

MICROBASIC

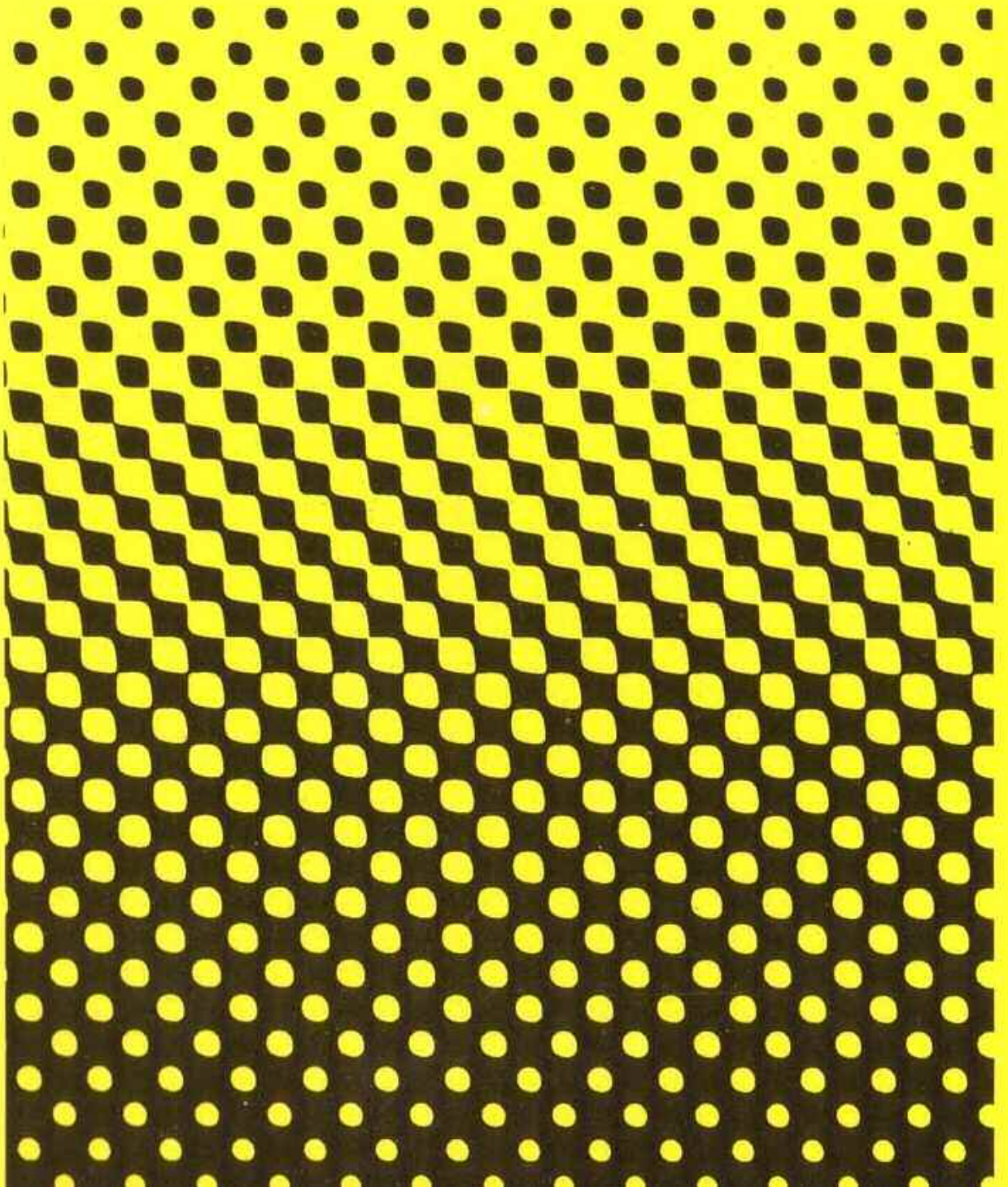
MICROBASIC

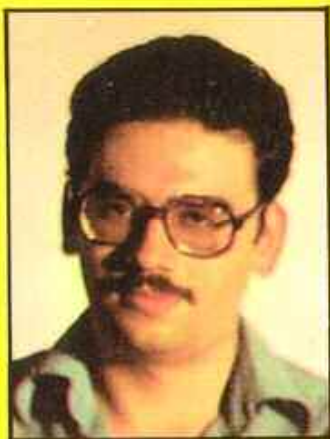
Basic SINCLAIR

Rafael PRADES

Rafael PRADES

BASIC SINCLAIR





Cuando en Junio de 1984 se empezó a gestar la creación del semanario MICRO-HOBBY, Rafael Prades fue la primera persona en contactar con la dirección del proyecto.

Nacido en agosto de 1956, casado y con tres hijos, ha realizado estudios de Maestría Industrial en la especialidad de Electrónica.

En 1974 entra en Standard Eléctrica, donde actualmente trabaja en el desarrollo del Sistema Doce de telefonía Digital como técnico de Hardware.

Curiosamente, y pese a ser un gran especialista en Hardware, se ha adaptado perfectamente a los temas de Software que le han sido encomendados, entre ellos, el Curso de Basic que nos ocupa.

El carácter eminentemente didáctico que imprime a todos sus artículos y, particularmente a este libro, se debe a su dilatada experiencia como Profesor de Fotografía, una de sus grandes pasiones, junto con el Montañismo.

HOBBY PRESS, S.A.
Editamos para gente inquieta.

MICROHOBBY

SEMANTAL

BASIC SINCLAIR

Rafael PRADES

Con la aparición del primer número de MICROHOBBY comienza también una serie de capítulos dedicados al estudio del sistema BASIC como lenguaje de programación del ordenador Spectrum. El estudio, que será bastante completo, no se limitará sólo a definir las funciones de los comandos o sentencias, tratará al mismo tiempo de explicar todos los posibles argumentos asociados a éstos. Asimismo, cada tema irá acompañado de una larga y clara lista de ejemplos. El curso está pensado tanto para quienes se han comprado un ordenador Spectrum para pasar el rato y matar marcianitos, como para aquellas personas que, teniendo cierta experiencia, deseen ampliar sus conocimientos sobre el tema. Una vez terminado el curso, presentaremos una serie de programas que serán explicados, analizados y convenientemente acompañados de diagramas de bloques y flujos.

HOP
EDITA
HOBBY
PRESS, S. A.

MICROBASIC 1

BASIC SINCLAIR

Dos personas que quieran comunicarse necesitan un medio de expresión; éste puede ser el lenguaje hablado, escrito, la mímica, dibujos, etc...; de la misma manera, nosotros para podernos comunicar con nuestro ordenador necesitamos conocer un lenguaje que él entienda. El utilizado por el Spectrum se denomina Lenguaje de Programación BASIC (iniciales en inglés de Beginner's All purpose Symbolic Instruction Code, que traducido al español significa «Código de instrucciones simbólicas de uso general para principiantes»).

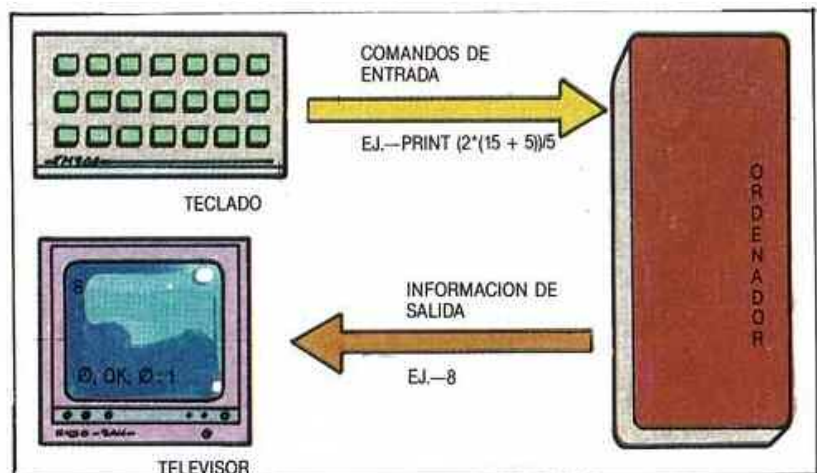
El lenguaje del Spectrum

El BASIC es un lenguaje utilizado también por otros ordenadores personales, aunque cada cual tiene ciertas particularidades que lo diferencian de los demás. En esta serie de artículos sólo vamos a tratar el lenguaje BASIC que el Sinclair Spectrum entiende.

Ya sabemos cuál es el medio de expresión que debemos utilizar con nuestro ordenador, pero ¿cómo se emplea? La respuesta es sencilla: el BASIC consta de una serie de comandos o sentencias que el programador introduce en el ordenador mediante el manejo del teclado. El aparato nos devuelve la información pedida en la pantalla del televisor. Por ejemplo, si quisiéramos que el ordenador nos calculara la operación:

$$\frac{2(15 + 5)}{5}$$

2 MICROBASIC



El ordenador, una vez recibido el comando que se ha introducido mediante el teclado, realiza las operaciones necesarias y visualiza el resultado en la pantalla del televisor.

y que nos devolviera el resultado, introduciríamos en el teclado el comando:

PRINT 2 * (15 + 5) / 5

El ordenador, una vez recibido el comando, calcula la expresión aritmética y posteriormente visualiza en la pantalla del televisor el resultado.

Como decíamos más arriba, el lenguaje BASIC consta de una serie de comandos o sentencias con las que se establece un diálogo o comunicación con el ordenador, pero ¿qué son las sentencias? Los comandos o sentencias son palabras clave derivadas del idioma inglés (PRINT, RUN, STOP...); estos términos, al ser introducidos en el ordenador mediante el teclado, indican a éste qué función debe de realizar.

Normalmente, las sentencias por sí solas no podrían realizar sus funciones si no

fueran acompañadas de un argumento, ya que éste aporta el dato o datos necesarios para que el comando o sentencia pueda ser ejecutado. Los argumentos pueden ser números, variables, expresiones aritméticas o cadenas de caracteres.

Se denomina *instrucción* al conjunto de una sentencia con su correspondiente argumento, y tiene la siguiente estructura:

| INSTRUCCION | |
|-------------|-----------|
| SENTENCIA | ARGUMENTO |

Por ejemplo:

- a) El argumento es un número.
Función: saltar a la subrutina localizada en la línea 350.

| SENTENCIA | ARGUMENTO |
|-----------|-----------|
| GOSUB | 350 |

- b) El argumento es una variable.
 Función: Leer un dato de una tabla y asignarlo a la variable «C».

| SENTENCIA | ARGUMENTO |
|-----------|-----------|
| READ | C |

- c) El argumento es una expresión matemática.
 Función: asignar a la variable «X» el resultado de la operación:

$$\frac{(15 \times 2) + (8/3)}{5}$$

| SENTENCIA | ARGUMENTO |
|-----------|----------------------------|
| LET | X = ((15 * 2) + (8/3)) / 5 |

- d) Y, por último, el argumento es una cadena de caracteres.
 Función: visualizar la siguiente cadena alfanumérica «Revista Microhobby».

| SENTENCIA | ARGUMENTO |
|-----------|----------------------|
| PRINT | «Revista Microhobby» |

No se preocupe si aún no entiende qué es una subrutina, qué es una variable o cómo se introducen estas sentencias, ya que esto se irá viendo en capítulos siguientes.

