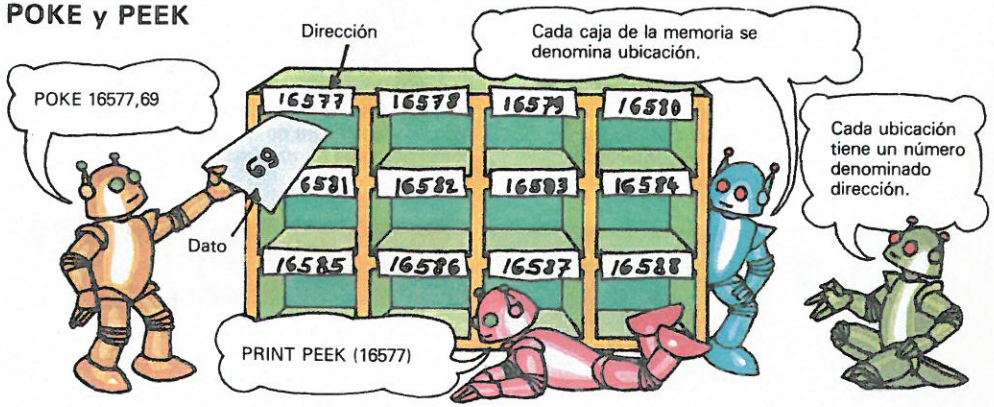
En el programa que necesita los datos sólo tienes que usar la segunda rutina que se muestra a la izquierda. Esta reserva espacio para los datos con una variable llena de estrellas, colapsa la variable y, a continuación, carga los datos desde la cinta o disco.

Tendrás que grabar P$ fuera de la computadora ya que ésta piensa que P$ está vacía. Cuando listes el programa, los datos no aparecerán listados y cada vez que ejecutes el programa los datos de la memoria serán borrados (la instrucción RUN borra todos los datos de las variables).

Otra manera de amontonar datos

Este es otro método alternativo para almacenar datos numéricos. En este caso usarás estrellas en una instrucción REM para reservar espacio para los datos. Luego sustituirás las estrellas por los datos usando la instrucción POKE*. Con este método los datos serán parte del listado del programa por lo que puedes ejecutar el programa tantas veces como quieras. Cuando grabes el programa los datos se grabarán también.

POKE y PEEK



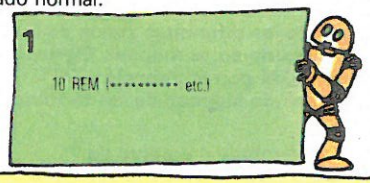
La instrucción POKE le dice a la computadora que almacene un dato en una ubicación de memoria con determinada dirección. Los datos se almacenan como números que la computadora interpreta como códigos ASCII.

PEEK hace lo contrario. Usado con PRINT le dice a la computadora que visualice el contenido de una ubicación de memoria determinada es decir, el carácter cuyo código ASCII estaba almacenado ahí.

Cómo usar POKE

Tienes que tener mucho cuidado con este método ya que si introduces datos en las ubicaciones de memoria que no debes puedes confundir a la computadora borrándole alguna información esencial. Si esto sucede tendrás que apagar y encender la computadora para que vuelva a su estado normal.

1



◀En la primera línea de tu programa necesitas una sentencia REM con tantas estrellas como números quieras almacenar. Puedes colocar las estrellas entre paréntesis para que resulten más fáciles de distinguir.

2

```
20 FOR I=A TO A+100
30 LET C=PEEK(I)
40 PRINT "DIRECCION ";I;"=";
50 IF C<31 OR C>127 THEN LET C=46
60 PRINT CHR$(C)
70 NEXT I
80 STOP
```

Sustituye la A por la primera dirección en la que tu computadora almacena programas en BASIC.

Lee (peek) cada ubicación y pon el contenido en la variable C. Si C < 31 o > 127 será un carácter no visualizable por lo que se sustituye por el 46, que es el código ASCII para el punto.

Escribe el carácter para el número de código C.

A continuación necesitarás estas líneas para que lean (peek) en la memoria de la computadora y busquen dónde están almacenadas las estrellas. Tendrás que sustituir la variable A de la línea 20 por la dirección de la primera ubicación en la que se almacenan los programas en BASIC en tu computadora. Para saber cual es esta dirección busca en el mapa de memoria del manual de tu computadora. (Para saber más sobre mapas de memoria mira en la página siguiente).

*Algunas computadoras no usan las instrucciones POKE y PEEK. Ver Guía BASIC al final de este libro.

Continuación de cómo leer datos

3

```
100 DATA lista tus números aquí
110 DATA etc
150 FOR I=X TO X+número de datos-1
160 READ N
170 POKE I,N
180 NEXT I
```

Pon el número de datos que tengas.

A continuación añade estas líneas poniendo tus números en líneas DATA comenzando en la línea 100. Después ejecuta el programa. La computadora escribirá el contenido de cada ubicación de memoria para parar en la línea 80. Cuando veas las estrellas en la pantalla, escribe la dirección de la primera estrella y asegúrate de que hay tantas estrellas como números en tus líneas DATA.

4

```
LET X=dirección de la primera estrella
GOTO 100
```

Escribe ahora esta sentencia LET como instrucción directa, es decir, sin números de línea. Y después escribe GOTO 100. Esto hará que la computadora empiece a ejecutar el programa desde la línea 100. Cada vez que se repite el bucle de las líneas 150 a 180, la computadora pone un número en N para introducirlo en la ubicación I. Ahora los números están amontonados en la memoria de la computadora.

5

```
PRINT PEEK (dirección+posición del dato-1)
```

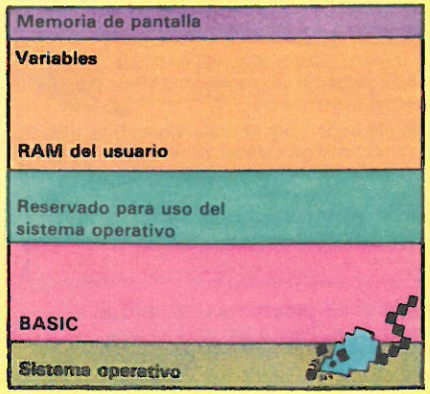
Dirección de la primera estrella.

Posición en la lista de datos del número que quieres.

Para encontrar cualquier número usa PRINT PEEK como se muestra arriba. Para comprobar que los datos están en la memoria puedes volver a ejecutar el programa. Ahora ya puedes borrar las líneas a partir de la 20 y añadir la línea 10 al programa que necesita los datos. No debes listar la línea 10 ya que la computadora puede bloquearse cuando intentes visualizar los datos en la pantalla. Puedes listar el resto del programa escribiendo LIST con un número de línea.

El mapa de memoria

El mapa de memoria es un diagrama que muestra el uso de las diferentes zonas de la memoria de la computadora. Debe haber un mapa de memoria en tu manual. El mapa muestra toda la memoria de la computadora, incluido el ROM con la dirección del principio y del final de cada zona. Aquí tienes explicaciones de algunos de los términos usados en un mapa de memoria.

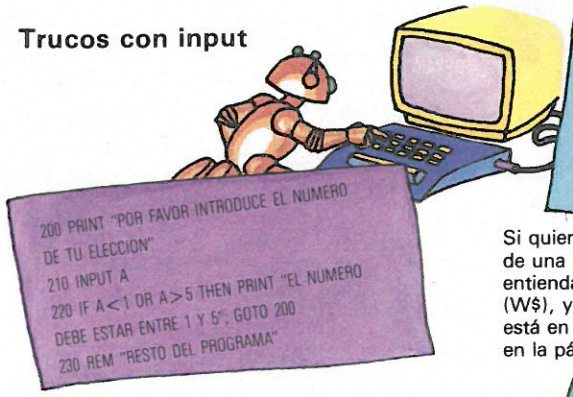


Memoria de pantalla o archivo de visualización. Aquí es donde se almacena cualquier cosa que va a ser visualizada en la pantalla.
Variables. Los datos para las variables y matrices se almacenan aquí.
Ram de usuario. Ram libre. Aquí es donde se almacenan tus programas en BASIC.
Reservado para uso del sistema operativo. La computadora usa esta zona para tener bajo control lo que está sucediendo.
BASIC. El programa ROM que permite a la computadora entender el BASIC.
Sistema operativo. El programa del ROM que le indica a la computadora cómo debe trabajar.

Control del input

Cuando usas la instrucción INPUT tienes muy poco control sobre lo que el usuario escribe. Los datos que introduce el usuario se almacenan directamene en una variable. Si introduce algo inesperado y después se usa en el programa puede causar el que el programa se pare con un mensaje de error. Siempre debes comprobar el input y asegurarte de que es válido antes de que pase a formar parte del programa.

Trucos con input



```
200 PRINT "POR FAVOR INTRODUCE EL NUMERO
DE TU ELECCION"
210 INPUT A
220 IF A<1 OR A>5 THEN PRINT "EL NUMERO
DEBE ESTAR ENTRE 1 Y 5": GOTO 200
230 REM "RESTO DEL PROGRAMA"
```

La manera más fácil de comprobar el input es mediante una prueba IF/THEN tal y como se muestra arriba. Si no pasa la prueba manda a la computadora de vuelta a por un nuevo input.

```
20 IF A$<"A" OR A$>"F" THEN PRINT
"LA LETRA DEBE ESTAR ENTRE A Y F"
```

Puedes comprobar las letras de la misma forma que los números, como se muestra arriba. En realidad la computadora compara los códigos ASCII para las letras.

```
200 PRINT "¿CUAL ES TU INSTRUCCION?"
210 INPUT A$:LET X=0
220 FOR J=1 TO 6
230 IF A$=W$(J) THEN LET X=1
240 NEXT J
250 IF X=0 THEN PRINT "INSTRUCCION NO
RECONOCIDA" :GOTO 200
```

Si quieres que el usuario escriba una palabra de una lista de ellas que la computadora entienda, almacénalas todas en una matriz (W$), y usa un bucle para saber si la palabra está en la matriz (Si tienes un Spectrum mira en la página 46).

```
200 PRINT "¿EN QUE MES ES TU CUMPLEAÑOS?"
210 PRINT "POR FAVOR, ESCRIBE UN NUMERO
ENTRE EL 1 Y EL 12"
220 INPUT B$
230 IF VAL(B$)<1 OR VAL (B$)>12 THEN
GOTO 210
```

Suele ser aconsejable usar variables de cadena con las sentencias INPUT. Si el usuario escribe una palabra y el programa está esperando un número, el programa no se interrumpirá. VAL sirve para que la computadora lea una variable de cadena como un número.

Truco con INKEY$

```
400 PRINT "ESCRIBE EL AÑO EN QUE NACISTE"
410 LET L=4                                  Número de caracteres que esperas en la respuesta.
420 LET A$=" "                               Variable vacía para almacenar la respuesta.
430 LET I=0                                  Contador.
440 LET I$=INKEY$:IF I$=" " THEN GOTO 440    Pone la siguiente tecla apretada en I$. Se repite la línea hasta que se aprieta una tecla.
450 LET CK=ASC(I$)                           Pone el código ASCII del carácter de I$ en CK,
460 IF CK>47 AND CK<58 THEN LET I=I+1:LET
A$=A$+I$:PRINT I$,                           Se asegura de que CK es el código ASCII de un número entre 0 y 9 luego suma 1 al contador y escribe I$.
470 IF I<L THEN GOTO 440                     Vuelve a buscar otro carácter del teclado.
480 LET A=VAL(A$)                            Convierte A$ en un número y lo almacena en A.
490 IF A=0 THEN GOTO 400                     Comprueba A.
```

Cuando usas INPUT no tienes control sobre la longitud de lo que el usuario va a escribir. Si quieres comprobar o restringir la longitud, usa INKEY$ en lugar de INPUT. *INKEY$ le dice a la computadora que tome la siguiente tecla apretada y la almacene en una variable.