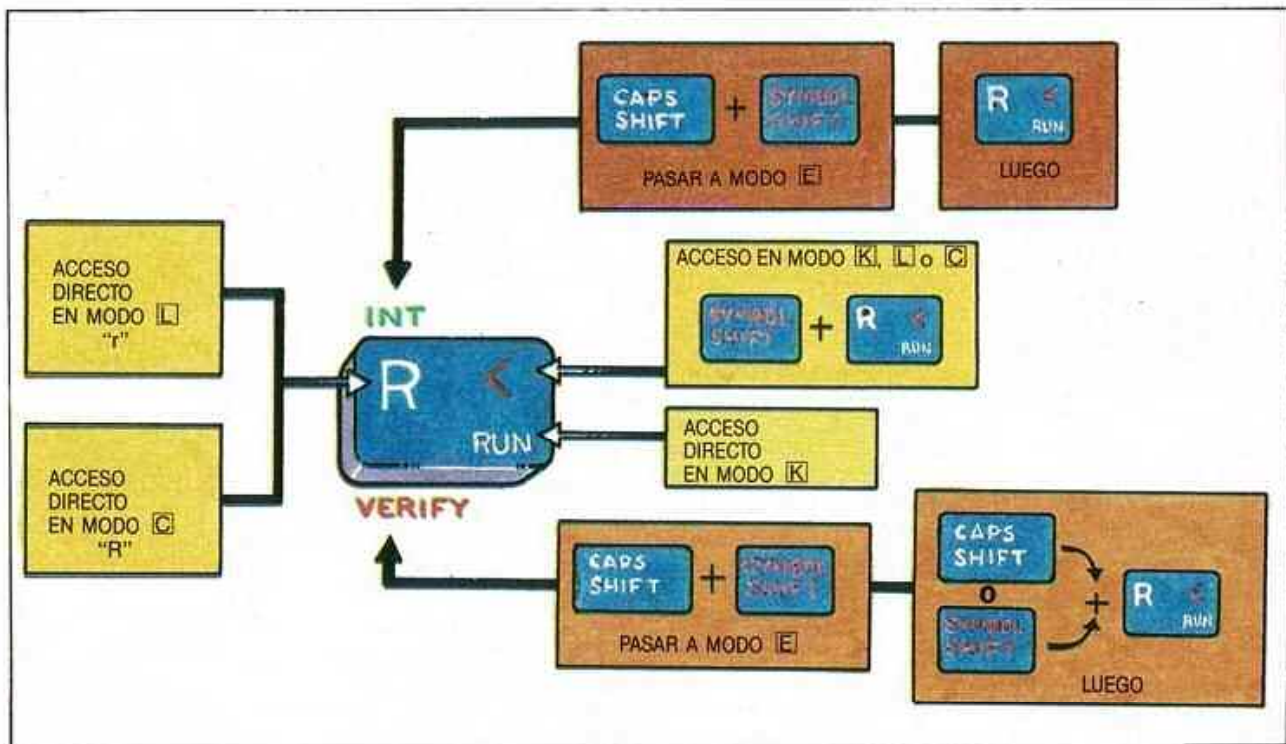
Acceso al teclado

Una de las características del Spectrum consiste en que no es necesario teclear los comandos o sentencias letra a letra, sino que éstos ya están definidos y asignados a una tecla. Por ejemplo: para acceder a la sentencia PRINT no se necesita pulsar las teclas correspondientes a las letras P, R, I, N y T; con sólo accionar la «P» aparece escrita en la pantalla del televisor el comando PRINT.

El Spectrum tiene 40 teclas; con éstas sería imposible albergar los 88 diferentes comandos que existen, así como las diferentes funciones de control, los distintos gráficos

definidos, etc...; esto obliga a que las teclas tengan funciones múltiples. La manera de acceder a las diferentes funciones que posee una tecla viene determinada por el color y la posición que ésta ocupa dentro o fuera de la tecla; también depende del modo en que se encuentre el cursor parpadeante que aparece en la parte inferior de la pantalla.

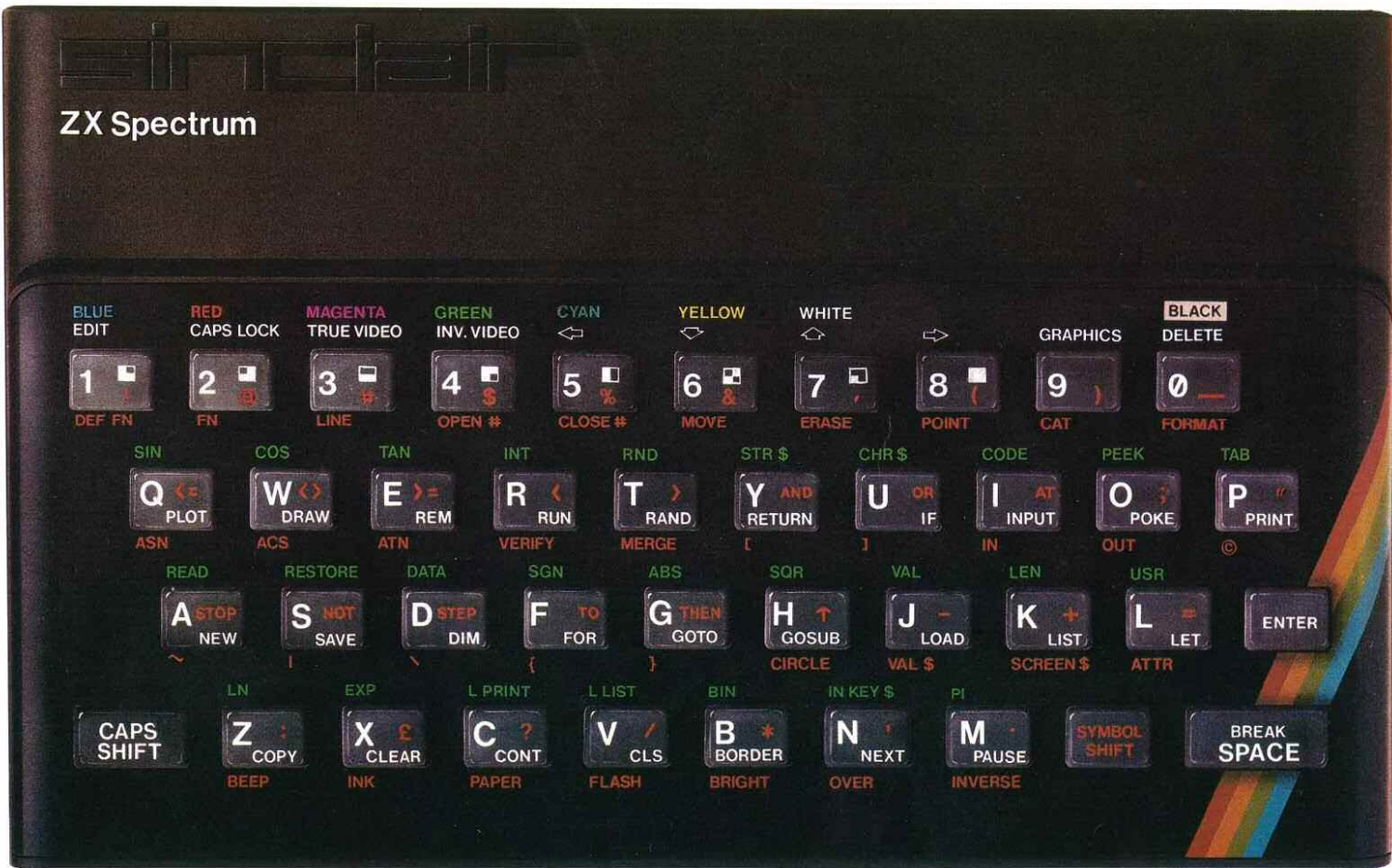
MODOS **K**.—De entrada, al conectar el ordenador, según indica en el capítulo primero del manual de introducción que acompaña a su Spectrum, aparece el mensaje «© 1982 Sinclair Research Ltd»; este mensaje desaparece al presionar la tecla ENTER y en su lugar aparece el cursor parpadeante indicando el modo **K**; en este modo se tiene acceso a los números situados en la fila de teclas superior o a las sentencias pintadas de blanco, dentro de las tres filas de teclas restantes.



En el Spectrum, las teclas son multifunción. Cada una de ellas tiene diversas utilidades, por lo que es imprescindible un correcto manejo de los «modos».

sinclair

ZX Spectrum



Tomemos, por ejemplo, la tecla correspondiente a la letra R; en modo **K** tendríamos acceso a la sentencia RUN.

MODO L.—Una vez visualizada la palabra RUN en la pantalla, el cursor cambia a modo **L**; con lo que se tiene acceso a la letra dibujada en blanco dentro de la tecla, o pulsando simultáneamente la tecla SYMBOL SHIFT y la tecla correspondiente tendríamos acceso a la sentencia o signo dibujado en rojo, dentro de la tecla, en la parte superior derecha. Tomando el mismo ejemplo y trabajando en modo **L** la tecla R, aparecería en la pantalla la letra r minúscula; si quisiéramos que apareciera como R mayúscula tendríamos que utilizar simultáneamente, como en una máquina de escribir, CAPS SHIFT y la tecla R. Pulsando simultáneamente SYMBOL SHIFT y la tecla R se visualizaría en la pantalla el signo < (menor que).

MODO C.—El modo **C** es una variante del **L**, ya que permite escribir siempre en mayúsculas sin tener que estar pulsando continuamente la tecla CAPS SHIFT; para acceder a este modo basta con pulsar simultáneamente CAPS SHIFT y CAPS LOCK, situado en la tecla correspondiente al número 2. A partir de este momento, el modo **C** sustituye al modo **L**. Presione la tecla R y comprobará que la R mayúscula aparecerá en lugar de la r minúscula.

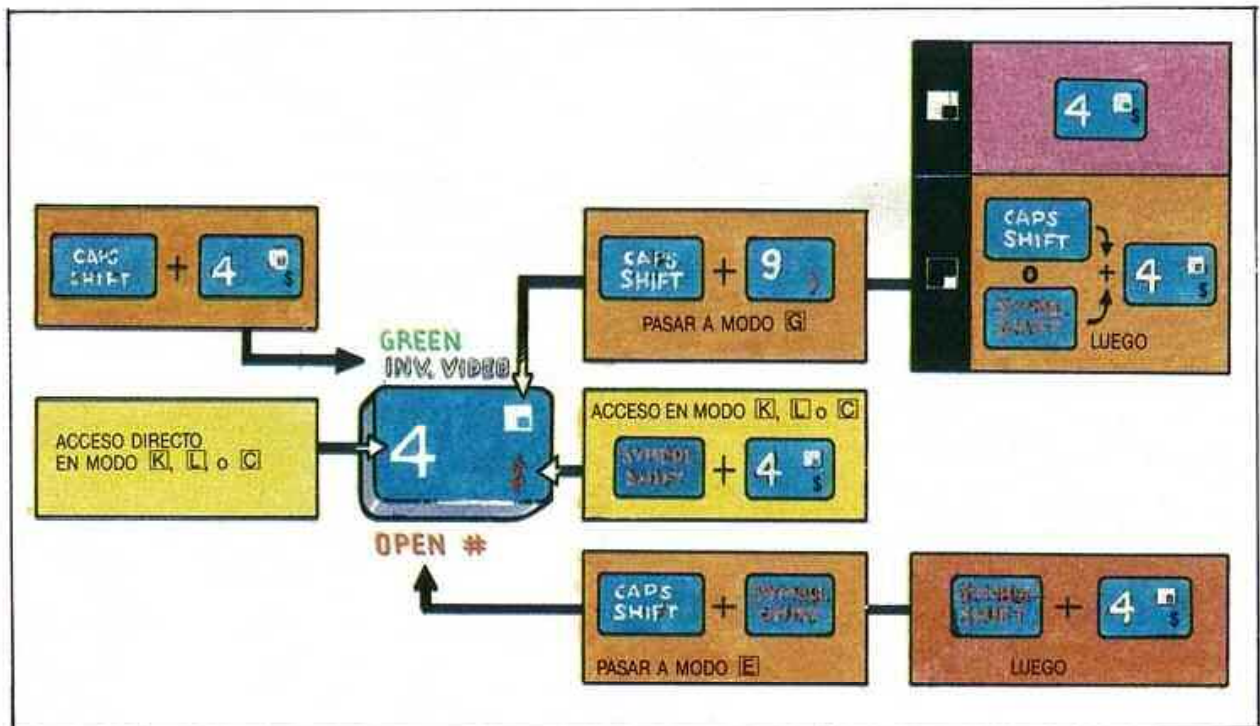
Si desea volver otra vez al modo **L**, sólo hay que repetir de nuevo la secuencia CAPS SHIFT + CAPS LOCK y de nuevo aparecerá el cursor en modo **L**. El modo que siempre presenta el Spectrum, si no se le indica lo contrario, es el **L** (minúsculas).

MODO E.—Para poder acceder a las sentencias situa-

das en la parte exterior de la tecla, en color verde las de la parte superior y en color rojo las de la inferior, es necesario pasar al modo **E**.

Al modo **E** se accede pulsando simultáneamente las teclas CAPS SHIFT y SYMBOL SHIFT, en este momento el cursor cambia a modo **E**, una vez realizada esta operación tenemos dos opciones: o bien acceder a las sentencias pintadas en verde, o bien a las pintadas en rojo. Para seleccionar las primeras basta con usar la tecla correspondiente una vez que estemos en modo **E**; para acceder a las segundas es necesario, además de estar en **E**, pulsar simultáneamente SYMBOL SHIFT o CAPS SHIFT y la tecla correspondiente.

En el ejemplo anterior, si se pasa a modo **E** y se pulsa la tecla R, se visualizaría en la pantalla de su televisor o mo-



Las teclas de la fila superior tienen un tratamiento distinto, ya que contienen los gráficos predefinidos y una serie de funciones de control.

6 MICROBASIC

nitro la sentencia INT; pero si, estando en modo **E**, se accionan simultáneamente las teclas SYMBOL SHIFT o CAPS SHIFT y R, aparecerá en la pantalla el comando VERIFY. En ambos casos se observará que después de visualizarse la sentencia correspondiente (INT o VERIFY), el cursor cambia a modo **L** o **C**.

Si, por error, al introducir una sentencia que no corresponde al modo **E** se ha pasado a éste, para volver al modo anterior bastaría con repetir la secuencia CAPS SHIFT + SYMBOL SHIFT.

Las teclas de la fila superior tienen un tratamiento distinto, ya que contienen los gráficos predefinidos en el Spectrum y una serie de funciones de control...

Las funciones de control se encuentran fuera de las teclas pintadas de blanco y en la parte superior. Estas funciones, como por ejemplo EDIT, DELETE, desplazamientos del cursor, etc..., ya irán siendo tratadas en los siguientes párrafos; ahora lo más importante es conocer como acceder a ellas; para ello basta con pulsar simultáneamente las teclas CAPS SHIFT y la tecla correspondiente a la función deseada; es decir, si quisiéramos acceder a la función EDIT bastaría con pulsar, a la vez las teclas CAPS SHIFT y la correspondiente al número 1.

MODOS G.—Para acceder a los gráficos situados en las teclas con los números 1 a 8 y a los definidos por el usuario, situados en las teclas con las letras de la A a la U, hay que pasar al modo **G**. Para

acceder a **G** es necesario hacer uso de la función GRAPHICS; que como ya conocen, se accede oprimiendo simultáneamente la tecla CAPS SHIFT y la que lleva el número 9. Una vez que la operación anterior ha sido realizada, el cursor parpadeante pasa a modo **G**.

Los símbolos predefinidos están representados en la esquina superior derecha de las teclas 1 a 8; para visualizar cualquiera de estos símbolos en la pantalla sólo es necesario pulsar la tecla correspondiente al gráfico elegido. También se pueden visualizar estos gráficos en negativo; es decir, que las partes claras se conviertan en oscuras y las oscuras en claras; para ello basta con accionar, a la vez que se selecciona un gráfico, la tecla CAPS SHIFT o SYMBOL SHIFT.

Para visualizar los gráficos definidos por el usuario es necesario que estos estén ya definidos con anterioridad; si algún lector no sabe todavía cómo definir sus propios gráficos, no se preocupe; en otro artículo de este curso se tratará este tema, y en otras secciones de «Microhobby» también se comentará. Si los gráficos ya estuvieran definidos, bastaría para visualizarlos con pulsar la tecla correspondiente a la que se le asignó; es decir, si definimos un gráfico y lo asignamos a la tecla con la letra R, para visualizarlo habría que pasar a modo **G** y pulsar simplemente la tecla R.

Aquellas letras entre la A y la U que no tengan un gráfico asignado, al pasar a modo **G** y seleccionarlas, se visualizaría en la pantalla la letra correspondiente a la letra pulsa-

da, escrita en mayúsculas, aunque el modo elegido anteriormente fuera el **L**.

Para poder volver al modo anterior basta con pulsar la tecla GRAPHICS, situada en el número 9, y el cursor volverá a parpadear en el modo anterior. Desde el modo **G** también se puede pasar directamente al modo **E**, pulsando simultáneamente CAPS SHIFT y SYMBOL SHIFT.

Para acceder a los signos en rojo, situados en la esquina inferior derecha, basta con pulsar simultáneamente SYMBOL SHIFT y la tecla correspondiente.

A los números del \emptyset al 9 se puede acceder tanto en modo **K** como en **L** o **C**.

Las sentencias pintadas en rojo y situadas en la parte inferior de las teclas no tienen el mismo tratamiento que sus homólogas de las restantes filas de teclas, ya que para acceder a éstas sólo es posible hacerlo cuando estemos en modo **E**, pulsando SYMBOL SHIFT más la tecla correspondiente; sin embargo, en las restantes filas de teclas podría ser o bien con CAPS SHIFT o con SYMBOL SHIFT.

Las funciones de color, situadas en la parte exterior de las teclas \emptyset a 7, indican la correspondencia que hay entre colores y números, ya que cuando en algún comando haya que indicar una función de color, ésta será dada por su número; es decir, si en una sentencia de visualización de pantalla hay que indicar que ésta sea en color rojo (RED), habrá que seleccionar la tecla con el número 2.

GUIA DE LOCALIZACION DE SENTENCIAS, FUNCIONES Y SIMBOLOS

Esta guía facilita la búsqueda de sentencias, funciones o símbolos, cuando aún no se tiene la suficiente destreza de manejo o cuando se produce alguna duda. Las sentencias están ordenadas por orden alfabético; para una mejor localización, a la izquierda de la sentencia aparece la inicial de ésta, y, a la derecha, la tecla donde está situada.

| | | |
|------|-----|---|
| A | ABS | G |
| | ACS | W |
| | AND | Y |
| | ASN | Q |
| | AT | I |
| | ATN | E |
| ATTR | L | |

| | | |
|---|-------|---|
| M | MERGE | T |
| | MOVE | 6 |

| | | |
|---|------|---|
| N | NEW | A |
| | NEXT | N |
| | NOT | S |

| | | |
|---|--------|---|
| B | BEEP | Z |
| | BIN | B |
| | BORDER | B |
| | BRIGHT | B |

| | | |
|---|--------|---|
| O | OPEN # | 4 |
| | OR | U |
| | OUT | O |
| | OVER | N |

| | | |
|-----|---------|---|
| C | CAT | 9 |
| | CRHS | U |
| | CIRCLE | H |
| | CLEAR | X |
| | CLOSE # | S |
| | CLS | V |
| | CODE | I |
| | CONT | C |
| | COPY | Z |
| COS | W | |

| | | |
|-------|-------|---|
| P | PAPER | C |
| | PAUSE | M |
| | PEEK | O |
| | PI | M |
| | PLOT | Q |
| | POINT | 8 |
| | POKE | O |
| PRINT | P | |

| FUNCIONES DE CONTROL | |
|----------------------|-------|
| BREAK | SPACE |
| CAPS LOCK | 2 |
| DELETE | 0 |
| EDIT | 1 |
| GRAPHICS | 9 |
| INV. VIDEO | 4 |
| TRUE VIDEO | 3 |

| SIMBOLOS | |
|----------|---|
| @ | 2 |
| ⊙ | P |
| # | 3 |
| \$ | 4 |
| £ | X |
| & | 6 |
| ~ | A |
| | 8 |
|) | 9 |
| | Y |
| | U |
| | F |
| | G |
| | D |
| | V |
| | S |
| | O |
| | 1 |
| | C |
| | M |
| | N |
| | O |
| | Z |
| | P |
| | 7 |
| | K |
| | J |
| | B |
| | L |
| | H |
| | 5 |
| | T |
| | R |
| <> | W |
| >= | E |
| <= | Q |

| | | |
|---|--------|---|
| D | DATA | D |
| | DEF FN | 1 |
| | DIM | D |
| | DRAW | W |

| | | |
|-----|---------|---|
| R | RAND | T |
| | READ | A |
| | REM | E |
| | RESTORE | S |
| | RETURN | Y |
| | RND | T |
| RUN | R | |

| OTRAS FUNCIONES | |
|-----------------|--------|
| CAPS SHIFT | IZDA.Z |
| ENTER | DCHA.L |
| SYMBOL SHIFT | DCHA.M |

| | | |
|---|-------|---|
| E | ERASE | 7 |
| | EXP | X |

| | | |
|--------|-----------|---|
| S | SAVE | S |
| | SCREEN \$ | K |
| | SGN | F |
| | SIN | O |
| | SQR | H |
| | STEP | D |
| | STOP | A |
| STR \$ | Y | |

| DESPLAZAMIENTOS DEL CURSOR | |
|----------------------------|---|
| ↑ | 7 |
| ↓ | 6 |
| ⇒ | 8 |
| ⇐ | 5 |

| | | |
|---|--------|---|
| F | FLASH | V |
| | FN | 2 |
| | FOR | F |
| | FORMAT | 0 |

| | | |
|---|------|---|
| T | TAB | P |
| | TAN | E |
| | THEN | G |
| | TO | F |

| GRAFICOS | | |
|----------|---|---|
| ▣ | ▣ | 1 |
| ▣ | ▣ | 2 |
| ▣ | ▣ | 3 |
| ▣ | ▣ | 4 |
| ▣ | ▣ | 5 |
| ▣ | ▣ | 6 |
| ▣ | ▣ | 7 |
| ▣ | ▣ | 8 |

| | | |
|---|-------|---|
| G | GOSUB | H |
| | GOTO | G |

| | | |
|---|-----|---|
| U | USR | L |
|---|-----|---|

| | | |
|---|----------|---|
| I | IF | U |
| | IN | I |
| | INK | X |
| | INKEY \$ | N |
| | INPUT | I |
| | INT | R |
| | INVERSE | M |

| | | |
|---|--------|---|
| V | VAL | J |
| | VAL \$ | J |
| | VERIFY | R |

| COLORES | |
|---------|---|
| BLACK | 0 |
| BLUE | 1 |
| CYAN | 5 |
| GREEN | 4 |
| MAGENTA | 3 |
| RED | 2 |
| WHITE | 7 |
| YELLOW | 6 |

| | | |
|---|---------|---|
| L | LEN | K |
| | LET | L |
| | LINE | 3 |
| | LIST | K |
| | LLIST | V |
| | LN | Z |
| | LOAD | J |
| | L PRINT | C |

Edición de programas

En el momento de introducir una instrucción en el ordenador se nos presentan dos opciones: que se ejecute nada más ser introducida o que quede almacenada en la memoria del ordenador para su posterior ejecución.

En la primera opción, una vez ejecutada la instrucción, si se desea repetir su ejecución es necesario teclear de nuevo; en el segundo caso no es necesario, ya que la primera vez que se introdujo quedó almacenada en la memoria del ordenador y podemos ejecutarla tantas veces como queramos, siempre y cuando no desconectemos la clavija de alimentación de 9V DC.

Para introducir una *instrucción* de ejecución inmediata, sólo es necesario teclearla y pulsar la tecla ENTER, que indica el ordenador que la *instrucción* se ha terminado de teclear; en ese momento el ordenador la analiza para que no tenga ningún error de sintaxis; es decir, que el argumento está correctamente tecleado y que está en concordia con el tipo de sentencia. Si todo ha sido correcto, la *instrucción* se ejecuta inmediatamente.

Para introducir una *instrucción* que no se ejecute inmediatamente, es necesario asignarle un número de línea comprendido entre 1 y 9999; este número se teclea estando el cursor en modo **K**; a continuación se teclea la *instrucción* y posteriormente, como en el caso anterior, se pulsa la tecla ENTER. Se analizan los posibles errores que la *instrucción* podría contener y si todo es correcto, no se ejecuta, sino que pasa a memoria; esto se puede comprobar, ya que

se visualiza en la parte superior de la pantalla.

Para poder ejecutar esa *instrucción*, es necesario ejecutar con anterioridad la sentencia inmediata RUN; para ello basta con pulsar las teclas R y ENTER. Como hemos dicho anteriormente, esa *instrucción* se puede repetir, tantas veces como queramos, simplemente introduciendo la sentencia RUN y pulsando ENTER.