Esta rutina usa INKEY$ con un bucle por lo que el usuario sólo puede introducir cuatro caracteres. Otra vengaja de INKEY$ es que el usuario no tiene que apretar RETURN y no aparece una interrogación en la pantalla.

*INKEY$ no es igual en todas las computadoras. Mira en la Guía Básica y comprueba tu manual.

Consejos para la visualización en pantalla

Puedes hacer que tus programas parezcan realmente profesionales haciendo aparecer en la pantalla mensajes claros y fáciles de leer que hagan que la pantalla resulte interesante.

Visualización de mensajes

Una buena regla a seguir es la de visualizar los mensajes siempre en la misma parte de la pantalla, ya sea arriba, abajo o en el centro de la pantalla. Sea cual sea el sitio que elijas, haz que aparezcan siempre en el mismo sitio para que la persona que ejecute el programa sepa dónde debe mirar. Puedes hacer esto con una subrutina para visualizar mensajes como los que se muestran aquí abajo.

```
80 CLS
90 GOSUB 1000:REM INICIALIZACION
100 LET M$="HOLA"
110 GOSUB 490
120 END
490 REM RUTINA PARA VISUALIZAR MENSAJES
500 PRINT TAB(X,Y);CL$
510 PRINT TAB(X,Y);M$
520 RETURN

1000 LET CL$=" "
1010 LET X=columna en la que visualizar el mensaje
1020 LET Y=línea en la que visualizar el mensaje
1030 RETURN
```

Mensaje

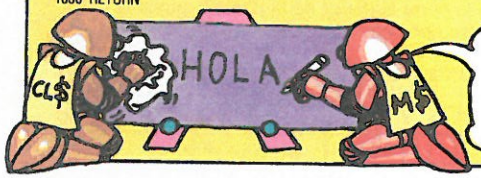
Escribe una línea de espacios en blanco para borrar antiguos mensajes.

Visualiza el mensaje en columna X, línea Y.

Hace que CL$ valga tantos espacios como columnas hay en tu pantalla.

Por ejemplo, para visualizar en la esquina superior izquierda haz que X=1 e Y=1.

En algunas computadoras puedes usar la instrucción SPC(N) para escribir N espacios en blanco en lugar de CL$.



PRINT TAB (X,Y) le dice a la computadora que visualice el mensaje en columna X, línea Y. Esto no es igual en todas las computadoras así es que revisa en tu manual y en las conversiones de la parte final de este libro.

Para usar esta rutina de visualización de mensajes tienes que poner los números de columna y fila en la que quieres que aparezcan tus mensajes, en las variables X e Y en la parte de inicialización del programa. También necesitas una variable CL$ con tantos espacios como ancha sea tu pantalla, para que borre antiguos mensajes. Cada vez que quieras visualizar un mensaje ponlo en M$ y manda a la computadora a la subrutina de visualización de mensajes.

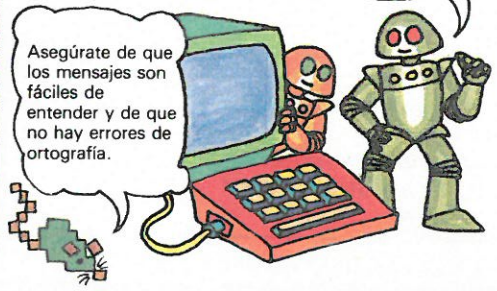
Centrar mensajes

```
497 LET L=LEN(M$)
498 LET X=INT((W-L)/2)
1010 LET W=ancho de la pantalla
```

Para centrar mensajes tendrás que añadir las líneas 497 y 498 y sustituir la 1010. La línea 497 calcula el número de caracteres en M$. La línea 498 resta este número del número de columnas en la pantalla y divide el resultado entre dos. INT redondea el número hacia abajo si la solución no es un número entero.

Procura no dejar al usuario mirando una pantalla vacía. Es aburrido y puede que piense que el programa no funciona. Para evitar esto no borres la pantalla hasta que estés listo para que aparezca el siguiente mensaje.

Asegúrate de que los mensajes son fáciles de entender y de que no hay errores de ortografía.



Visualizar grandes cantidades de información

Si tienes gran cantidad de información que dar al usuario piensa en la pantalla como en un trozo de papel y distribuye la información en ella con títulos y margen. Usa títulos y párrafos que hagan más fácil la lectura. Cuando la pantalla esté llena no permitas que automáticamente pase una línea ya que esto arruinará tu visualización. Calcula el número de líneas que te caben por pantalla y una vez que está llena bórrala y vuelve a empezar repitiendo si quieres el título y el margen (esto es muy fácil si las tienes en una subrutina).

Subrutina para el margen*

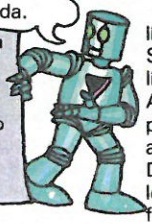
```
10 REM LINEA HORIZONTAL
20 LET L$="----------etc"
30 PRINT TAB(X,Y);L$
```

Podrías usar esta subrutina para subrayar un título. Crea una variable con tantos caracteres como los que haya en tu título. Da, a continuación, valor a las variables X e Y en la instrucción PRINT TAB para que aparezca la línea donde tú quieras.

```
10 REM LINEA VERTICAL
20 LET V$="*"+CHR$(CD)+CHR$ (CL)
30 LET L$=" "
40 FOR I=1 TO longitud de la línea
50 LET L$=L$+V$
60 NEXT I
70 PRINT TAB (X,Y);L$
```

Después de dibujar cada estrella, los códigos del cursor hacen que la computadora se mueva una posición hacia abajo a la izquierda listo para dibujar la siguiente estrella.

Sustituye CD y LC por los códigos ASCII de tu computadora para subir el cursor y para moverlo hacia la izquierda.



Esta rutina dibuja una línea en la pantalla. Sustituye CD y CL en la línea 20 por los códigos ASCII de tu computadora para mover el cursor hacia abajo y hacia la izquierda. Da a la X y a la Y el valor de los números de columna y fila en las que quieres que empiece la línea.

```
90 REM RUTINA VENTANA
100 CLS
110 GOSUB 400
120 GOSUB 250
130 GOSUB 200
140 LET Y=Y+I:GOSUB 250
150 END
195 REM VERTICALES
200 FOR I=1 TO H-2
210 PRINT TAB(X,Y+I);L$
220 PRINT TAB(X+W-1,Y+I);L$;
230 NEXT I
240 RETURN
245 REM HORIZONTALES
250 FOR J=0 TO W-1
260 PRINT TAB(X+J,Y);L$;
270 NEXT J
280 RETURN
395 REM INICIALIZAR VARIABLES
400 LET L$="*"
410 LET H=altura de la ventana
420 LET W=ancho de la ventana
430 LET X=posición del lado izquierdo de la ventana
440 LET Y=posición de la parte superior de la ventana
450 RETURN
```

Lado izquierdo

Lado derecho

U otro carácter

Utiliza las instrucciones de color de tu computadora para trazar líneas de colores.

Esta es una rutina para dibujar margen en cualquier lado de la pantalla. Tendrás que introducir las variables para el tamaño de ventana que desees. Por ejemplo, para una ventana por el borde de la pantalla haz que X e Y valgan ambas 1, que H sea el número de líneas de tu pantalla menos 1 y que W sea el número de columnas menos 1.

Centrar el texto en la ventana

Añade estas líneas para centrar el texto en la ventana dibujado por la rutina de la izquierda.

```
145 GOSUB 300
295 REM CENTRAR MENSAJE
300 LET C=INT ((W-LEN(M$))/2)
310 PRINT TAB(X+C,Y-INT(H/2));M$
320 RETURN
405 LET M$=mensaje
```


*Mira en la página 47 si necesitas ayuda para convertir las instrucciones PRINT TAB de estas rutinas.

Preparación de menús

Un menú es una forma de permitir al usuario escoger qué parte del programa desea ejecutar. El menú consiste en una lista de opciones de las cuales el usuario puede elegir la que desee. Una vez elegida la computadora irá a la parte del programa que lleve a cabo la función por la que optó el usuario. El menú tiene que ser claro y fácil de entender, además de resultar interesante. En estas dos páginas tienes algunas ideas para presentar menús.

```
CREAR FIGURAS
Desearías

1) Dibujar una figura
2) Modificar una figura
3) Usar la biblioteca de figuras
4) Grabar una figura

Por favor, introduce el número de tu elección
Luego aprieta RETURN
```

```
JUEGO DE ESPIAS
¿Qué desearías hacer?

>Jugar una nueva partida
 Ver pistas
 Estudiar las claves

Aprieta A o Z para mover el cursor hacia la
opción que prefieres

Luego aprieta RETURN
```

Asegúrate de que las instrucciones de tu menú son claras.

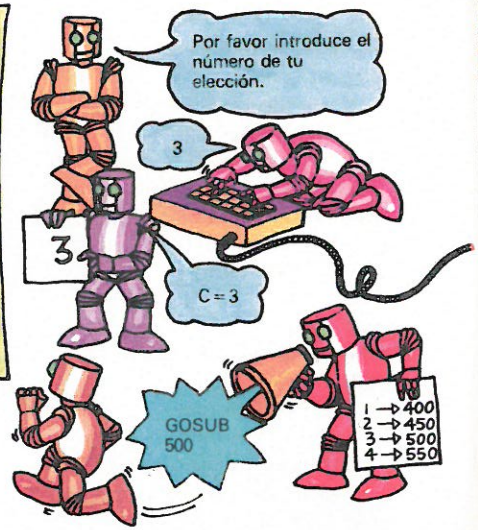
Cuando estás planeando un menú, la principal decisión a realizar es cómo vas a hacer que el usuario indique su preferencia. Lo más sencillo es escribir un número, como en el ejemplo de la izquierda. Otra forma es mover el cursor o cualquier símbolo hacia la opción. En la página opuesta tienes una rutina para mover el cursor en el menú.

Decir a la computadora dónde debe ir

Para decir a la computadora qué parte del programa ejecutar tendrás que almacenar la elección del usuario en una variable. Puedes hacer esto con un INPUT en los menús de elección numérica. Para los menús de movimiento del cursor tendrás que calcular la posición de éste tal y como se indica en la página opuesta. Luego puedes usar la instrucción ON GOSUB* como se muestra abajo, para indicar a la computadora a qué subrutina ir.

```
200 PRINT "CREAR FIGURAS"
210 PRINT "Desearías:"
220 PRINT "1) Dibujar una figura"
230 PRINT "2) Modificar una figura"
240 PRINT "3) Usar la biblioteca de figuras"
250 PRINT "4) Grabar una figura"
260 PRINT "Por favor introduce",
270 PRINT "el número de tu elección"
280 PRINT "Luego aprieta RETURN"
290 INPUT C
295 IF C<1 OR C>4 THEN GOTO 260
300 ON C GOSUB 400, 450, 500, 590,
```

Menú — Elección del usuario — Comprobación del input



ON GOSUB se usa como se muestra en la rutina de arriba con el valor de una variable. El valor de la variable le dice a la computadora qué subrutina debe llevar a cabo. Por ejemplo, si C es 3 la computadora irá a la tercera subrutina listada en la línea 300.

*No todas las computadoras tienen la instrucción ON GOSUB - ver Guía BASIC.

Entrada de palabras

```
250 PRINT "¿QUE COLOR QUIERES?"
260 PRINT "ROJO, NEGRO, VERDE, AZUL"
270 PRINT "ESCRIBE TU COLOR",
280 PRINT "Y DESPUES APRIETA RETURN"
290 INPUT A$
300 LET C=0
310 FOR J=1 TO 4
320 IF W$(J)=A$ THEN LET C=J
330 NEXT J
340 IF C=0 THEN PRINT "PALABRA NO
RECONOCIDA, VUELVE A INTENTARLA" GOTO 270
350 ON C GOSUB etc
```

Bucle para buscar en una matriz que contiene palabras.

Si quieres que el usuario elija introduciendo una palabra, tendrás que almacenar todas las palabras entre las que puede elegir en una matriz (W$). Luego compara las palabras que el usuario introduzca con las de la matriz. Cuando alguna coincida, almacena el subíndice de la palabra matricial en una variable (C) para usar con ON GOSUB*.

Entrada de letras

```
200 PRINT "POR FAVOR INTRODUCE UNA LETRA
DE LA A A LA F"
210 PRINT "Y DESPUES APRIETA RETURN"
220 INPUT A$
230 LET A=ASC(A$)
240 IF A<65 OR A>70
TEHN GOTO 200
250 LET C=A-64
260 ON C GOSUB etc.
```

Convierte la letra en su código ASCII.

Comprueba que los códigos están entre los correspondientes a las letras A-F.

Para elegir introduciendo una letra tendrás que convertirla en un número que puedas usar con ON GOSUB. Para ello convierte la letra en su código ASCII usando la instrucción ASC (o CODE en el Spectrum). Luego resta el número necesario para obtener una secuencia de números que comience en el 1. Por ejemplo, los códigos de las letras A a la F empiezan en 65 por lo que tienes que restar 64 como se ve en la línea 250 del ejemplo superior.

Menú de movimiento del cursor

Para hacer esta subrutina tienes que calcular la posición TAB de cada línea del menú. Luego calcula los números TAB para situar el cursor a la izquierda de la primera opción del menú. También tendrás que saber el número de líneas desde la parte superior de la pantalla hasta la primera opción del menú, y el número de líneas entre las opciones.**

```
270 CLS
280 GOSUB 1000:REM INICIALIZAR
290 PRINT TAB(1);"JUEGO DE ESPIAS"
300 PRINT TAB(1);"¿QUE DESEARIAS HACER?"
310 PRINT TAB(1,NL); "JUGAR UNA NUEVA PARTIDA"
320 PRINT:PRINT TAB(1);"VER PISTAS"
330 PRINT:PRINT TAB(1);"ESTUDIAR LAS CLAVES"
340 PRINT
350 PRINT TAB(1);"APRIETE A O Z PARA MOVER EL"
360 PRINT TAB(1);"CURSOR HACIA LA OPCION QUE PREFIERAS"
370 PRINT:PRINT TAB(1);"LUEGO APRIETA RETURN"
380 PRINT TAB(CX,CY),">":LET OY=CY
390 LET I$=INKEY$:IF I$=" " THEN GOTO 390
400 IF I$="A" AND CY>NL THEN LET CY=CY-SP
410 IF I$="Z" AND CY<LC THEN LET CY=CY+SP
420 IF CY<>OY THEN PRINT TAB (CX,OY);" "
430 IF I$<>CHR$(13) THEN GOTO 380
440 LET C=INT( (CY-NL)/SP)+1
450 ON C GOSUB 500,600,700
495 END
1000 LET NL=4:LET NC=3:LET SP=2:LET CX=0
1010 LET CY=NL:LET LC=NL-SP+(NC*SP)
1020 RETURN
```

Rutina para mover el cursor.

Tecla RETURN = código ASCII 13.

Calcula un valor de C para usar con ON GOSUB

El *Archivo-Clasificador* usa un menú como este. Ver líneas 885-980.

CX = posición X del cursor (columna).
CY = posición Y del cursor (línea)
NL = número de líneas desde lo alto de la pantalla hasta la primera opción del menú.
SP = número de líneas entre dos opciones.
NC = número de opciones.
LC = línea de la última opción.

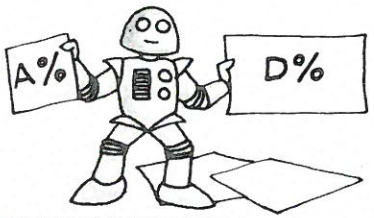
*Si tienes un Spectrum mira en la página 47.
**Mira en la página 47 si necesitas ayuda para convertir esta rutina.

Acelerar programas

Debajo tienes algunos trucos para hacer que los programas vayan más rápidos. En la página opuesta tienes ideas para reducir o «comprimir» un programa para que ocupe menos memoria. Cuanto más corto sea un programa más rápido se ejecutará, por lo que si te resulta importante la velocidad puedes también usar alguno de estos trucos para ahorrar espacio.

Muchos de estos trucos desharán lo que has hecho en los programas para hacerlos más fáciles de leer y entender, por lo que siempre es bueno guardar una copia del programa antes de comprimirlo. Puedes usar la copia si en la versión comprimida encuentras errores, o si quieres desarrollar el programa más adelante. Aquí tienes algunos trucos para acelerar la ejecución de un programa.

1



Si tu computadora tiene variables enteras, es decir variables, en las que sólo pueden almacenarse números enteros, úsalos siempre que te sea posible. Las variables enteras tienen un signo % detrás de su nombre (ver página 13). Ocupan mucha menos memoria que las variables numéricas normales y la computadora las maneja más rapidamente.

2

Bucle normal

```
FOR J=1 TO 500
PRINT J
NEXT J
```

Bucle entero

```
FOR J%=1 TO 500
PRINT J%
NEXT J%
```

Usa siempre variables enteras (si las tienes) para los contadores de los bucles como se describe en la página 13. Van al doble de velocidad que los de variables normales.

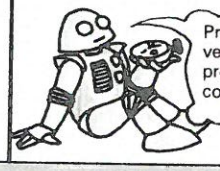
Con un cronómetro calcula en tu computadora la diferencia entre un bucle repetido 500 veces con una variable entera y otro igual con una variable normal.

3

En un programa con muchas subrutinas distribuye las subrutinas según su frecuencia de uso. Pon las más usadas justo detrás del programa principal para que la computadora las encuentre con facilidad. Si existe una función para la que la velocidad sea muy importante colócala justo al principio del programa como se describe en la página 10.

4

Acorta el programa borrando las sentencias REM y poniendo varias instrucciones en cada línea como se explica en la página opuesta.



Prueba a cronometrar dos versiones de un mismo programa para ver cuánto has conseguido acelerarlo.

Velocidad de la computadora

La velocidad de un programa también depende de la velocidad de tu computadora. La velocidad de una computadora se mide en megahertzios (MHz) siendo más rápida una computadora que trabaja a 2 MHz que una que trabaje a 1 MHz.

La velocidad a la que ejecutes programas también depende de cuántas operaciones tienen que realizar para cumplir una función. Esto está controlado por el software de la computadora, es decir, el sistema operativo que le dice cual hacer, y el intérprete que traduce el BASIC a código de máquina.

Trucos para ahorrar espacio

Los trucos que hay a continuación y en la página siguiente ahorran gran cantidad de memoria, aunque hacen los programas más difíciles de leer y corregir. Es mejor no usarlos hasta que hayas terminado el programa y funcione perfectamente sin errores. Graba una copia del programa en cinta o disco, y si es posible cópialo en papel para poder trabajar con él.

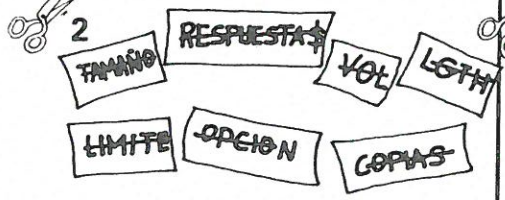
Puedes usar estos trucos si quieres por ejemplo añadir más datos a un programa y no tienes suficiente memoria o si quieres mejorar los gráficos o simplemente si quieres que vaya más rápido.

1

```
IF A=C THEN PRINT "SI"
IF X=Y THEN GOTO 270
IF X=Y THEN GOTO 270
LET T=3
IF R>100 THEN LET R=R-1
```

En muchas computadoras puedes suprimir algunas palabras de las instrucciones, como LET, THEN y GOTO. Comprueba en tu manual ya que esto varía en cada computadora y después repasa el programa suprimiendo estas palabras.

2



Haz más cortas todas las variables, hasta que tengan una o dos letras. Debes tener mucho cuidado al hacer esto ya que si se te olvida modificar la variable en alguna línea obtendrás un error. Ten también cuidado de que todos los nuevos nombres de variables sean diferentes entre sí. Este es el motivo por el que al escribir un programa conviene que todas las variables comiencen con letras diferentes.

3

```
395 REM SUBRUTINA MENSAJE
400 PRINT TAB(X,Y);M$
```

Borra todas las sentencias REM. Estas servían sólo para hacer que el programa fuese más fácil de leer y por tanto no son esenciales para la ejecución del programa. Los REM que estén en líneas por sí solos son fáciles de borrar suprimiendo la línea. No obstante debes tener cuidado de que ninguna instrucción GOTO o GOSUB mande a la computadora a una línea que has borrado.

4

```
701FX>THENPRINT;X-1;"PAGINAS"
40FORI=N TOP
150 ON C GOSUB200,300,400
```

Suprime el espacio de detrás del número de línea.

Si quitas este espacio la computadora puede leer esto como variable NTOP.

Necesitas estos espacios.

A continuación suprime cualquier espacio innecesario. Cada espacio ocupa un byte de memoria, por lo que puedes ahorrar mucho si los suprimes. Sin embargo, debes tener cuidado ya que algunas sentencias necesitan espacios para resultar claras.

Más trucos en la página siguiente

Toolkit del programador

Se denomina Toolkit del programador a un programa especial que puedes comprar para que haga todas estas cosas. Una vez cargado el programa en la memoria de la computaora, te ofrece entre borrar palabras determinadas como REM o LET, renumerar el programa y buscar y cambiar variables. Existen diferentes variedades que te ofrecen gran cantidad de facilidades. Este es un ejemplo de las opciones que presentaría un buen programa.

```
Toolkit
Por favor aprieta el número que deseas
1) Renumerar el programa
2) Buscar y modificar
3) Merge (mezclar programas)
4) Listar variables
5) Cambiar líneas
6) Comprimir el programa
7) Borrar REMs
```

Más trucos para ahorrar espacio

5

```
200 FOR J=1 TO 5
210 PRINT J
220 NEXT
```

En muchas computadoras puedes olvidar la variable del bucle. La computadora enseguida supone que significa NEXT J.

6

```
200PRINT"UN MOMENTO":FORI=1TO99:
READA$(II):NEXT:PRINT:PRINT"ACABADO".
GOSUB500
```

Después de una subrutina la computadora vuelve a la siguiente sentencia de la línea.

```
90LETL$=CHR$(42):GOSUB100:PRINTTAB
(X,Y);M$:LETL$=CHR$(46):GOSUB100
```

Combina tantas líneas como puedas en líneas multi-sentenciales. No juntes IF THEN o ON GOSUB con otras sentencias que no dependan de ellos.

Comprueba cuál es el máximo de caracteres por línea que admite tu computadora. Suelen ser 255 caracteres (eso equivale a seis líneas aproximadamente en una computadora con pantalla de 40 columnas) Asegúrate de que las subrutinas todavía comienzan en las líneas indicadas con las instrucciones GOSUB.

7

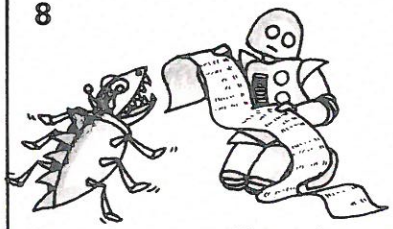
```
50 PRINT M$:GOBUB 100
```

```
50P.M$:GOS.100
```

Comprueba en tu manual si puedes usar abreviaturas.

En algunas computadoras se permiten formas abreviadas para palabras en BASIC; por ejemplo, P por PRINT. El uso de éstas permite meter más sentencias en una línea de programa. Sin embargo, por sí solas no ahorran espacio de memoria ya que la computadora siempre almacena la versión reducida.

8



Tras todos estos cambios puede que tengas que renumerar el programa. Comprueba que todas las subrutinas y GOTOs van al sitio adecuado.

También puedes ahorrar espacio de memoria usando los trucos con datos de las páginas 20-26.