Veamos unos ejemplos; pruebe a introducir la siguiente *instrucción* inmediata:

```
PRINT "Curso BASIC/Spectrum"
```

al terminar de teclear y pulsar ENTER se ejecutará la *instrucción* y aparecerá en la parte superior de la pantalla, la cadena alfanumérica: Curso BASIC/Spectrum, y en la parte inferior un mensaje que envía el ordenador, indicando que la *instrucción* ha sido ejecutada correctamente, "Ø OK, Ø : 1". Si quiere repetir la ejecución, es necesario repetir también la *instrucción*.

Para que ésta no se ejecute hasta que usted quiera, debe de asignarle un número de línea, por ejemplo el 1Ø; por tanto teclee:

```
1Ø PRINT "Curso BASIC/Spectrum"
```

al pulsar ENTER, comprobará que la *instrucción* tecleada pasa a la parte superior de la pantalla; también observará que después del número de línea 1Ø, el ordenador ha colocado el símbolo >; éste es un indicador de presencia, conocido también como "prompt" que indica cuál es la última línea editada.

Para poder ejecutar esa *instrucción*, deberá introducir

RUN, y como siempre pulsar ENTER. El listado de la *instrucción* se borrará, y ésta se ejecutará de la misma manera que si hubiese sido introducida directamente. Aunque el listado haya desaparecido de la pantalla, no se preocupe: está almacenado en la memoria del ordenador para poder visualizarlo de nuevo, simplemente pulse la tecla ENTER. Podrá volver a ejecutar la *instrucción*, sin tener que teclearla otra vez, introduciendo la sentencia RUN y pulsando ENTER.

El numerar una sola *instrucción*, en principio, no tiene mucha utilidad, lo que sí tiene y bastante, es numerar una cantidad más o menos larga de *instrucciones*; esto es lo que se conoce como *programa*. Un programa es, por tanto, una secuencia de *instrucciones* ordenadas que realizan una función determinada.

Un ejemplo de estructura de un programa puede ser el siguiente:

| | | |
|-----|-------------|--------|
| 1Ø | Instrucción | n.º 1 |
| 2Ø | Instrucción | n.º 2 |
| 3Ø | Instrucción | n.º 3 |
| 1ØØ | Instrucción | n.º 10 |

En el lenguaje BASIC, las *instrucciones* se suelen numerar de diez en diez; esto se debe a que muy pocas veces un programa funciona a la primera; siempre es necesario modificar, borrar o añadir alguna *instrucción* que se nos quedó olvidada en el tintero. Si las tenemos numeradas a intervalos de diez, siempre tenemos la posibilidad de insertar una *instrucción*, asignándole un número de línea intermedio, ya que el ordenador, cuando eje-

cuta el programa que tiene en memoria, siempre lo hace empezando por la instrucción con el número de línea más bajo y continúa en orden creciente.

En el ejemplo anterior, si queremos insertar una instrucción entre las líneas 1 ϕ y 2 ϕ , le asignaríamos, por ejemplo, el número 15, de manera que el programa quedaría de la siguiente forma:

| | | | |
|-----------|-------------|-----|----|
| 1 ϕ | Instrucción | n.º | 1 |
| 15 | Instrucción | n.º | 2 |
| 2 ϕ | Instrucción | n.º | 3 |
| 3 ϕ | Instrucción | n.º | 4 |
| 10 ϕ | Instrucción | n.º | 11 |

En el Spectrum, los números de línea tienen que estar comprendidos entre el 1 y el 9999; cualquier instrucción que se asigne con una numeración distinta, será rechazada por el ordenador. En el caso de que se asigne a una instrucción un número de línea igual a ϕ , la pantalla se limpiará de caracteres, y en la zona reservada para los informes del ordenador, aparecerá el mensaje: C Nonsense in BASIC, ϕ : 1. Para volver a recuperar el listado, si lo hubiere, pulse ENTER. Si el número de línea fuese superior a 9999 al pulsar ENTER aparecerá una ϕ parpadeante a la derecha del número de línea; en el capítulo denominado "Corrección de errores", se dan orientaciones necesarias para poder corregir este fallo.

Se habrá dado cuenta que, según se van introduciendo las instrucciones, el prompt > se va desplazando, y siempre apunta a la última línea editada.

Dentro de una misma línea se pueden introducir varias

10 MICROBASIC



Ejemplo de edición de un programa.

```

10 REM *****
   *          *
   *  CURSO BASIC  *
   *          *
   *   EJER: 1     *
   *          *
   * *****      *

LS 50 BORDER 1: PAPER 7: INK 2: C
60 GO SUB 1000
90 REM
   *          *
   *  REJILLA    *
   *          *
   * *****      *

100 FOR n=8 TO 248 STEP 8
110 PLOT n,175
120 DRAW 0,-175
130 BEEP 0.05,n/8
140 NEXT n
150 FOR n=167 TO 7 STEP -8
160 PLOT 0,n
170 DRAW 255,0
180 BEEP 0.05,n/8
190 NEXT n
200 REM

```

```
*****
* MENSAJE VERTICAL *
*****
```

```
210 RESTORE 270
220 FOR n=1 TO 88
230 READ y: READ x
240 PRINT AT y,x;CHR$ 20+CHR$ 1
;CHR$ 32
250 BEEP 0.05,y
260 NEXT n
270 DATA 3,2,4,2,5,2,6,2,7,2,3,
5,7,5
280 DATA 2,8,3,8,4,8,5,8,6,8,7,
8,2,11,3,11,4,11,5,11,6,11,7,11
290 DATA 2,14,3,14,4,14,5,14,6,
14,7,14,8,14,6,16,7,17,3,18,4,18
,8,18
300 DATA 3,21,4,21,7,21,3,24,6,
24,7,24
310 DATA 3,27,4,27,5,27,6,27,7,
27,3,30,4,30,5,30,6,30,7,30
320 DATA 11,3,12,3,13,3,14,3,15
,3,16,3,17,3,12,6,13,6,15,6,16,6
330 DATA 12,9,13,9,14,9,15,9,16
,9,17,9,12,12,13,12,14,12,15,12,
16,12,17,12
340 DATA 12,15,13,15,16,15,12,1
8,15,18,16,18
350 DATA 12,22,13,22,14,22,15,2
2,16,22
360 DATA 12,26,13,26,14,26,15,2
6,16,26,12,29,16,29
400 REM
```

instrucciones; éstas tienen que estar separadas por el signo :. Utilizando el separador, el programa ocupa menos memoria y la ejecución de éste se hace más rápida; sin embargo, tiene la desventaja de que a veces los listados de los programas no quedan lo suficientemente claros, como para poder interpretarlos rápidamente.

Un ejemplo de utilización del separador, podría ser el siguiente:

| | |
|-----|-----------------------|
| 1Ø | Instrucción n.º 1 |
| 2Ø | Instrucción n.º 2:3:4 |
| 3Ø | Instrucción n.º 5 |
| 1ØØ | Instrucción n.º 12 |

Corrección de errores

Mientras efectúa la laboriosa tarea de edición de un programa, seguro que alguna vez se habrá olvidado de introducir alguna instrucción: un número de línea que no correspondía o bien se habrá encontrado que en el momento de insertarla en memoria no se podía, ya que aparecía una **?** parpadeante, indicando un error de sintaxis. En este capítulo, se va a tratar de aclarar las posibles dudas que se tengan sobre borrado, modificación o inserción de nuevas líneas.

Para hablar de las posibles correcciones a efectuar en

nuestro programa, vamos a distinguir si las instrucciones están insertadas en memoria o no, ya que su tratamiento depende de este detalle.

Para modificar una instrucción que estamos tecleando, es necesario hacer uso de varias de las funciones situadas en la fila superior de teclas; éstas son: los desplazamientos del cursor, izquierda (←) y derecha (→), y la función DELETE. Los desplazamientos del cursor no necesitan casi explicación, ya que como su propio nombre indica, desplazan el cursor parpadeante en la dirección que indica la flecha, y la función DELETE borra el carácter situado a la izquierda del cursor. Estas funciones se seleccionan apretando la tecla CAPS SHIFT simultáneamente con la de función correspondiente; al igual que ocurre con cualquier otra tecla, si se mantienen apretadas ambas durante cierto tiempo, la función se repite. En el modo **G**, para seleccionar la función DELETE no se necesita pulsar CAPS SHIFT; simplemente apretando la tecla con el número Ø se borra el carácter anterior al cursor.

Una vez explicadas las funciones, para poder realizar una modificación, es necesario desplazar el cursor a la izquierda, hasta situarlo en el lugar que queramos; ya en él podemos, o bien insertar algún carácter que se nos olvidó o bien hacer uso de la función DELETE. Si quisiéramos seguir escribiendo al final de la línea será necesario ir desplazando el cursor hacia la derecha, hasta llegar a él.

En caso de que apareciera una **?** parpadeante al inten-

tar insertar una instrucción, la secuencia de operación sería la misma; la interrogación nos facilita la búsqueda del error, ya que se coloca al lado de él.

Para borrar una línea completa que estemos tecleando podríamos, haciendo uso de la función DELETE, ir borrando carácter por carácter, pero cuando la línea es larga resulta más cómodo operar de otra manera: se accede a la función EDIT; para ello basta accionar simultáneamente la tecla CAPS SHIFT y la corres-

pondiente al número 1; en ese momento nos desaparece la línea que queríamos borrar, y en su lugar aparece una copia de la última línea editada; es decir, la que en el listado contenía el prompt >. Para reestablecer esta línea a la memoria, basta con pulsar ENTER.

Una vez que las sentencias están en memoria, para insertar una nueva simplemente basta, como ya dijimos en el capítulo "Edición de programas", asignarle una numeración intermedia; es decir, si queremos insertar dos nuevas

instrucciones entre las de líneas 110 y 120, podríamos asignarles los números de línea 113 y 117; de esta manera, el ordenador las ejecutará después de la 110 y antes de la 120.

Para borrar una instrucción, basta con teclear su número de línea que tenía antes; este método es cómodo cuando las instrucciones son pequeñas, pero cuando, por el contrario, son largas, es más cómodo hacer uso de la función EDIT. Cuando se hace uso de esta función, una copia de la línea



Aspecto del teclado profesional del ZX-Spectrum +, incluyendo 18 nuevas teclas.

12 MICROBASIC

que contenga el prompt > aparece en la zona inferior de la pantalla; para poder desplazar el prompt hasta la instrucción que deseamos modificar, es necesario hacer uso de las funciones de desplazamiento del cursor, arriba (↑) y abajo (↓), situadas en las teclas con los números 7 y 6 respectivamente; estas funciones, al igual que EDIT, se realizan con ayuda de la tecla CAPS SHIFT.

Una vez que la línea a modificar está en la parte inferior, ésta se rectifica de la misma manera que cuando todavía

no está en memoria, es decir, con los desplazamientos de cursor derecha e izquierda y con la función DELETE; cuando la línea está corregida se pulsa ENTER. Si se usa la función EDIT para modificar un número de línea, al pulsar ENTER, no desaparece la instrucción con la numeración antigua, sino que se mantienen ambas en memoria, como se puede constatar por la información que se visualiza en pantalla. Esto puede servir para copiar una instrucción tantas veces como queramos,

con sólo darle una numeración de línea distinta; si, por el contrario, lo que deseábamos era cambiar el número de instrucción, tendremos que borrar la instrucción correspondiente a la numeración antigua; como hemos explicado anteriormente, esto se hace tecleando el número de línea y pulsando ENTER.

Existe otro método para modificar una línea sin tener que estar desplazando el prompt de línea en línea, este método lo explicaremos cuando veamos la sentencia LIST.



Ejercicio

Como comprobación de que ha entendido lo explicado hasta este momento, intente editar el programa que a continuación le proponemos; que combine los semigráficos con sonido y color.

No se preocupe si no entiende el significado de las sentencias o la filosofía del programa; ya que el objeto de éste es practicar el acceso al teclado, la edición y la corrección de los errores que pudieran surgir.

El programa que proponemos es un poco enigmático, hasta que no lo ejecute no sabrá el contenido del mensaje que aparece en pantalla. Aquellos lectores que entiendan algo de BASIC, y les guste lo misterioso, pueden intentar descifrar este mensaje, ya que la clave está en las sentencias DATA.