Más sobre la memoria

Casi todas las computadoras personales tienen entre 32K y 48K RAM, lo que suele ser suficiente para programas de gran longitud. No obstante parte del RAM puede estar ocupado por funciones internas de la computadora, es decir, para almacenar la información que necesita mientras ejecuta el programa. Esto puede ocupar unos 3K que por tanto reducen el espacio para el programa.

Muchas computadoras necesitan además mucho espacio RAM al usar gráficos de alta resolución, cuando deben almacenar información para gráficos. En la resolución más alta esto puede ser de hasta 20K, lo que deja sólo unos 10K para el programa.

La mayor memoria que una computadora de ocho-bit (la que usa bytes compuestos de ocho bits o señales) puede tener es 64 ROM y RAM combinados. Esto se debe a que cada ubicación de memoria ha de tener un número como dirección. Cada dirección está representada por dos bytes de códigos máquina, y el máximo número que puede escribirse con dos bytes (16 bits) es 65535. Esto permite 65536 ubicaciones de memoria numeradas del 0-65535 lo que equivale a 64K (65536 ÷ 1024).

Sin embargo es posible usar más memoria en una computadora de ocho bit, conectando diferentes bloques de memoria. Puedes comprar unidades de expansión de memoria para hacer esto.

Programa *Archivo-Clasificador*

En las páginas siguientes hay un listado del programa *Archivo-Clasificador*. El programa sirve como ilustración de muchas de las técnicas de programación descritas en este libro. Cada función la lleva a cabo una parte diferente del programa por lo que resulta muy fácil estudiar cada parte por separado. El programa está escrito en una versión standard de BASIC aunque también se ofrecen conversiones habituales al final del listado.

Cómo usar el *Archivo-Clasificador*

El *Archivo-Clasificador* es más o menos un archivo de fichas. El programa está destinado a almacenar nombres, direcciones y teléfonos, pero también puedes modificarlo para que almacene cualquier otra cosa. Al final del programa tienes algunos consejos para hacer esto.

```
FLIP- -FILE
NO CARDS IN FILE
                WOULD YOU LIKE TO:
*****************************************
>ADD A CARD
 REMOVE A CARD
 ALTER A CARD
 FLIP CARDS
 LOAD CARDS
 FILE THESE CARDS

*****************************************
USE KEYS ( ) TO MOVE CURSOR UP & DOWN
PRESS RETURN TO CHOOSE AN OPTION
```

```
FLIP- -FILE
1 CARD ON CARD 1
          ADD A CARD
*****************************************
  NAME     COMPUTER CLUB
>STREET    75*
  TOWN
  COUNTY
  CODE
  PHONE
  INFO
*****************************************
USE KEYS ( ) TO MOVE CURSOR UP & DOWN
PRESS RETURN FOR MENU
```

Cuando ejecutas el programa este menú aparece en la pantalla. Para elegir mueves el cursor a la línea que deseas y después aprietas RETURN o ENTER.

> Buscar en las fichas

Cuando eliges la opción Buscar en las fichas, la computadora te visualiza cada ficha para que puedas encontrar la que deseas. También puedes decir a la computadora que busque una línea determinada de una ficha. Para hacer esto elige la opción Buscar en las fichas y aprieta la tecla @ . Aparecerá en la pantalla una ficha en blanco. Escribe el dato que quieres buscar en la línea correspondiente.

Por ejemplo, si quieres encontrar todas las fichas de las personas que viven en una determinada ciudad, escribe el nombre de la ciudad en el apartado correspondiente a la ciudad. A continuación aprieta RETURN. La computadora recorrerá las fichas hasta que encuentre lo que quieres. Debes escribir el dato que buscas exactamente como lo escribiste en la ficha ya que si no la computadora no será capaz de encontrarlo.

Puedes continuar buscando apretando cualquier tecla o volver al menú apretando RETURN.

Para escribir información en una ficha tienes que mover el cursor a la línea que desees. La ★ señala dónde va a aparecer el siguiente caracter.

> Hacer cambios en una ficha

Para hacer cambios, elige la opción de Buscar fichas para encontrar la ficha que quieres. Luego vuelve al menú y selecciona Hacer cambios en una ficha. La ficha aparecerá en la pantalla y podrás usar las teclas específicas de tu computadora para borrar palabras.

> Suprimir una ficha

Para suprimir una ficha aprieta la tecla @ . El mensaje de borrar aparecerá en el apartado denominado INFO de la ficha en cuestión. La ficha desaparecerá cuando almacenes todas las fichas.

> Cargar fichas > Archivar estas fichas

Puedes cargar o grabar fichas a y desde cinta o disco. Las notas sobre conversiones te enseñan cómo hacer los cambios en el programa para hacer esto con las marcas principales de computadoras. Si quieres grabar las fichas en disco, cada vez que grabes un nuevo grupo de fichas, se borrará el antiguo.

Variables del *Archivo-Clasificador*

Debajo tienes una lista de las variables usadas en el programa. Te puede ayudar el acudir a esta lista mientras estudias el programa.

Para ahorrar espacio de memoria todas las fichas están almacenadas en una matriz de una dimensión, como se explica en la página 20. La matriz se denomina C$ y cada registro (ver página 20) contiene toda la información para una ficha. El primer registro está en blanco para ser usado en la rutina para buscar (ver página anterior).

Para encontrar información para cada ficha, el programa usa la siguiente fórmula «número de ficha × longitud de ficha + 1». Esta es la misma fórmula que vimos en la página 20, salvo que no es necesario restar 1 del número de ficha ya que los datos de las fichas empiezan en el registro 2 (el registro 1 es la ficha en blanco para la rutina de buscar).

C$	Matriz de fichas-matriz de una dimensión que contiene los datos de las fichas.
MX	Máximo número de fichas permitidas.
M$	Datos para el menú.
ML	Longitud del menú, es decir, número de opciones.
H$	Títulos de las fichas.
CL	Longitud de las fichas-número de líneas por ficha.
FICHA	Número actual de ficha.
ULTIMA	Número de la última ficha en el archivo.
SC	Ficha para buscar. Es el primer registro en C$ y se usa como ficha en blanco en la que el usuario escribe el dato que busca.
NC	Número de fichas que van a ser cargadas o almacenadas en o desde el archivo de cinta o disco.
ND	Número de fichas borradas.
KP	Código ASCII para tecla apretada.
OPCION	Opción del menú elegido por el usuario.
IL	Longitud de entrada-máximo número de caracteres que se pueden introducir.
B$	Buffer. Variable temporal para contener el input.
S	Espacios. Si S = 2 hay un espacio entre cada línea de la ficha.

L	Para calcular la línea en la que aparece el cursor.
OL	Antigua línea-posición anterior del cursor.
LMAX	Posición más baja del cursor, es decir, parte inferior del menú o ficha.
DP	Margen. Número de líneas desde lo alto de la ficha hasta la primera línea con mensaje.
TP	Parte superior de la ficha, es decir, primera línea del menú o título de la ficha.
DENT	Posición de la columna de los datos de la ficha.
W	Ancho de la pantalla en columnas.
W$	Variable que contiene el mensaje que le dice al usuario qué teclas apretar.
X$	Variable para mensajes en la parte superior de la pantalla.
Y$ Z$	Variables para mensajes en la parte inferior de la pantalla.
L$	Línea de estrellas.
D$	Borra mensaje escrito en ficha que va a ser suprimida.
ARRIBA ABAJO	Códigos ASCII para mover el cursor hacia arriba y hacia abajo.
OTRA	Código ASCII de la tecla para suprimir ficha.
RETRO	Código ASCII de la tecla para retroceder un espacio y borrar el último carácter escrito.

Listado del *Archivo-Clasificador*

Este es el listado del *Archivo-Clasificador*. Las líneas de conversión para las computadoras nombradas abajo están en las páginas 41-42. Si tienes otra computadora podrás adaptar el programa estudiando tu manual y las conversiones.

Las líneas que han de ser cambiadas en todas las computadoras se señalan con ●. Aquéllas. que sólo han de cambiarse para algunas computadoras están señaladas con la inicial de la correspondiente computadora. Las líneas señaladas con ★ tienen una instrucción PRINT TAB (X, Y) y han de ser cambiadas para todas las computadoras menos la BBC y Electron.

C: C64; **V**: VIC-20; **A**: Apple; **T**: TRS-80 Colour Computer con BASIC ampliado; **B**: BBC; **E**: Electron; **S**: Spectrum.

Programa principal

```
T    6 REM Ver conversiones TRS-80
     10 GOSUB1260:REM INICIALIZACION
     20 LET X$="DESEAS:"
     30 LET LMAX=ML:LET L=OPCION
     40 GOSUB1170
TA★  50 PRINTTAB(0,TP);
     60 FOR I=1 TO ML
A    70 PRINTTAB(1);M$(I)
     80 IF S=2 THEN PRINT
     90 NEXT I
     100 LET Y$=W$+"PARA MOVER EL CURSOR HACIA ARRIBA Y HACIA ABAJO"
TS   110 LET Z$="APRIETA RETURN PARA ELEGIR UNA OPCION"
     120 GOSUB1140
     130 GOSUB890:IF KP<>13 THEN GOTO 130
     140 LET OPCION=L
     150 IF OPCION=1 AND ULTIMO=MX-1 THEN LET X$="ARCHIVO
     LLENO":GOTO 30
     160 IF ULTIMO=0 AND OPCION 1 AND OPCION<>5 THEN
     LET X$="NO HAY FICHAS":GOTO 30
     170 LET X$=M$ OPCION:LET LMAX=CL
TS   180 LET Z$="APRIETA RETURN PARA VOLVER AL MENU":LET L=1
S    190 ON OPCION GOSUB 230,290,350,390,690,690
     200 IF OPCION<6 THEN GOTO20
CVA  210 CLS:PRINT"FICHAS GRABADAS. YA NO ESTAS EN EL
     ARCHIVO-CLASIFICADOR."
     220 STOP
```

Subrutina principal

```
     225 REM AÑADIR UNA FICHA AL ARCHIVO
     230 LET FICHA=ULTIMA+1
     240 LET ULTIMA=ULTIMA+1
     250 GOSUB810:GOSUB1140
     260 GOSUB990
     270 IF KP<>13 THEN GOTO 250
     280 RETURN
```

20: Pone el mensaje en X$.

30: Crea la variable que indica la línea en la que aparecerá el cursor.

40: Visualiza el título y el mensaje superior.

50: Sitúa la posición para escribir en la parte alta de la ficha.

60-90: Visualiza el menú. Si S = 2 deja un espacio entre cada línea.

100-110: Crea los mensajes de la parte inferior de la pantalla.

120: Visualiza los mensajes en la parte inferior de la pantalla.

130: Subrutina para elegir moviendo el cursor. Si OPCION <>13 (Código ASCII para la tecla RETURN) se repite la subrutina.

140: Da a OPCION la posición de línea actual del cursor.

150-160: Si el usuario opta por añadir fichas y el fichero está lleno o no hay ninguna ficha y el usuario no ha elegido la opción 5 (cargar fichas) se escribe el mensaje y se repite el menú.

170: Crea el mensaje X$ para escribir la opción del menú. Da a LMAX la longitud de la ficha.

180: Pone el mensaje en Z$ y hace que L ponga el cursor en la línea 1.

190: Va a la subrutina correspondiente a la elección del menú.

200: Vuelve aquí después de cada subrutina y vuelve a visualizar el menú si OPCION<6 (Archivar fichas).

210-220: Para el programa después de archivar las fichas.

230: Calcula el número de ficha con el número de la última ficha archivada.

240: Suma uno al número de fichas en el archivo.

250: Visualiza el contenido de la ficha y ofrece los mensajes en la parte inferior.

260: Rutina input.