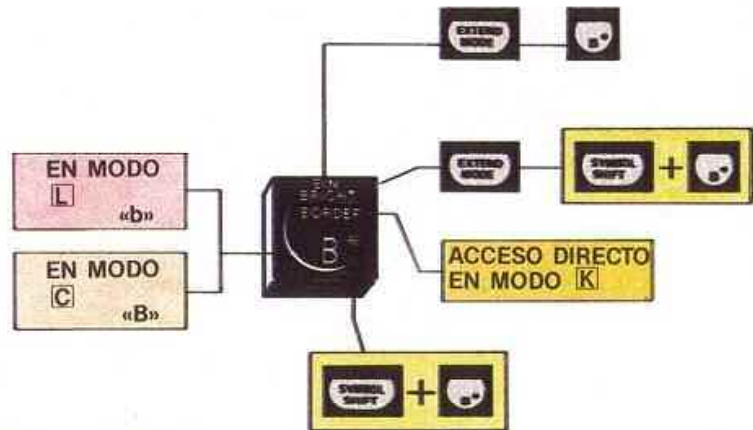
Es fácil de teclear, aunque hay que tener cuidado al pulsar los argumentos de las sentencias DATA, ya que si algún dato estuviese cambiado o faltase, el mensaje se vería desvirtuado. Cuando termine de teclear las 78 líneas de que se compone el programa, podrá ejecutarlo, pulsando RUN y ENTER.

Si todo es correcto, cuando termine de ejecutarse el programa, el ordenador enviará el siguiente mensaje:

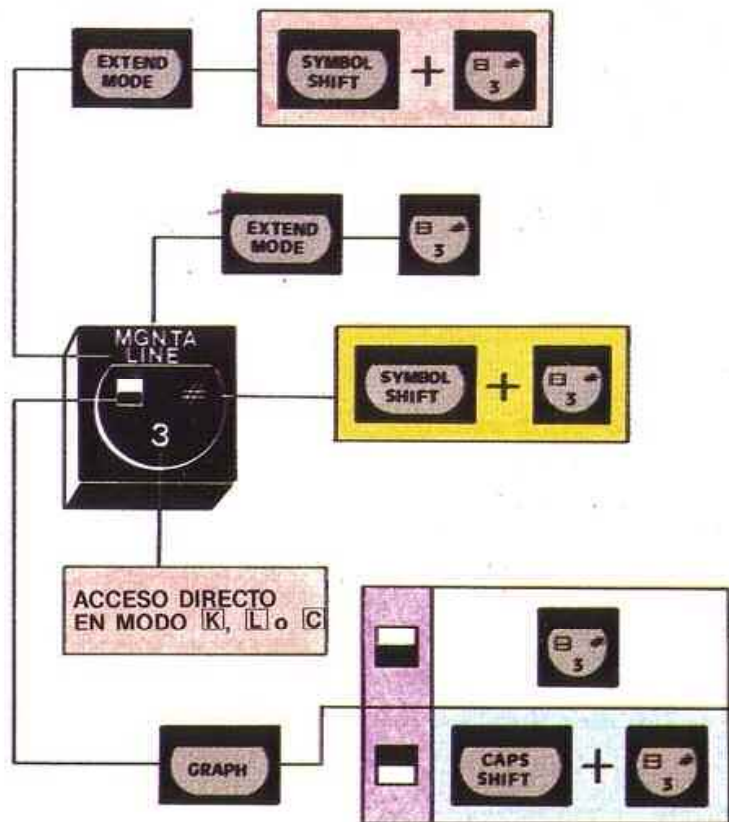
9 STOP statement, 999 : 1
si el ordenador enviara otro distinto, revise las instrucciones o los números de línea, ya que seguramente habrá cometido algún error. Si tiene dudas, no sienta reparo en volver a repasar los capítulos anteriores.

Esperamos que no tengan

14 MICROBASIC



Descripción del manejo de los modos en el nuevo teclado.



La fila superior tiene un tratamiento ligeramente distinto.

ningún tipo de dificultad y que el programa lo editen y ejecuten correctamente.

Teclado del «ZX Spectrum +»

El «ZX Spectrum +», también conocido como «Spectrum Plus», presenta algunas diferencias con respecto a su

homólogo. Aparte de su aspecto exterior, que evidentemente es distinto, se percibe que existe una diferencia en cuanto al número de teclas, ya que éste tiene 58 en lugar de 40. En la tecla adjunta se observa cuáles son las nuevas teclas incorporadas y en la figura representativa del tecla-

do su localización. Las demás teclas, a excepción del SPACE, mantienen su situación y funcionamiento anterior. La tecla «SPACE» ha sido sustituida, al igual que en otros ordenadores, por una **barra espaciadora** semejante a la que incorporan las máquinas de escribir.

Otra de las diferencias del teclado, es la duplicidad de las teclas «CAPS SHIFT» y «SYMBOL SHIFT», esto proporciona un mejor manejo de estas funciones, ya que al estar situadas a ambos lados del teclado podrán ser utilizadas con cualquiera de las dos manos.

Modos **L** y **C**

Después de haber pulsado una tecla, el modo que presenta el «Spectrum +» por defecto es el **L** (Letter Mode), de este modo al pulsar una tecla, la letra inscrita en su interior se visualiza en minúscula; si desea que aparezca en mayúscula sin tener que cambiar de modo, es necesario pulsar simultáneamente junto con la letra elegida, cualquiera de las teclas «CAPS SHIFT». Cuando el texto a escribir en mayúscula es largo, conviene pasar al modo **C** (Capitals Mode). En el «Spectrum +» para acceder a este modo, basta simplemente con pulsar la tecla «CAPS LOCK»; a partir de ese momento, el modo **C** sustituye al **L** hasta que se pulse de nuevo esta tecla.

Modo **E**

El «Spectrum +» tiene una tecla específica para pasar al modo extendido **E** ésta se denomina «EXTEND MODE». Para seleccionar la sentencia situada en la parte superior de la tecla que en el modelo an-

```

*****
*                               *
* MENSAJE HORIZONTAL          *
*                               *
*****

410 RESTORE 470
420 FOR n=1 TO 48
430 READ y: READ x
440 PRINT AT y,x;CHR$ 20+CHR$ 1
;CHR$ 32
450 BEEP 0.05,x
460 NEXT n
470 DATA 2,3,2,4,2,15,2,16,2,17
,2,22,2,23,2,28,2,29
480 DATA 5,15,5,16,5,17,5,22,5,
23
490 DATA 8,3,8,4,8,9,8,10,8,22,
8,23,8,28,8,29
500 DATA 11,4,11,5,11,10,11,11,
11,16,11,17,11,21,11,22,11,23,11
,27,11,28
510 DATA 14,4,14,5,14,10,14,11,
14,16,14,17
520 DATA 17,4,17,5,17,16,17,17,
17,21,17,22,17,23,17,27,17,28
600 REM

*****
*                               *
* TRAGON                       *
*                               *
*****

610 FOR y=0 TO 21
620 FOR x=0 TO 31
640 IF (x AND y)=0 THEN GO TO 6
60
650 IF SCREEN$(y,x)=CHR$ 32 TH
EN GO TO 670
660 PRINT AT y,x;CHR$ 16+CHR$ 3
;CHR$ 144: BEEP 0.01,20: PRINT A
T y,x;CHR$ 17+CHR$ 4;CHR$ 32: GO
TO 680
670 PRINT AT y,x;CHR$ 16+CHR$ 3
;CHR$ 144: BEEP 0.02,-15: PRINT
AT y,x;CHR$ 20+CHR$ 1;CHR$ 16+CH
R$ 2;CHR$ 32
680 NEXT x
690 NEXT y
700 REM

*****
*                               *
* LOGO                          *
*                               *
*****

710 RESTORE 770
720 FOR x=7 TO 24
730 READ dato

```


terior del Spectrum estaba pintada en verde, se pulsa la tecla «EXTEND MODE» y a continuación, cuando el cursor de modo cambia a **E**, se pulsa la tecla seleccionada.

Para acceder a la setencia inmediatamente inferior, pintada de rojo en el otro modelo, se pulsa «EXTEND MODE» y seguidamente, junto con la tecla seleccionada, cualquiera de las teclas «SYMBOL SHIFT». Cuando termina de visualizarse una sentencia en modo extendido, el cursor parpadeante cambia al modo anterior, **L** o **C**.

Modo **G**

Para pasar al modo **G** (Graphics Mode), el «Spectrum +» tiene la tecla «GRAPH». Pulsando esta tecla se tiene acceso a los semigráficos situados en las teclas con los números «1» a «8» y a los gráficos definidos por usuario, en las teclas con las letras de la «A» a la «V». Para retornar al modo anterior, es necesario pulsar de nuevo la tecla «GRAPH».

Edición de programas

Para la edición de programas y corrección de errores, el «Spectrum +» tiene teclas independientes con la misma funcionalidad que en el modelo anterior. Estas teclas son «EDIT», «DELETE» y los controles de cursor (arriba, abajo, izquierda y derecha), situados en este modelo a los lados de la barra espaciadora.

Con esta disposición se hace más agradable y rápida la edición y corrección de programas, ya que con sólo pulsar una tecla se consigue la función deseada. ■

16 MICROBASIC

```

740 PRINT AT 20,x;CHR$ 17+CHR$
4;CHR$ 16+CHR$ 0;CHR$ dato
750 BEEP 0.05,x
760 NEXT x
770 DATA 145,146,147,148,149,15
0
780 DATA 32,90,88,32,83,112,101
,99,116,114,117,109
800 BORDER 4
999 STOP
1000 REM

```

```

*****
*
*  GRAFICOS  *
*
*****

```

```

1005 RESTORE 1050
1010 FOR d=USR "a" TO USR "g"+7
1020 READ dato
1030 POKE d,dato
1040 NEXT d
1050 DATA 195,36,126,90,126,165,
189,129
1060 DATA 0,0,0,254,128,254,2,25
4
1070 DATA 0,128,0,191,160,160,16
0,160
1080 DATA 0,0,0,191,160,160,160,
191
1090 >DATA 0,32,32,175,32,47,40,1
75
1100 DATA 0,8,0,235,42,234,42,23
4
1110 DATA 0,0,0,248,0,0,0,0
1120 RETURN

```

FUNCIONES DE CONTROL UTILIZADAS EN LA CORRECCION DE ERRORES

| | |
|--------|--|
| EDIT | Permite modificar la línea que contenga el prompt «>». |
| DELETE | Borra el carácter situado a la izda. del cursor. |
| ↑ | Desplaza el prompt hacia la línea superior. |
| ↓ | Desplaza el cursor hacia la línea inferior. |
| → | Desplaza el cursor de edición hacia la derecha. |
| ← | Desplaza el cursor de edición hacia la izda. |

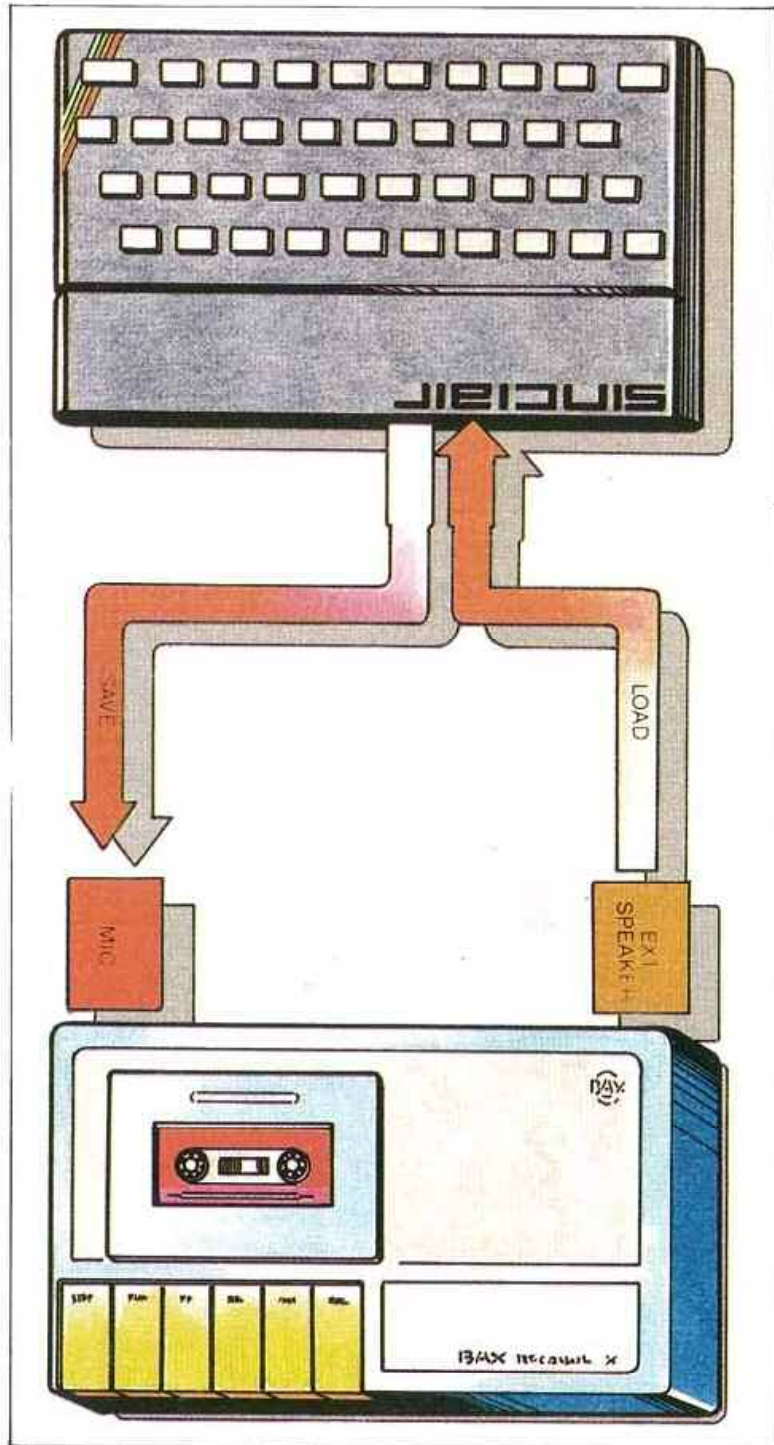
ALMACENAMIENTO DE PROGRAMAS

Mientras no desconecte el ordenador, podrá ejecutar cuantas veces quiera el programa almacenado en memoria, por ejemplo, el propuesto en el capítulo anterior. Si desea volver a ejecutarlo en alguna otra ocasión, no parece lógico volver a teclearlo o tener el ordenador enchufado para que el contenido de la memoria no se pierda; por este motivo, los ordenadores personales tienen algún sistema de almacenamiento de programas. El Spectrum tiene la posibilidad de hacerlo en cintas de *cassette* comerciales, en cartuchos para *Microdrive* o en discos flexibles, también conocidos como *Diskettes*; en esta ocasión sólo vamos a tratar el almacenamiento en cintas de cassette.

Una vez que tengamos editado el programa, y sepamos con certeza que éste se ejecuta correctamente, pasaremos a grabarlo. Antes, es necesario conectar el cassette según se indica en el capítulo 6 del Manual de Instrucciones del Spectrum.

Para efectuar la grabación es necesario hacer uso del comando `SAVE`, que tiene como argumento el nombre que deseemos poner al programa. Debe ir entre comillas y no superar la cantidad de diez caracteres, éstos pueden ser letras, números o símbolos.

Pongamos un ejemplo, que queremos grabar el programa del ejercicio anterior y que de-



Grabación/recuperación.

seamos llamarle "EJER/1", la estructura de la instrucción sería:

SAVE "EJER/1"

para ejecutarla es necesario pulsar la tecla ENTER. Si hubiéramos asignado otro nombre al programa y nos apareciera el mensaje:

F Invalid file name, ϕ :1

significaría que el nombre es más largo de diez caracteres o que lo intentamos almacenar como cadena vacía; es decir:

SAVE ""

Si no aparece el mensaje de error, es porque el nombre asignado es correcto, y en su lugar aparecerá el mensaje:

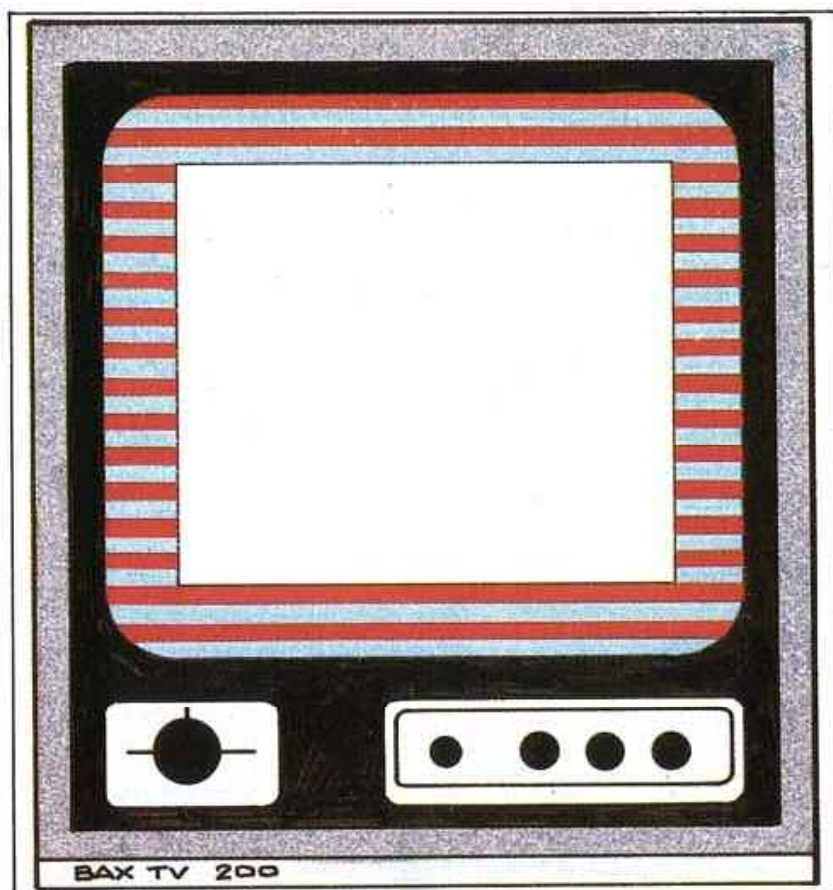
Start tape, then press any key

Este mensaje indica que se ponga el aparato de cassette en posición de grabar, es decir, pulsando las teclas PLAY y RECORD; si la tecla RECORD no entrara, es que la cinta utilizada está protegida contra posibles grabaciones, por tanto, es necesario utilizar otra cinta que no lo esté.

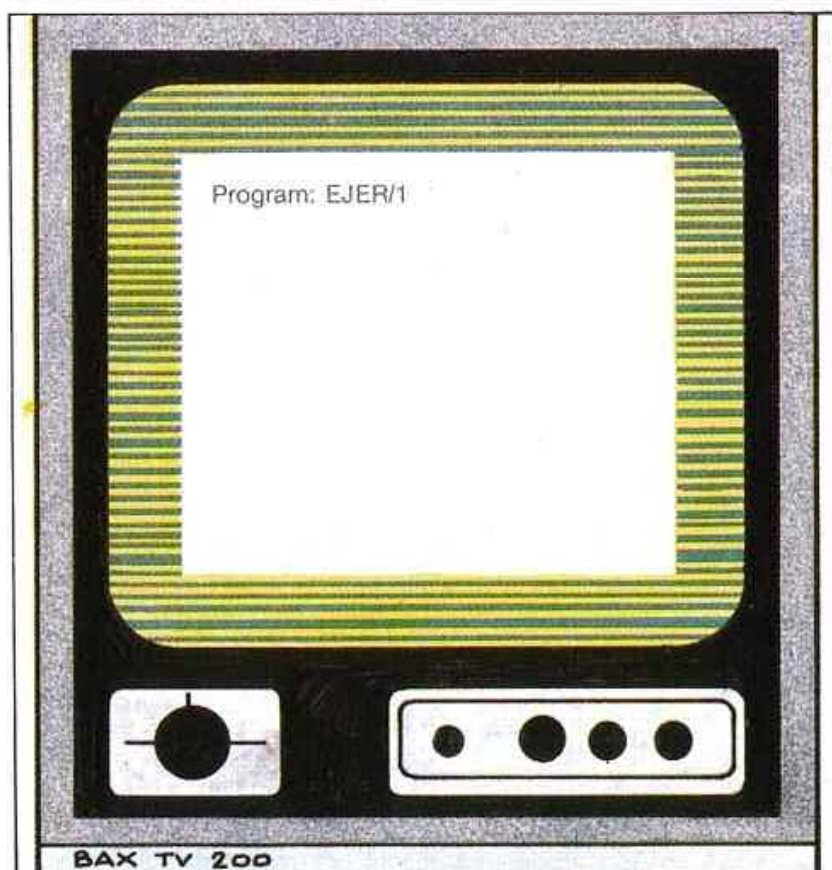
Cuando el aparato esté en marcha, pulsar cualquier tecla del Spectrum, excepto CAPS SHIFT o SYMBOL SHIFT, y la grabación empezará a efectuarse.

Mientras tanto, en el contorno de la pantalla se verán configuraciones de bandas coloreadas horizontales, que se van desplazando. Cuando la grabación termina, el ordenador envía el mensaje ϕ OK, ϕ :1; en ese instante el aparato de cassette debe pararse. Hay

18 MICROBASIC



Señal de preajuste del nivel de grabación.



Señal de «grabación/recuperación».

que poner especial atención en no empezar a grabar en la zona transparente del comienzo de la cinta, ya que en esa zona no se puede grabar.

Verificación

Antes de realizar cualquier otra tarea, es necesario cerciorarse de que el programa ha sido correctamente grabado en la cinta; para ello, se utiliza la sentencia VERIFY. Esta compara lo que hay grabado en la cinta con el contenido de la memoria.

Para verificar el programa, es necesario rebobinar la cinta de cassette hasta un punto anterior al comienzo de la grabación; para este fin es útil contar con un aparato de cassette que disponga de contador.

Utilizando el ejemplo ante-

rior, la estructura de la instrucción sería la siguiente:

```
VERIFY "EJER/1"
```

Al pulsar ENTER, el contorno de la pantalla cambiará de color alternativamente. A partir de este momento, ya se puede poner en marcha el cassette pulsando la tecla PLAY. Cuando la cabecera del programa, es decir el principio, sea encontrado, aparecerá en la pantalla el siguiente mensaje:

```
Program: EJER/1
```

y a continuación se verificará la grabación.

En el contorno de la pantalla se visualizarán las típicas configuraciones de bandas horizontales mencionadas anteriormente. Cuando la verificación ha sido efectuada, el

ordenador nos enviará, si la grabación es correcta, el mensaje "Ø OK, Ø:1".

Si ésta no fuera correcta, pueden ocurrir dos cosas; primera, que la cabecera del programa no sea detectada, y por consiguiente, el ordenador se queda en un bucle sin fin esperando encontrarla. Lo segundo que puede ocurrir, es que la cabecera sea detectada, pero que algún dato del programa grabado no coincida con el contenido de la memoria. En este caso el ordenador nos enviará el mensaje:

```
R Tape loading error, Ø : 1
```

En ambos casos, volver a repetir la secuencia de verificación, y si el error persiste, grabar la cinta de nuevo.

Para salir del bucle sin fin del primer caso, es necesario hacer uso de la función



Almacenamiento en cinta o cassette.



Almacenamiento en Diskette.

BREAK, situada en la tecla espaciadora (SPACE); mantener esta tecla pulsada hasta que aparezca el mensaje:

```
D BREAK-CONT repeats, Ø : 1
```

Una vez que esté grabado y verificado el programa, éste quedará almacenado *permanentemente* en la cinta, siempre y cuando no hagamos otra grabación en la misma zona y mientras la cinta sea conservada en las debidas condiciones.

Recuperación de programas

Si tenemos almacenado algún programa en cinta, pode-

mos volverlo a ejecutar cuando queramos; para ello es necesario copiar la información contenida en la cinta, en la memoria del ordenador; este proceso se conoce como *carga* o *recuperación* de programas.

El comando de carga es el denominado LOAD, éste, como es lógico, tiene por argumento el nombre del programa que deseamos recuperar. El asignar un nombre a un programa, con el comando SAVE, no fue una cosa caprichosa, sino que responde a una necesidad; cuando en una cinta tenemos grabados varios programas, uno a continuación de otro, la única forma de que el ordenador sepa cual debe cargar es

por el nombre que cada uno tiene asignado y que lo diferencia de los demás.