270: Repite la rutina input hasta que KP (tecla apretada) = 13. (Código ASCII de RETURN).

```
285 REM BORRAR UNA FICHA
290 GOSUB810
300 LET Y$="APRIETA "+CHR$ (PARA)+" OTRA":LET Z$=X$
310 GOSUB1140
320 GOSUB890
330 IF KP=PARA THEN LET C$(FICHA*CL+CL)=D$:LET
ND=ND+1:GOTO 290
340 RETURN

345 REM HACER CAMBIOS EN UNA FICHA
350 GOSUB810:GOSUB1140
360 GOSUB990
370 IF KP<>13 THEN GOTO 360
380 RETURN

385 REM PASAR LAS FICHAS MANUALMENTE
390 LET FICHA=1
400 LET Y$=W$+" PARA PASAR, "+CHR$ (PARA)
+" PARA BUSCAR":GOSUB810

410 GOSUB1140:LET L=1:GOSUB890
420 IF KP=ARRIBA AND FICHA<LAST THEN LET FICHA=FICHA+1
430 IF KP=ABAJO AND FICHA>1 THEN LET FICHA=FICHA-1
440 IF KP=PARA THEN GOSUB470
450 IF KP <>13 THEN GOTO 400
460 RETURN

465 REM SUBRUTINA BUSCAR

470 LET CT=FICHA:LET FICHA=SC

480 FOR I=1 TO CL:LET C$(SC+I)=" ":NEXT I

490 LET X$="PASAR-BUSCAR":LET Y$="MUEVETE UNA LINEA
Y ESCRIBE EL TEMA"

T S 500 LET Z$="APRIETA RETURN PARA EMPEZAR A BUSCAR"

510 GOSUB810:GOSUB1140

520 LET KP=0:GOSUB990

530 IF C$(SC+L)=" " THEN LET FICHA=CT:RETURN

540 LET Y$="BUSCANDO: "+C$(SC+L):LET
Z$="APRIETA CUALQUIER TECLA PARA CONTINUAR"

550 LET KP=0: LET ENCONTRADO=0

560 FOR I=1 TO ULTIMA

570 LET FLAG=0



580 IF C$(I*CL+L)=C$(SC+L) THEN LET CT=I: LET FLAG=1



590 LET FICHA=I
```

Subrutinas principales

290: Visualiza el contenido de la ficha.

300: Mensajes para la parte inferior de la ficha.

310: Visualiza mensajes en la parte inferior de la ficha.

320: Mueve el cursor y lee el código ASCII de la tecla apretada.

330: Si el código ASCII de la letra apretada corresponde con la tecla para quitar una página, hace que el último dato de la ficha sea = D$.

350: Visualiza el contenido de la ficha y los mensajes inferiores.

360: rutina input.

370: Repite la rutina input hasta que el usuario apriete RETURN.

400: Crea mensajes. Visualiza el contenido de la ficha.

410: Visualiza mensajes. Pone el cursor en la línea 1 y va a la rutina de movimiento del cursor.

420-430: Ajusta el número de ficha según la tecla apretada.

440: Va a la 470 si el usuario elige la subrutina buscar.

450: Si la tecla apretada <>RETURN repite la rutina pasar.

470: Almacena el número de ficha en la variable temporal CT. Asigna al número de ficha el valor SC, es decir, la ficha 0 que ocupa el registro 1 de la matriz C$.

480: Borra la ficha de búsqueda.

490-500: Mensajes.

510: Visualiza el contenido de la ficha y escribe mensajes.

520: Borra KP y usa la rutina input.

530: Abandona la rutina buscar si el usuario no escribe nada.

540: Escribe el mensaje de búsqueda. C$ (SC + L) = dato que busca el usuario.

550: Borra las variables. LOCALIZADO = Variable que contiene el número de veces que se ha encontrado el dato.

560-630: Bucle para buscar en las fichas.

570: Borra la variable FLAG que se usa para indicar cuándo se ha encontrado el dato.

580: Si línea de la ficha = línea en ficha de búsqueda, da a CT el valor del contador del bucle (es decir, el número de ficha) y hace que FLAG valga 1.

590: Da al número de ficha el valor del contador del bucle para poder usar la variable FICHA en las subrutinas 810 y 1140.


```
600 GOSUB810:GOSUB1140
610 IF FLAG=1 THEN GOSUB 960 :LET LOCALIZADO=LOCALIZADO+1
620 IF KP=13 THEN LET I=ULTIMO
630 NEXT I
640 LET FICHA=CT
650 LET X$="BUSQUEDA":GOSUB1170



660 IF LOCALIZADO>0 THEN LET Z$="LOCALIZADO:"+STR$
(LOCALIZADO)+"VECES"

670 IF LOCALIZADO=0 THEN LET Z$="NO ENCONTRADO:"
+"C$(SC+L)

680 RETURN

685 REM CARGAR Y GRABAR FICHAS

  • 690 IF OPCION=5 THEN abrir archivo para
    input:input #, NC

  • 700 IF OPCION=6 THEN abrir archivo para
    output:print #, LAST-ND

A 710 GOSUB1170:GOSUB1140

S 720 LET J=0: IF OPCION=5 THEN LET
  J=ULTIMO:LET ULTIMO=ULTIMO+NC

S 730 LET J=J+1

S 740 FOR K=1 TO CL

  • 750 IF OPCION=5 THEN input #, C$(J*CL+K)

  • 760 IF OPCION=6 AND C$(J*CL+CL)<>D$ THEN
    print #, C$(J*CL+K)
S 770 NEXT K
S 780 IF J<MX AND J ULTIMO THEN GOTO730
  • 790 cerrar archivo
  800 RETURN

  805 REM VISUALIZAR CONTENIDO DE LA FICHA
  810 GOSUB1170
TA★ 820 PRINT TAB(0,TP);
  830 FOR K=1 TO CL
  840 LET T$=C$(FICHA*CL+K)
A 850 PRINT TAB(1);H$(K);TAB(DENT);T$
  860 IF S=2 THEN PRINT
  870 NEXT K
  880 RETURN

  885 REM MOVER EL CURSOR ARRIBA Y ABAJO
  ★ 890 PRINTTAB (0,TP+(L-1)*S);">";
```

Subrutinas principales

Subrutinas de apoyo

600: Visualizar contenido de la ficha y mensajes inferiores.

610: Si encuentra el dato va a rutina input para la siguiente instrucción.

620: Si tecla apretada = RETURN asigna el valor del contador al del valor final del bucle terminado. De igual forma que se describe en página 17.

640: Da a FICHA el valor almacenado temporalmente en CT.

650: Escribe el título de la ficha y el mensaje superior.

660-670: Comprueba LOCALIZADO y pone el mensaje en Z$. STR$ convierte una variable numérica en variable de cadena.

680: Vuelta a 450.

690: Si OPCION=5 (cargar fichas) abre un archivo llamado F$ (FLIPDAT) y carga el dato NC (el número de fichas almacenadas en el archivo).

700: Si OPCION=6 (Archivar fichas) abre un archivo llamado F$ (FLIPDAT) y graba el número de fichas menos el número de fichas borradas.

710: Visualiza mensajes superior e inferior.

720: Da al contador de bucle J el valor 0, o si es para cargar fichas el número de la última ficha. Este es el principio de un bucle repetir/hasta (ver página 16).

740: Bucle para repetir una vez por cada dato de la ficha.

750: Carga los datos en la matriz C$ usando la fórmula descrita en la página 20. J = número de registro, CL = longitud del registro y K = número de dato.

760: Si el último dato de la ficha <>D$ (borrar mensaje) graba el dato en el archivo.

780: Se repite el bucle si J es menor que el máximo de fichas permitidas y menor que el número de la última ficha.

790: Cierra archivo.

810: Escribe el título y el mensaje superior.

820: Coloca la posición para escribir en la parte superior de la ficha.

830: Bucle para repetir una vez por cada dato en la ficha.

840: Usa la fórmula descrita en la página 20 para copiar datos en la variable T$ para la siguiente línea de la ficha.

850: Escribe el título de la ficha H$ seguida de T$.

860: Si S=2, deja una línea en blanco.

890: Aparece el símbolo del cursor junto a la primera línea de la ficha o menú.

```
      900 LET OL=L
      910 GOSUB960
      920 IF KP=ARRIBA AND L>1 THEN LET L=L-1
      930 IF KP=ABAJO AND L<LMAX THEN LET L=L+1
  ★   940 IF L<>OL THEN PRINTTAB(0,TP+(OL-1)*SI;" ";
      950 RETURN

      955 REM ESPERA A QUE SE APRIETE UNA TECLA
CV ABE 960 LET A$=INKEY$:IF A$="" THEN GOTO 960
    S 970 LET KP=ASC(A$)
      980 RETURN

      985 REM INTRODUCE UNA O MAS LINEAS DE LONGITUD "IL"
      990 LET L=1:LET OL=1:LET B$=""

     1000 LET B$=C$(FICHA*CL+L)

    S 1005 REM Ver conversiones Spectrum

     1010 LET BL=LEN(B$):LET X=0

     1020 IF BL<IL AND KP<>ARRIBA AND KP<>ABAJO
          AND KP<>RETRO AND KP>31 THEN LET X=1

     1030 IF X=1 THEN LET B$=B$+CHR$(KP)

   AS 1040 IF KP=RETRO AND BL>0 THEN LET
          B$=LEFT$(B$,BL-1)

  ★  1050 PRINT TAB(DENT,TP+(L-1)*SI;B$;"*";

     1060 GOSUB890

     1070 IF L<>OL OR KP=13 THEN GOSUB1100
     1080 IF KP<>13 THEN GOTO1010
     1090 RETURN

     1095 REM COMPLETAR LA FICHA CON LA LINEA AÑADIDA
     1100 LET C$ (FICHA*CL+OL)=B$
  ★  1110 PRINT TAB(DENT,TP+(OL-1)*SI;C$(FICHA*CL+OL);" ";
     1120 LET B$=C$(FICHA*CL+L)
    S 1125 REM Ver conversiones Spectrum
     1130 RETURN

     1135 REM VISUALIZA LOS CARACTERES EN LA BASE DE LA PANTALLA
  ★  1140 PRINTTAB(0,TP+DP);S$;
  ★  1150 PRINTTAB(0,TP+DP);Y$;PRINTZ$;
     1160 RETURN

     1165 REN PRESENTACION GENERAL DE LA PANTALLA CON NUMERO DE FICHA
 CVA 1170 CLS
     1180 PRINT"ARCHIVO-CLASIFICADOR"
     1185 IF ULTIMA=0 THEN PRINT "NO HAY FICHAS EN EL ARCHIVO"
   V 1190 IF ULTIMA>0 THEN PRINT:ULTIMA;" FICHAS";
```

Rutinas de apoyo

900: Graba la línea actual del cursor en la variable OL.

910: Rutina INKEY$.

920-930: Calcula L (nueva posición del cursor) dependiendo de la tecla que se apriete.

940: Si la posición del cursor <>OL, quita el cursor de la posición OL.

960: Pone el carácter de la siguiente letra apretado en A$.

970: KP = código ASCII de la tecla apretada.

990: Da valor al número de línea del cursor y al antiguo número de línea del cursor y borra B$ para contener una nueva entrada.

1000: Pone cualquier dato que haya en la línea L en B$.

1010: Cuenta los caracteres en B$ y los graba en BL. X es una variable bandera que indica cuando la tecla apretada es un carácter válido para almacenar en B$.

1020: Da valor 1 a la variable X si BL < max. longitud en input y tecla apretada <> tecla para mover el cursor hacia arriba o hacia abajo, o para borrar un carácter, y KP>31 (es decir un carácter representable).

1030: Si X = 1 suma el carácter KP a B$.

1040: Si el usuario aprieta la tecla de borrar y BL (longitud de B$) >0 entonces borra el último carácter de B$.

1050: Escribe B$ seguido por el símbolo "*".

1060: Subrutina para mover el cursor e introducir el siguiente carácter.

1070: Si el usuario se ha movido a una nueva línea (L<>OL) o ha apretado RETURN, va a la subrutina para poner lo último introducido en la matriz C$.

1080: Si el usuario no ha apretado RETURN se repite la rutina input.

1100: Pone B$ en la matriz C$. El subíndice se calcula usando la fórmula descrita en la página 20.

1110: Escribe la línea recién metida en C$ seguida de un espacio para borrar el símbolo *.

1120-1130: Pone el dato en la línea L (posicion actual del cursor) en B$ para volver a 1080 y de aquí a 1010 para recibir el siguiente input.

1140: «Escribe» dos líneas de espacios en blanco para borrar la zona inferior.

1150: Escribe mensajes.

1180-1190: Escribe el título y el número de fichas.


```
     1200 IF FICHA<MX+1 AND ULTIMA>0 THEN
          PRINT " FICHA "; FICHA; " "

     1210 LET XC=W2-(LEN(X$)/2)
     1220 PRINTTAB(XC);X$
     1230 PRINTL$
  ★  1240 PRINTTAB(0,TP+DP-1);L$
     1250 RETURN

     1255 REM INICIALIZACION
     1260 GOSUB1440
     1270 LET MX=30:LET SC=0:LET CL=7:LET ML=6
    S 1280 DIM C$(MX*CL)
    S 1290 DIM M$(ML),H$(CL)
     1300 DATA"AÑADIR UNA FICHA","SUPRIMIR UNA FICHA","HACER CAMBIOS EN
     UNA FICHA"
     1310 DATA"BUSCAR FICHAS","CARGAR FICHAS","ARCHIVAR ESTAS FICHAS"
     1320 FOR I=1 TO ML:READ M$(I):NEXT I
     1330 DATA"NOMBRE","CALLE","CIUDAD",
     "CODIGO","TELEFONO","INFO"
     1340 FOR I=1 TO CL:READ H$(I):NEXT I
     1350 LET ULTIMA=0:LET FICHA=1:LET ND=0
     1360 LET F$="FLIPDAT":LET D$="! FICHA BORRADA !"
   AT 1380 LET OPCION=1
  BES 1390 LET ARRIBA=ASC("["): LET ABAJO=ASC("]"):LET
     OTRA=ASC("£"): LET RETRO=20
     1400 LET W$="USA "+CHR$ ARRIBA+" "+CHR$ (ABAJO)
     1410 LET L$=" ":FOR I=1 TO W:LET L$=L$+"•":NEXT I
     1420 LET S$=" ":FOR I=1 TO W•2:LET S$=S$+" ":NEXT I
     1430 RETURN
     1435 REM DETALLES DE PANTALLA
  VTS 1440 LET W=40:LET TP=5:LET DP=15
   VT 1450 LET S=2
    T 1460 LET W2=W/2:LET DENT=9:LET IL=W-DENT-2
     1465 REM Ver conversiones para Commodore, Apple, BBC Electron
     1470 RETURN
```

Rutina de apoyo

Inicialización

1200: Indica el número actual de ficha si no supera el máximo.

1210-1220: XC es la columna en la que escribe el mensaje X$ centrado en la parte superior izquierda de la pantalla.

1230-1240: Traza una línea de estrellas bajo el mensaje y a través de la parte inferior de la ficha.

1260: Subrutina con detalles específicos para la pantalla.

1270-1470: Ver tabla de variables para saber su significado.

1280: Crea una variable de una dimensión para contener MX × CL datos.

1300-1320: Opciones para el menú y bucle para introducirlos en M$.

1330-1340: Títulos de las fichas y bucle para introducirlos en H$.*

1360: FLIPDAT es el nombre del archivo para almacenar fichas.

1390: Pone los códigos ASCII de las teclas elegidas para mover el cursor, en las variables ARRIBA, ABAJO, RETRO y OTRA.

1400: Pone el mensaje y los símbolos de las teclas elegidas para mover el cursor, en la variable W$.

1410-1420: Crea la variable L$ para trazar una línea de estrellas a través de la pantalla, y S$ para «escribir» dos líneas de espacios en blanco para borrar mensajes.

1460: IL (nº de caracteres que puedes meter en una línea) = ancho de la pantalla menos el margen izquierdo de la ficha menos 2.

Alterar el programa

Para cambiar el programa y que almacene información distinta tendrás que variar los datos de los titulares de las fichas, que van a H$ (línea 1330). Si necesitas un número diferente de títulos tendrás que cambiar también CL en la línea 1270. Si tus títulos tienen más caracteres que los de *Archivo-Clasificador,* tendrás que cambiar la variable DENT en la línea 1460.

El *Archivo-Clasificador* está preparado para 30 fichas, pero si tienes suficiente memoria puedes aumentar el número de fichas cambiando MX en la línea 1270. Cada ficha llena ocupa unos 200 bytes en una pantalla de 40 columnas y 100 bytes en una pantalla de 20 columnas. El programa en sí ocupa unos 4 ½K por lo que puedes calcular cuántas tarjetas puede controlar tu computadora.

Conversión del *Archivo-Clasificador* BBC, Electron

```
690 IF OPCION=5 THEN FILE=OPENIN F$:INPUT
  # FILE,NC
700 IF OPCION=6 THEN FILE=OPENOUT F$
  :PRINT # FILE,LAST-ND
750 IF OPCION=5 THEN INPUT # FILE,C$(J•CL+K)
760 IF OPCION=6 AND C$(J•CL+CL)<>D$ THEN
  PRINT # FILE,C$(J•CL+K)
790 CLOSE # 0
960 LET A$=INKEY$(0):IF A$="" " THEN GOTO
  960
1390 LET ARRIBA=ASC("["):LET ABAJO=ASC("]"):LET
  OTRA=ASC("£"):LET RETRO=127
1465 MODE 6
1466 •FX4,1
1467 VDU23;8202;0;0;0;
```


*En otros países que no sean Inglaterra pon ESTADO en lugar de Provincia.

Continúa en la página siguiente.

Conversiones *Archivo-Clasificador*

C64, VIC-20 (★ = VIC-20 sólo)

Convierte las instrucciones PRINT TAB (X, Y) usando el método de la página 45.

```
210,1170 Sustituye CLS por PRINT CHR$(147);
690 IF OPCION=5 THEN OPEN 1,1,0,F$:INPUT # 1
    ,NC
700 IF OPCION=6 THEN OPEN 1,1,1,F$:PRINT # 1
    ,LAST-ND
750 IF OPCION=5 THEN INPUT # 1,C$(J*CL+K)
752 IF OPCION=5 AND C$(J*CL+K)="!!" THEN
    LET C$(J*CL+K)=" "
755 IF OPCION=6 AND C$(J*CL+K)=" " THEN LET
    C$(J*CL+K)="!!"
760 IF OPCION=6 AND C$(J*CL+CL)<>D$ THEN
    PRINT # 1,C$(J*CL+K)
790 CLOSE 1
960 GET A$:IF A$=" " THEN GOTO 960
★ 1190 IF ULTIMA>0 THEN PRINT; ULTIMA;"FICHAS";
★ 1440 LET W=22:LET TP=6:LET DP=10
★ 1450 LET S=1
1465 LET C$=" ":FOR I=1 TO 30:LET C$=C$+CHR$(17):NEXT I
```

Apple

Convierte las instrucciones PRINT TAB (X, Y) como se describe en la página 45, cambiando TAB (0) por HTAB 1.

```
50,820 No pongas los puntos y coma
70,850 Cambia TAB(1) por TAB(2)
210,1170 Sustituye CLS por HOME
690 IF OPCION=5 THEN PRINT:PRINT G$;"OPEN";F$:PRINT
    G$;"READ";F$:INPUT NC
700 IF OPCION=6 THEN PRINT:PRINT G$;"OPEN";F$:PRINT
    G$;"WRITE";F$:PRINT ULTIMA-ND
710 No pongas esta línea
750 IF OPCION=5 THEN INPUT C$(J*CL+K)
760 IF OPCION=6 AND C$(J*CL+CL)<>D$ THEN PRINT
    C$(J*CL+K)
790 PRINT G$;"CLOSE";F$
960 LET A$=" ":IF PEEK(-16384)>127 THEN GET A$
965 IF A$=" " THEN GOTO 960
1040 IF KP=RETRO AND BL>1 THEN LET B$=LEFT$(B$,BL-1)
1045 IF BL=1 AND KP=RETRO THEN LET B$=" "
1390 LET ARRIBA=ASC("["):LET ABAJO=ASC("]"):LET
    OTRA=ASC("€"):LET RETRO=8
1465 LET G$="CTRL D"
```

↖ aprieta las teclas CTRL y D al mismo tiempo

TRS-80 Colour Computer

Convierte las instrucciones PRINT TAB (X, Y) como se describe en la página 45. Tendrás que poner paréntesis alrededor de la expresión Y.

```
6 CLEAR 8000
50 PRINT€32*TP," ";
110,180,500 Cambia la palabra RETURN por ENTER
690 IF OPCION=5 THEN OPEN "I", # -1,F$:INPUT # -1,NC
700 IF OPCION=6 THEN OPEN "O", # -1,F$:PRINT # -1,LAST-ND
750 IF OPCION=5 THEN INPUT # -1,C$(J*CL+K)
760 IF OPCION=6 AND C$(J*CL+CL)<>D$ THEN PRINT # -1,C$
    (J*CL+K)
790 CLOSE # -1
820 PRINT€32*TP," ";
1390 LET ARRIBA=ASC("["):LET ABAJO=ASC("]"):LET OTRA=ASC("€"):
    LET RETRO=8
1440 LET W=32:LET TP=4:LET DP=8
1450 LET S=1
1460 LET W2=W/2:LET DENT=9:LET IL=W-DENT-3
```

Spectrum

No puedes borrar fichas ya que el Spectrum graba todas las fichas, incluyendo las borradas. Usa la opción Hacer cambios para quitar las fichas que no quieras.

Cambia las instrucciones PRINT TAB (X, Y) como se describe en la página 45.

```
110,180,500 Cambia la palabra RETURN por ENTER
185 LET C=OPCION
190 GOSUB 230*(C=1)+290*(C=2)+350*(C=3)+390*(C=4)+
    690*(C>4)
690 IF OPCION=5 THEN LOAD F$ DATA N(0):LET NC=N(1):LOAD F$
    DATA C$()
700 IF OPCION=6 THEN LET N(1)=ULTIMA:SAVE F$ DATA N():
    SAVE F$ DATA C$()
720 IF OPCION=5 THEN LET ULTIMA=ULTIMA+NC
730-790 No pongas estas líneas
970 LET KP=CODE(A$)
1005 IF LEN(B$)>1 AND B$(LEN(B$))=" " THEN LET B$=B$
    (TO LEN(B$)-1):GOTO 1005
1040 IF KP=RETRO AND BL>1 THEN LET B$=B$( TO BL-1)
1125 IF LEN(B$)>1 AND B$(LEN(B$))=" " THEN LET
    B$=B$( TO LEN(B$)-1):GOTO 1125
1280 DIM C$(MX*CL,22):DIM N(1)
1290 DIM M$(ML,16):DIM H$(CL,6)
1390 LET ARRIBA=CODE "[":LET ABAJO=CODE "]":LET    OTRA=CODE
    "€":LET RETRO=12
1440 LET W=32:LET TP=5:LET DP=14
```


Guía BASIC

En las dos páginas siguientes hay una lista de las instrucciones en BASIC usadas en este libro, con breves explicaciones. Las instrucciones que varían según la computadora tienen una estrella al lado y puedes encontrar sus conversiones en la página 45.

AND Es para dar a la computadora más de una prueba en una instrucción IF/THEN. La instrucción que haya detrás de THEN sólo se llevará a cabo si las dos pruebas unidas por un AND son verdaderas.

```
IF X>Y AND Y<100 THEN PRINT "VERDADERO"
```

*** ASC** Da el número de código ASCII de un carácter.

```
PRINT ASC("J")          Código ASCII
74                      de la letra J.
```

CHR$ Da el carácter correspondiente a un número en código ASCII.

```
PRINT CHR$(74)
J
```

*** CLS** Borra la pantalla.

DEF FN nombre Es la abreviatura de función definida. Se usa esta instrucción para decir a la computadora que lleve a cabo un determinado cálculo. Luego, cada vez que quieras que realice dicho cálculo usas la instrucción FN nombre. El ejemplo le dice a la computadora que convierta una distancia en millas en kilómetros. PRINT FNKILO escribe la solución para la distancia D.

```
DEF FNKILO=M*1.6093
INPUT D
PRINT FNKILO(D)
```

DIM Le dice a la computadora cuánto espacio debe reservar para una matriz. Por ejemplo DIM D$ (30) crea una matriz de una dimensión para 30 datos. * DIM (4,7) crea una matriz de dos dimensiones para 28 datos distribuidos en cuatro filas de siete columnas.