La estructura de la instrucción, siguiendo con el mismo ejemplo, es:

```
LOAD "EJER/1"
```

Antes de poner el cassette en marcha, es necesario rebobinar la cinta hasta un punto anterior al comienzo del programa. Para empezar la carga, se necesita pulsar la tecla ENTER una vez introducido el comando y, a continuación, pulsar la tecla PLAY del aparato de cassette; a partir de este momento, tanto los mensajes de funcionamiento correcto, como los de error, son los mis-

mos que los proporcionados por la sentencia VERIFY.

Tanto con la sentencia VERIFY como con la LOAD, si la cinta se rebobinó hasta un punto lejano del comienzo del programa, al conectar el cassette irán apareciendo en pantalla los nombres de aquellos programas que el ordenador encuentre antes de llegar al especificado en el argumento de la sentencia.

Una vez que el programa está copiado en la memoria del ordenador, el aparato de cassette debe ser parado y la cinta puede retirarse para ser utilizada en otra ocasión.

Hay una variante de la sentencia LOAD, en la que el argumento es una cadena de caracteres vacía; con esta estructura el ordenador carga el primer programa que encuentre, aunque no se le especifique el nombre de éste; el formato es el siguiente:

LOAD ""

En este capítulo hemos explicado las estructuras básicas de las sentencias SAVE, VERIFY y LOAD, en otro posterior, se tratarán con más detalle.

Protección de programas

Cuando una cinta de cassette está grabada con programas definitivos, es conveniente proteger ésta contra posibles regrabaciones accidentales.

Las cintas de cassette disponen en su parte posterior de dos lengüetas, una a cada lado, que sirven para indicar si la cinta es de *lectura/escritura* o por el contrario si es de sólo *lectura*. Cuando las lengüetas están intactas, la cinta permite ser regrabada cuantas veces se quiera, por eso se dice que es de *lectura/escritura*.

ra, en cambio, cuando son arrancadas, la cinta no permite posteriores regrabaciones y por tanto es de sólo *lectura*.

Cuando la lengüeta A es arrancada, los programas de la cara 1 están protegidos contra escritura, si por el contrario es la B serán protegidos los de la cara 2. Para arrancar dichas lengüetas es conveniente la utilización de un destornillador a modo de palanca.

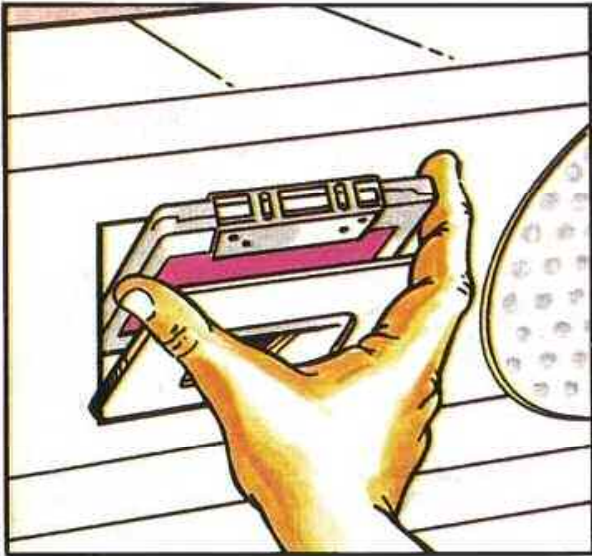
Si una nueva regrabación fuese necesaria, los agujeros donde se encontraban las lengüetas deben ser tapados con cinta adhesiva.

ADVERTENCIA

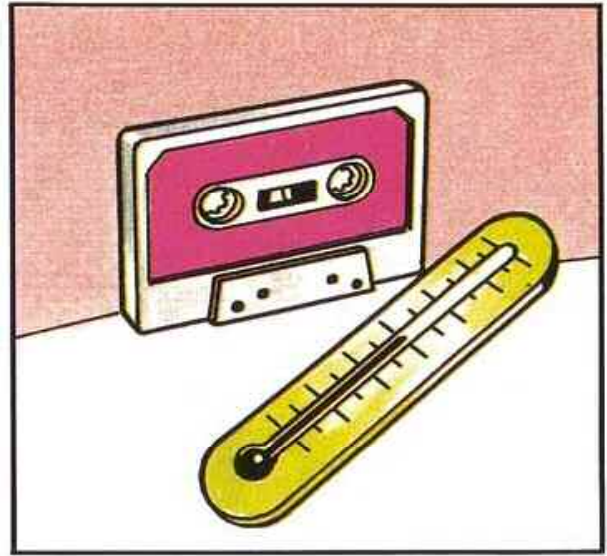
Cuando se usan cintas de cassette protegidas contra escritura, la tecla RECORD de su cassette no permite ser pulsada. No intente forzarla, ya que podría dañar el aparato.



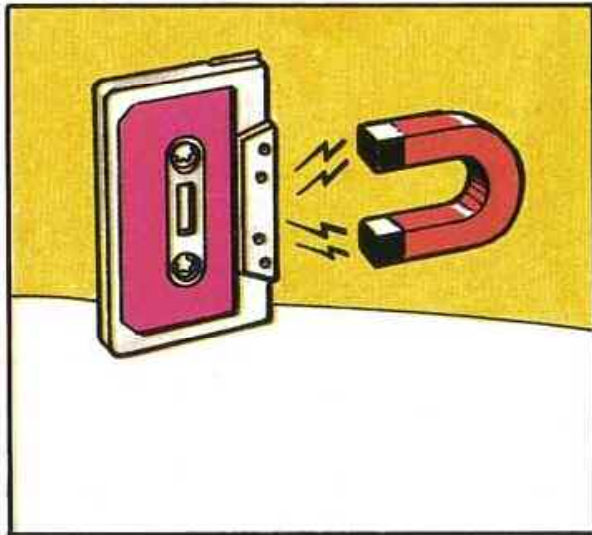
Almacenamiento en cartucho «Microdrive».



Inserte con cuidado.



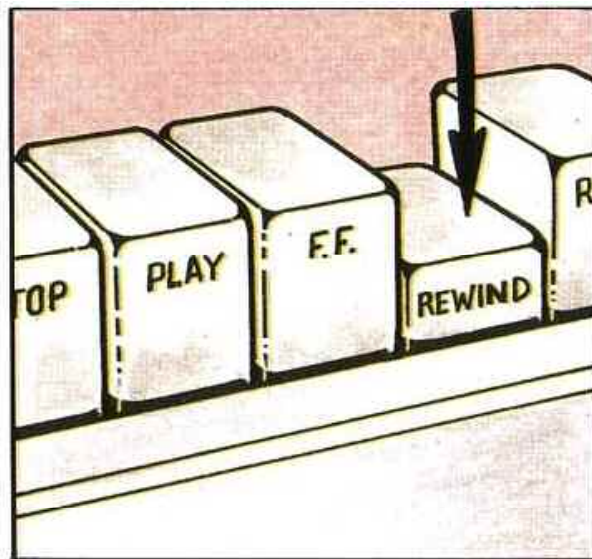
Conserve a 10° C. - 52° C./8%-80% Hr.



Evite campos magnéticos.



No tocar.



Rebobine al final.
22 MICROBASIC

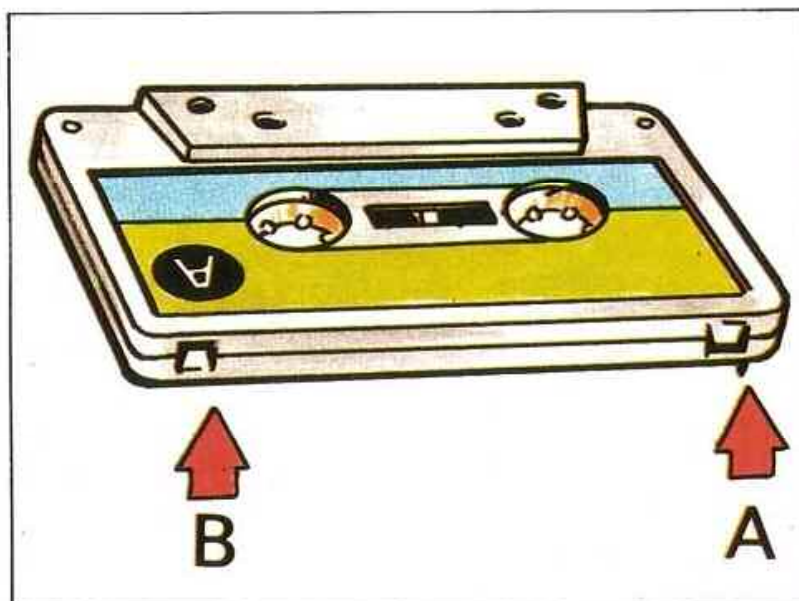


Conserve la cinta en su estuche.

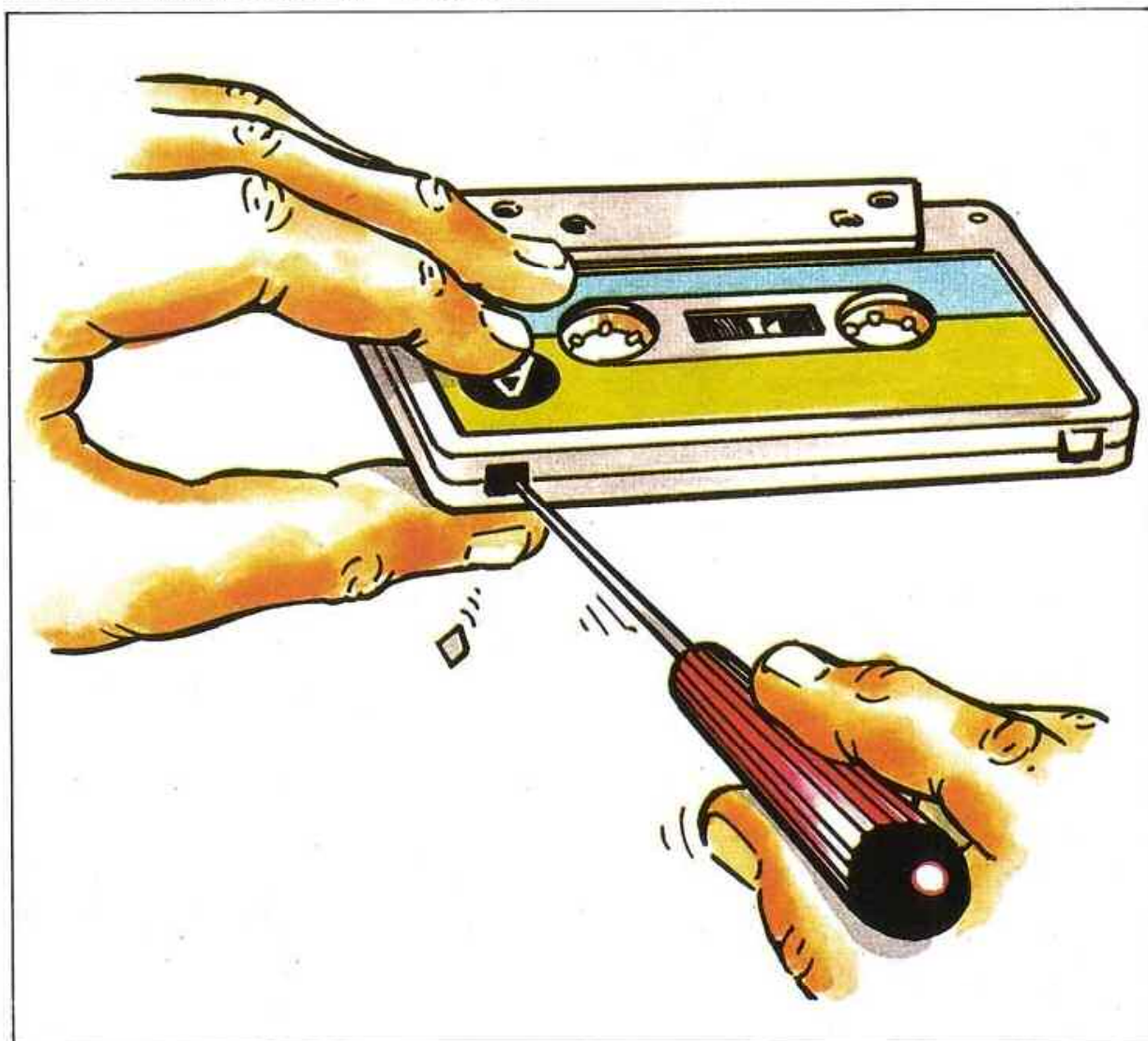
Conservación de cintas

Para no deteriorar los programas grabados, es necesario tener un cuidado especial con las cintas. Para una mejor conservación de éstas, conviene tener en cuenta los siguientes consejos:

El colocar una cinta en las proximidades de un televisor, puede traer graves consecuencias, ya que éstos, internamente, llevan incorporados transformadores y bobinas de alta tensión que generan *campos magnéticos*. Si son lo suficientemente potentes pueden alterar la información contenida



Lengüetas.