END Para el programa que se está ejecutando.

ELSE Le dice a la computadora lo que debe hacer si una prueba IF/THEN no es verdadera.

```
IF X>Y THEN PRINT X ELSE LET X=X*2
```

Si tu computadora no tiene esta instrucción puedes lograr el mismo efecto con una nueva instrucción IF/THEN en la línea siguiente.

```
IF X>Y THEN PRINT X
IF X<=Y THEN LET X=X*2
```

FOR/TO NEXT Le dice a la computadora que repita una serie de instrucciones un número determinado de veces. Este ejemplo repite la instrucción PRINT 5 veces.

```
FOR J=1 TO 5
PRINT J
NEXT J
```

GOSUB Manda a la computadora a una subrutina que comienza en una determinada línea. Al final de la subrutina la instrucción RETURN hace que la computadora vuelva a la línea posterior a GOSUB.

```
100 GOSUB 200
110 PRINT "PROXIMA INSTRUCCION POR FAVOR"

200 PRINT "ESTO ES UNA"
210 PRINT "BREVE SUBRUTINA"
220 RETURN
```

GOTO Manda a la computadora a un número específico de línea desde donde continúa ejecutando el programa.

IF/THEN Hace que la computadora compruebe algunos datos, y si la prueba es verdadera continúa con las instrucciones que siguen a THEN. Puedes usar los siguientes símbolos para comprobar datos: = (igual); <> (no igual); > (mayor que); < (menor que); >= (mayor o igual que); <= (menor o igual que).

```
IF X>Y THEN PRINT "X ES MAYOR"
```

*** INKEY$** Le indica a la computadora que lea el siguiente carácter tecleado. En el ejemplo el carácter se almacena en la variable A$.

```
LET A$=INKEY$
```

INPUT Hace que la computadora espere a que el usuario introduzca algún dato. Estos se almacenan en la variable dada detrás de la instrucción INPUT.

```
INPUT T$
```

INT Convierte fracciones decimales en números enteros ignorando todas las cifras decimales.

```
PRINT INT(5.6093)
5
```

*Para matrices de cadena en el Spectrum también debes decir a la computadora la longitud del dato más largo. Ej: DIM D$ (30, X) donde X es el número de caracteres del dato más largo de la cadena.

*** LEFT$** Toma un determinado número de caracteres desde el lado izquierdo de la variable de cadena. PRINT LEFT$ (A$, 3) escribe 3 caracteres de A$ contados desde la izquierda.

```
LET A$="ABCDEF"
PRINT LEFT$(A$,3)
ABC
```

LEN Cuenta el número de caracteres en una variable de cadena.

```
LET A$="ABCDEF"
PRINT LEN(A$)
6
```

LET Pone algún dato en una variable determinada.

```
LET B$="NIGHT INGALE"
LET R=5
```

*** MID$** Toma cierto número de caracteres de una variable de cadena, comenzando por un carácter específico. El ejemplo escribe 2 caracteres de A$ comenzando desde el carácter 3.

```
LET A$="ABCDEF"
PRINT MID$(A$,3,2)
CD
```

*** ON GOSUB** Manda a la computadora a una determinada subrutina dependiendo del valor de la variable. En el ejemplo si C = 3 la computadora irá a la tercera subrutina listada detrás de GOSUB, es decir, a la que empieza en la línea 250.

```
ON C GOSUB 100,170,250,400,700
```

OR Se usa en instrucciones IF/THEN para dar a la computadora más de una prueba. Cuando las pruebas están unidas por OR, las instrucciones de detrás del THEN se llevarán a cabo si una cualquiera de las pruebas es verdadera.

```
IF X>Y OR Y>100 THEN PRINT "VERDADERO"
```

*** PEEK** Hace que la computadora mire el contenido de una determinada ubicación de memoria. PRINT PEEK (dirección) escribe el contenido de dicha ubicación. El BBC y el Electron usan PRINT? dirección en lugar de PEEK.

*** POKE** Almacena un número en la ubicación de memoria especificada en la instrucción. Por ejemplo, POKE 4057, 64 pone el número 64 en la ubicación 4057. El BBC y el Electron usan ?4057 = 64.

```
POKE 4057,64
```

PRINT Le dice a la computadora que visualice algún dato, o el resultado de un cálculo en la pantalla. Por sí sólo deja una línea en blanco.

```
PRINT "SOLUCION"
PRINT LEN(B$)
PRINT 256
```

READ/DATA La instrucción READ le dice a la computadora que almacene los datos listados en las líneas que comienzan con la palabra DATA, en variables o en matrices. Suele usarse con un bucle FOR/NEXT que repita la instrucción READ.

```
FOR J=1 TO 5
READ B(J)
NEXT J
DATA 43,67,123,34,46
```

REM Quiere decir recordatorio o comentario. La computadora ignora cualquier cosa que vaya detrás de la palabra REM y se usa para poner notas aclaratorias del programa.

RESTORE Se usa con READ y DATA. Cada vez que usas la instrucción READ la computadora lee el siguiente dato de la lista. La instrucción RESTORE hace que vuelva a empezar a leer los datos desde el principio. Algunas computadoras tienen la instrucción RESTORE «número de línea» que hace que lea los datos a partir de una determinada línea.

RETURN Ver GOSUB.

RIGHT$ Toma un determinado número de caracteres empezando por la derecha de la variable de cadena.

```
LET A$="ABCDEF"
PRINT RIGHT$(A$,3)
DEF
```

STEP Le dice a la computadora cómo debe contar al repetir el bucle FOR/NEXT. En el ejemplo la computadora cuenta hasta 6 de dos en dos y, por tanto, repite el bucle tres veces.

```
FOR J=1 TO 6 STEP 2
PRINT J
NEXT J
```

VAL Le dice a la computadora que trate una cadena como un número y por tanto es útil cuando has almacenado números en una variable de cadena.

```
LET A$="1066"
LET A=VAL(A$)
```


Tabla de conversión

Cuando no se da la conversión, la computadora usa las instrucciones standard.

BASIC standard	C64/VIC-20	Apple	TRS-80	BBC/Electron	Spectrum
PRINT ASC("J")					PRINT CODE("J")
CLS	PRINT CHR$(147)	HOME			
LET I$=INKEY$	GET I$	GET I$		LET I$=INKEY$(X) X = tiempo que debe esperar la computadora.	INKEY$ La tecla apretada se almacena en una variable llamada INKEY$.
PRINT LEFT$(A$,3) PRINT MID$(A$,3,2) PRINT RIGHT$(A$,3)					PRINT A$(1 TO 3) PRINT A$(3 TO 5) PRINT A$(3 TO 8) A$ contiene 6 caracteres.
PRINT TAB(X,Y);M$	Ver debajo.	HTAB X:VTAB Y:PRINT M$	PRINT @32*Y+X;M$		PRINT AT Y,X;M$
Instrucciones para controlar el casete. "nombre" = nombre del archivo. D$ = datos para ser grabados. Las instrucciones dadas para el Apple son para disco. Sólo puedes grabar matrices enteras en casete.	REM SAVING D$ OPEN 1,1,1,"nombre" PRINT #1,D$ CLOSE 1 REM LOADING D$ OPEN 1,1,0,"nombre" INPUT # 1,D$	REM SAVING DATA PRINT D$;"OPEN";"nombre" PRINT D$;"WRITE";"nombre" PRINT nombre de la variable PRINT D$;"CLOSE";"nombre" REM LOADING DATA PRINT D$;"OPEN";"nombre" PRINT D$;"READ";"nombre" INPUT nombre de la variable PRINT D$;"CLOSE";"nombre" Pon esta instrucción en la zona de inicialización del programa*. LET D$="CTRL D"	REM SAVAING D$ OPEN "O", #-1,"nombre" PRINT #-1,D$ CLOSE #-1 REM LOADING D$ OPEN "I", #-1,"nombre" INPUT #-1,D$ CLOSE #-1	REM SAVING D$ LET A=OPENOUT "nombre" PRINT # A,D$ CLOSE # A REM LOADING D$ LET A=OPENIN "nombre" INPUT # A,D$ CLOSE # A	REM SAVING D$ SAVE "nombre" DATA D$() REM LOADING D$ LOAD "nombre" DATA D$() Sólo puedes grabar matrices. No puedes grabar variables.

PRINT TAB (X, Y) en C64 y VIC-20
Para escribir en la columna X, línea Y, añade esta línea en la inicialización del programa.

```
LET C$=" ":FOR J=1 TO número de líneas en pantalla:
LET C$=C$+CHR$(17):NEXT J
```

CHR$ (17) es el código ASCII para mover el cursor hacia abajo. Esta línea crea una variable C$ con tantos códigos como líneas hay en tu pantalla.

Usa esta instrucción para escribir M$ en la posición de pantalla X, Y.

```
PRINT CHR$(19);LEFT$(C$,Y);SPC(X);M$
```

CHR$ (19) es el código ASCII para mover el cursor a la parte superior izquierda de la pantalla. LEFT$ (C$, Y) mueve el cursor desde C$. Y códigos hacia abajo. SPC (X) mueve X espacios a través de la pantalla.

Spectrum ON GOSUB
El Spectrum no tiene esta instrucción. En su lugar puedes hacer que la computadora calcule el lugar correcto al que ir. Por ejemplo, esto es la conversión de la línea 120 del esqueleto del Archivo-Clasificador de la página 7.

```
120 GOSUB CH+100+200
```

Cuando CH es 1 la computadora va a la línea 300. Cuando CH es 2 va a la línea 400, etc. Cuando escribas tus propios programas, elige números de líneas para las subrutinas que puedan ser usadas para calcular con GOSUB.

*Aprieta las teclas CTRL y D a la vez (nada sucede en la pantalla).
No puedes usar D$ como variable en ninguna otra parte del programa ya que se usa en esta instrucción.

Conversión de las rutinas de este libro

En estas dos páginas hay algunas conversiones que te ayudarán a ejecutar las rutinas de este libro en las computadoras Commodore 64, VIC-20, Apple, TRS-80 Colour Computer con BASIC ampliado, BBC, Electron y Spectrum. Si tienes otro tipo de computadora puedes hacer las conversiones consultando en tu manual y en los listados que aquí damos.

Página 17. Trucos con bucles para el Spectrum

En el Spectrum debes decir a la computadora cuántos caracteres debe buscar en D$, tal y como se muestra aquí.

```
30 IF D$(J)I TO LEN(R$))=R$ etc.
```

Página 21. Rutina de claves para el Spectrum

En el Spectrum no puedes grabar variables, sólo puedes grabar matrices como archivos de datos. En la rutina de la clave, puedes almacenar la clave en una matriz tal y como ves aquí.

```
REM RUTINA DE LA CLAVE
DIM P$(1,n.º de caracteres)
LET P$(1)=;"clave"
SAVE "nombre del archivo" DATA P$()
Graba datos en el mismo archivo
```

La línea 20 crea una matriz PW$ con un elemento. Tienes que completar con el número de caracteres en tu clave.

```
REM COMPROBACION DE LA CLAVE
PRINT "CLAVE POR FAVOR"
INPUT Q$
LOAD "nombre del archivo" DATA P$()
IF Q$< >P$(1) THEN LET P$(1)=" ":NEW
```

Tendrás que añadir estas líneas en la rutina de comprobación de la clave.

Página 23. Truco para amontonar datos-TRS-80

En el TRS-80 si tienes más de 200 números debes crear espacio en la zona de variables de la memoria usando la instrucción CLEAR seguida por el número de números que quieres almacenar.

Página 23. Truco para amotonar números para el Spectrum

En el Spectrum debes almacenar los números en una matriz llamada P$, para que puedas grabarlos como un archivo de datos. Para hacer esto haz los siguientes cambios:

```
5 DIM P$(1, n.º de números)
60 FOR I=1 TO (n.º de números)
70 LET P$(1,I)=CHR$ (CANTIDAD(I))
80 NEXT I
90 PRINT CODE(P$(1,21))
120 SAVE "nombre del archivo" DATA P$()
```

Borra líneas 100, 110, 130 y 140.

Esto crea una matriz P$ con un elemento para contener tantos caracteres como números tengas. Estás reservando espacio de memoria al usar una matriz por lo que no tienes que llenar P$ con estrellas. Las líneas 60-80 ponen el código de cada uno de los números almacenados en CANTIDAD, en la matriz P$. La línea 90 escribe el valor ASCII del veintiunavo carácter en P$. La línea 120 graba la matriz P$ como un archivo de datos.

En la rutina para usar P$ tienes que sustituir las líneas 10-50 por estas dos líneas:

```
10 DIM P$(1, n.º de números)
20 LOAD "nombre del archivo" DATA P$()
```

Página 27. Control del input

En el Spectrum tienes que decirle a la computadora cuántos caracteres debe comprobar.

```
230 IF A$=W$(J) ( TO LEN(A$)) THEN LET X=1
```

Página 27. Truco con INKEY$

Esta rutina funcionará en el TRS-80. Estas son las conversiones a realizar para las demás computadoras.

C64, VIC-20, Apple

```
440 GET I$:IF I$=" " THEN GOTO 440
```

BBC, Electron

```
440 LET I$=INKEY$(0):IF I$=" " THEN GOTO 440
475 PRINT
```

Spectrum

```
450 LET CK=CODE(I$)
465 FOR T=1 TO 80:NEXT T
```


Páginas 28-29 Consejos para la visualización en pantalla

Estas rutinas funcionarán en BBC y Electrón, aquí tienes la conversión para las otras computadoras.

C64, VIC 20

Estas conversiones usan el método PRINT TAB descrito en la página 45.

Rutina mensaje

```
80 PRINT CHR$(147)
500 PRINT CHR$(19); LEFT$(C$,Y);SPC(X);CL$
510 PRINT CHR$(19);LEFT$(C$,Y);SPC(X);M$
1025 LET C$=" ":FOR J=1 TO n.º de líneas en la
pantalla:LET C$=C$+CHR$(17):NEXT J
```

25 para el C64; 23 para el VIC-20

Línea horizontal

```
25 LET C$=como línea 1025 anterior
30 PRINT CHR$(19);LEFT$(C$,Y);SPC(X);L$
```

Línea vertical

Cursor abajo = 17; cursor izquierda = 157.

```
25 LET C$=como línea 1025 anterior
70 PRINT CHR$(19);LEFT$(C$,Y);SPC(X);L$
```

Rutina ventana

```
100 PRINT CHR$(147)
210 PRINT CHR$(19);LEFT$(C$,Y+I);SPC(X);L$
220 PRINT CHR$(19);LEFT$(C$,Y+I);
SPC(X+W-1);L$;
260 PRINT CHR$(19);LEFT$(C$,Y);SPC(X+J);L$;
445 LET C$=como línea 1025 anterior
```

Centrar el texto

```
310 PRINT CHR$(19);LEFT$(C$,Y-INT(H/2));
SPC(X+C);M$
```

Apple

Rutina mensaje

```
80 HOME
500 HTAB X:VTAB Y:PRINT CL$
510 HTAB X:VTAB Y:PRINT M$
```

Línea horizontal

```
30 HTAB X:VTAB Y:PRINT L$
```

Línea vertical

Cursor abajo = 10; cursor izquierda = 8.

```
70 HTAB X:VTAB Y:PRINT L$
```

Rutina ventana

```
100 HOME
210 HTAB X:VTAB Y+1:PRINT L$
220 HTAB X+W-1:VTAB Y+I:PRINT L$;
260 HTAB X+J:VTAB Y:PRINT L$;
```

Centrar el texto

```
310 HTAB X+C:VTAB Y-INT(H/2):PRINT M$
```

TRS-80 Colour Computer

Rutina mensaje

```
510 PRINT @32*Y+X,M$
510 PRINT  @ 32*Y+X,M$
```

Línea horizontal

```
30 PRINT @32*Y+X,L$
```

Línea vertical

Esta rutina no funcionará en esta computadora.

Rutina ventana

```
210 PRINT @32*(Y+I)+X,L$
220 PRINT @32*(Y+I)+X+W-1,L$;
260 PRINT @32*Y+X+J,L$;
```

Centrar el texto

```
310 PRINT @32*(Y-INT(H/2))+X+C,M$;
```

Spectrum

Rutina mensaje

```
500 PRINT AT Y,X;CL$
510 PRINT AT Y,X;M$
```

Cambia CL$ por C$
Cambia END por STOP

Línea horizontal

```
30 PRINT AT Y,X;L$
```

Línea vertical

No puedes usar este método en el Spectrum.

Rutina ventana

```
210 PRINT AT Y+I,X;L$
220 PRINT AT Y+I,X+W-1;L$;
260 PRINT AT Y,X+J;L$;
```

Centrar el texto

```
310 PRINT AT Y-INT (H/2),X+C;M$
```

Página 31. Entrada de palabras

En el Spectrum tienes que decir a la computadora cuántos caracteres comparar.

```
320 IF W$(J)(TO LEN(A$))=A$ THEN LET C=J
```

También tendrás que convertir la instrucción ON GOSUB como se explica en la página 45.

Menú moviendo el cursor C64, VIC-20

```
270 PRINT CHR$(147)
310 PRINT CHR$(19);LEFT$(C$,NL);SPC(1);
"JUGAR UNA NUEVA PARTIDA"
380 PRINT CHR$(19);LEFT$(C$,CY);SPC(CX);">":LET
OY=CY
390 GET I$:IF I$=" " THEN GOTO 390
420 IF CY< >OY THEN PRINT CHR$(19);
LEFT$ C$,OY);SPC (CX); " "
1015 LET C$=" ":FOR J=1 TO n.º de líneas en la
pantalla: LET C$=C$+CHR$ (17):NEXT J
```


Continúa en la página siguiente.

Menú moviendo el cursor

Apple

En las líneas 290, 300, 320, 330, 350, 360, 370 cambia PRINT TAB (1) por PRINT TAB (2).

```
270 HOME
310 HTAB 2:VTAB NL:PRINT "JUGAR UNA NUEVA PARTIDA"
380 HTAB CX:VTAB CY:PRINT ">":LET OY=CY
390 GET I$:IF I$=" " GOTO 390
420 IF CY<>OY THEN HTAB CX:VTAB OY:PRINT " "
```

En la línea 1000 haz que la variable CX valga 1.

TRS-80 Colour Computer

Borra la línea 370.

```
310 PRINT @32*NL+1,"JUGAR UNA NUEVA PARTIDA"
380 PRINT @32*CY+CX,">";:LET OY=CY
420 IF CY<>OY THEN PRINT @32*OY+CX," ";
```

BBC, Electron

```
390 LET I$=INKEY$(0):IF I$=" " THEN GOTO 390
```

Spectrum

```
310 PRINT AT NL,1;"JUGAR UNA NUEVA PARTIDA"
380 PRINT AT CY,CX;">":LET OY=CY
390 IF INKEY$=" " THEN GOTO 390
400 IF INKEY$="A" AND CY>NL THEN LET CY=CY-SP
410 IF INKEY$="Z" AND CY<LC THEN LET CY=CY+SP
420 IF CY<>OY THEN PRINT AT OY,CX;" "
430 IF INKEY$<>CHR$(13) THEN GOTO 380
450 GOSUB C*100+400
```

Indice

Colección Electrónica

distribuidor exclusivo
cesma sa
C/ Aguacate, 25
28044 MADRID

Colección Electrónica

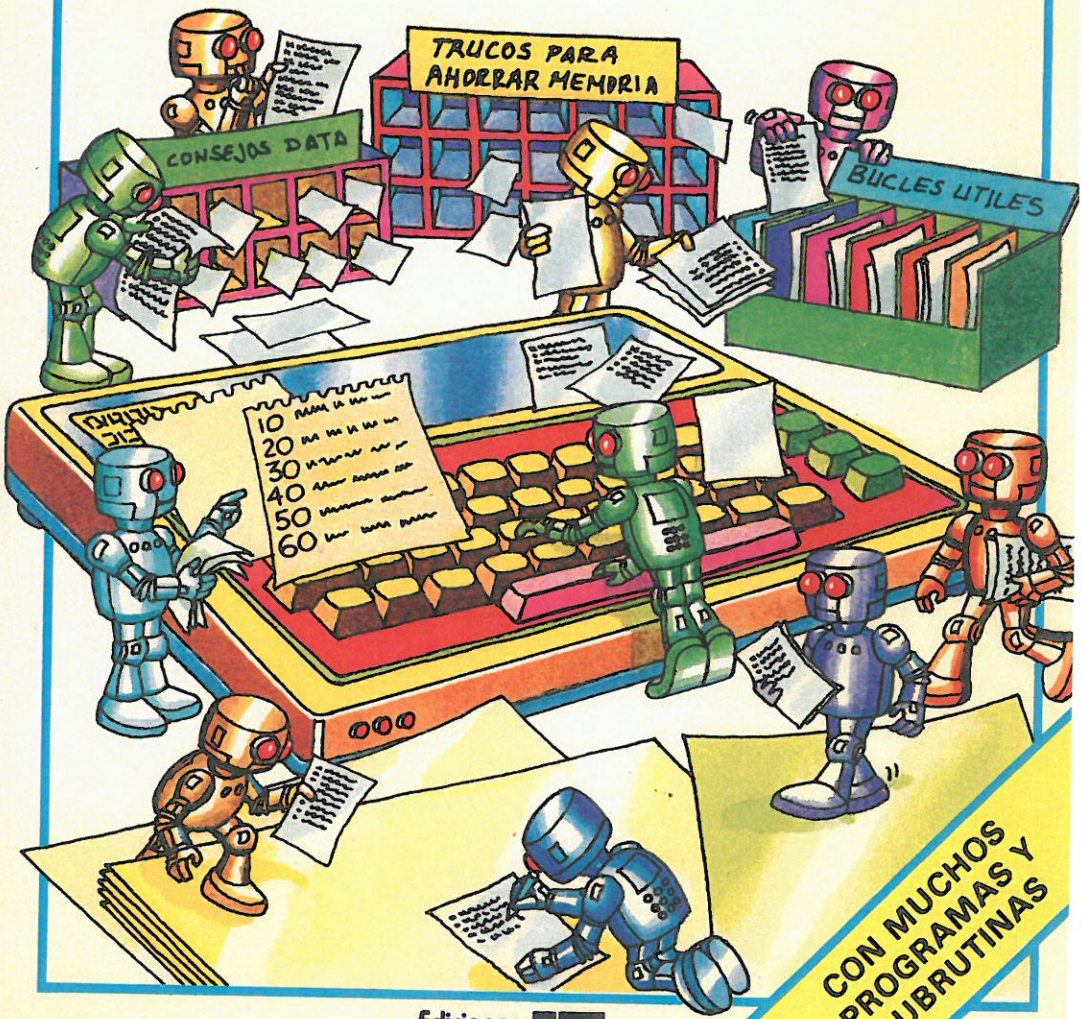
TECNICAS Y TRUCOS DE PROGRAMACION

PLESA-SM

Colección Electrónica

TECNICAS Y TRUCOS DE PROGRAMACION

Métodos y consejos para una mejor programación



CON MUCHOS PROGRAMAS Y SUBRUTINAS

Ediciones Pleso sm