Protección de programas.

en la cinta, con la siguiente destrucción de los programas.

Oscilación

La mayoría de los aparatos de cassette comerciales disponen de un *circuito monitor*, que permite al usuario escuchar en el altavoz la grabación que se está efectuando desde

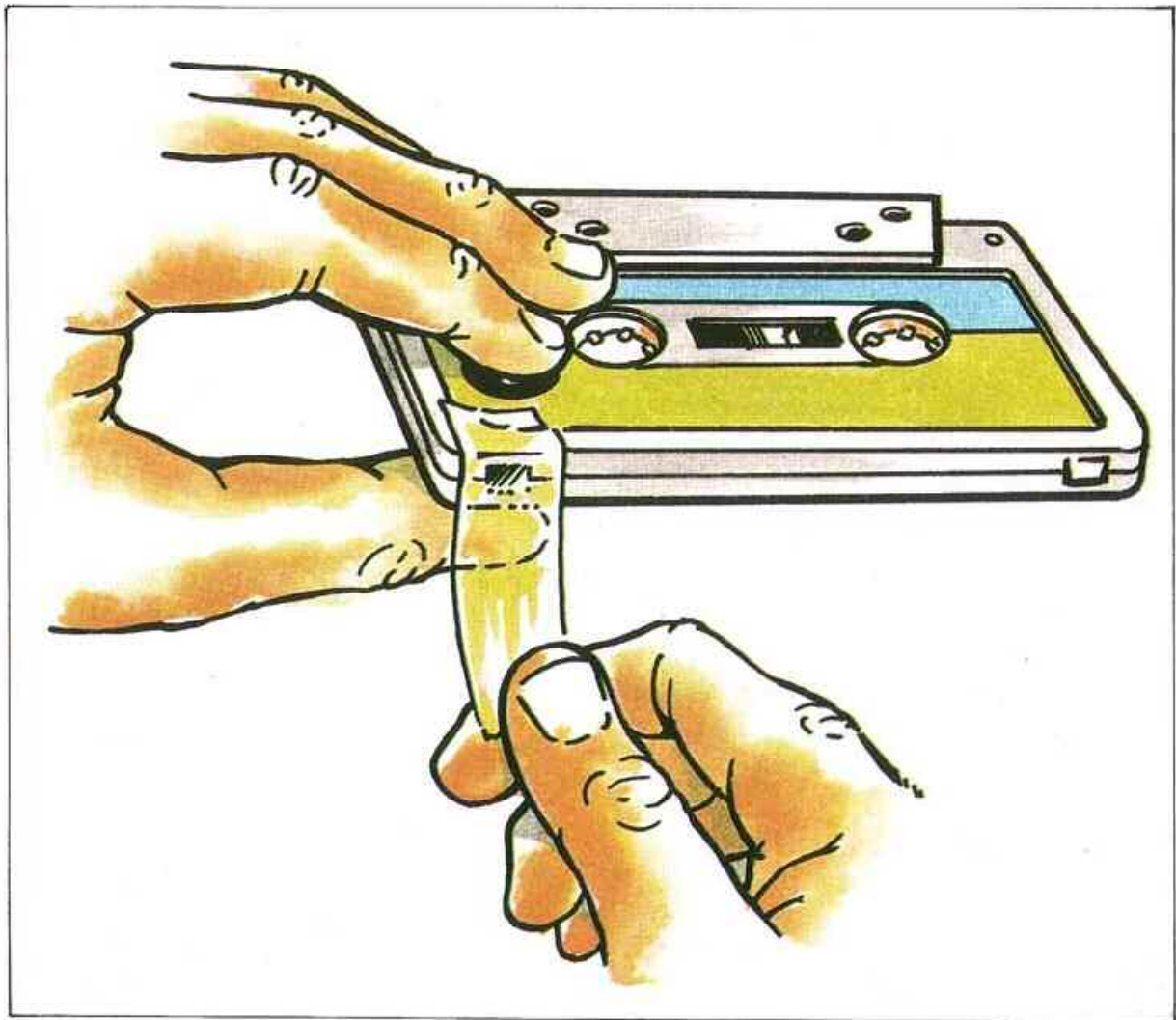
ñal de grabación se verá alterada, impidiendo que el programa pueda ser cargado posteriormente.

La solución consiste en abrir el bucle. Para ello desconecte cualquiera de las dos clavijas que unen la entrada EAR con la salida de auricular o altavoz externo.

té bien grabada, pueden presentarse dos situaciones:

—Que el nivel de grabación de la cinta sea distinto al de reproducción. Para subsanar esta anomalía, basta con aumentar ligeramente el volumen del cassette y proceder a cargar de nuevo.

—Que el ajuste de la cabe-



Tapado de lengüetas.

una radio, desde otro cassette o desde un plato giradiscos. Cuando desea grabar un programa y conecta los dos pares de clavijas (EAR y MIC), la señal de grabación procedente del ordenador puede retornar de nuevo a él debido al circuito monitor. Este bucle formado empezará a oscilar, y la se-

24 MICROBASIC

Ajuste

Si emplea una cinta grabada en un aparato de cassette distinto al que maneja, puede ocurrir que no le sea posible cargar un programa, ya que aparece el correspondiente mensaje de error.

Suponiendo que la cinta es-

za grabadora/reproductora fuera distinto. Para reajustar su cassette, utilice un destornillador del tipo estrella y retoque la altura de la cabeza hasta conseguir que el programa se cargue sin errores. Si desea volver al ajuste original, inserte una cinta grabada en ese cassette y retoque de nuevo.

Constantes y variables

Los datos que el Spectrum procesa son de dos tipos: *constantes y variables*; ambos forman parte del argumento de algunas instrucciones.

Las constantes son datos que, durante la ejecución de un programa, no varían su valor, mientras que las variables pueden tomar distinto valor. La forma de representar una constante es por su valor; sin embargo, las variables se indican con un nombre simbólico.

Los datos también pueden dividirse en *numéricos y string* o *cadena de caracteres*. Podemos tener, por tanto, las siguientes combinaciones:

| DATOS | |
|------------|-----------|
| CONSTANTES | NUMERICAS |
| | STRING |
| VARIABLES | NUMERICAS |
| | STRING |

Veamos unos ejemplos:

a) La fórmula que calcula el área de una superficie esférica es:

$$S = 4 \pi r^2$$

Como constantes numéricas tenemos el número "4", el exponente del radio (r) "2" y el valor de pi (π) "3,1415927"; las variables numéricas están indicadas por los símbolos "r" (radio) y "S" (superficie), que pueden tomar distintos valores durante la ejecución de un programa.

b) El argumento de la instrucción que imprime "El autor de RÍMAS y LEYENDAS es BECQUER", es:

"El autor de " ;L\$;" es " ; N\$

las constantes de cadena, son los string:

```

10 REM "CURSO BASIC/SINCLAIR"
20 REM
25 POKE 23658,8
30 BORDER 2: PAPER 2: INK 7: CLS
40 DRAW 0,175: DRAW 255,0: DRAW
W 0,-175: DRAW -255,0
42 PLOT 48,135: DRAW 0,-15: DR
AW 15,0: DRAW 0,3: DRAW -12,0: D
RAW 0,9: DRAW 12,0: DRAW 0,3: DR
AW -15,0
44 PLOT 72,135: DRAW 0,-15: DR
AW 15,0: DRAW 0,15: DRAW -15,0:
PLOT 75,132: DRAW 0,-9: DRAW 9,0
: DRAW 0,9: DRAW -9,0
46 PLOT 96,135: DRAW 0,-15: DR
AW 15,0: DRAW 0,23: DRAW -3,0: D
RAW 0,-8: DRAW -11,0: PLOT 99,13
2: DRAW 0,-9: DRAW 9,0: DRAW 0,9
: DRAW -9,0
48 PLOT 120,135: DRAW 0,-15: D
RAW 15,0: DRAW 0,3: DRAW -12,0:
DRAW 0,3: DRAW 12,0: DRAW 0,9: D
RAW -15,0: PLOT 123,132: DRAW 0,
-3: DRAW 9,0: DRAW 0,3: DRAW -9,
0
50 PLOT 144,143: DRAW 0,-23: D
RAW 15,0: DRAW 0,15: DRAW -12,0:
DRAW 0,8: DRAW -3,0: PLOT 147,1
32: DRAW 0,-9: DRAW 9,0: DRAW 0,
9: DRAW -9,0
52 PLOT 168,135: DRAW 0,-15: D
RAW 15,0: DRAW 0,3: DRAW -12,0:
DRAW 0,12: DRAW -3,0: PLOT 168,1
43: DRAW 0,-4: DRAW 3,0: DRAW 0,
4: DRAW -3,0
54 PLOT 192,135: DRAW 0,-15: D
RAW 3,0: DRAW 0,12: DRAW 9,0: DR
AW 0,-12: DRAW 3,0: DRAW 0,15: D
RAW -15,0
56 PRINT FLASH 1; AT 12,9; "PARE
LA CINTA"
58 PLOT 0,23: DRAW 255,0
60 PRINT AT 20,1; " ";
62 FOR X=1 TO 28
64 READ A
66 PRINT CHR$ A; : BEEP 0.05,A/2
68 NEXT X
70 DATA 127,32,77,73,67,82,79,
72,79,66,66,89,32,38,32,82,65,70
,65,69,76,32,80,82,65,68,69,83
72 PRINT AT 12,9; "
74 PRINT #0; AT 1,1; "PULSE UNA
TECLA PARA CONTINUAR"
76 PAUSE 0
78 BEEP 0.2,20
100 BORDER 1: PAPER 1: INK 7: C
LS
200 REM
201 REM "PLANTILLA"
202 REM
210 DRAW 0,175: DRAW 255,0: DRA
W 0,-175: DRAW -255,0
212 PLOT 0,151: DRAW 255,0
218 PRINT AT 1,1; " ";

```

```

220 RESTORE
222 FOR X=1 TO 28
224 READ A
226 PRINT CHR$ A;
228 NEXT X
230 PLOT 63,128
240 DRAW 129,0: DRAW 0,-9: DRAW
-129,0: DRAW 0,9
250 PRINT AT 6,12;"BINARIO"
260 PLOT 63,103
270 DRAW 128,0: DRAW 0,-24: DRA
W -128,0: DRAW 0,24
280 FOR X=71 TO 183 STEP 8
290 PLOT X,103: DRAW 0,-24
300 NEXT X
330 PLOT 63,48
340 DRAW 57,0: DRAW 0,-9: DRAW
-57,0: DRAW 0,9
360 PLOT 63,26
370 DRAW 57,0: DRAW 0,-13: DRAW
-57,0: DRAW 0,13
380 PRINT AT 16,8; OVER 1;"DECI
MAL"
470 REM
480 REM "ENTRADA DE DATOS"
490 REM
500 LET N$="0"
510 PRINT #0;AT 1,0; FLASH 1;">
"; FLASH 0;" ";
520 FOR x=1 TO 6
530 PAUSE 0
540 LET A$=INKEY$
550 IF x<>1 THEN GO TO 700
560 IF A$="-" THEN LET S$=A$: P
RINT #0;S$;: GO TO 745
570 IF A$>"0" AND A$<="9" THEN
LET S$=" ": LET N$=A$: PRINT #0;
S$+N$;: GO TO 745
580 GO TO 530
700 IF CODE A$=13 THEN BEEP 0.0
5,20: GO TO 800
710 IF A$>="0" AND A$<="9" THEN
GO TO 730
720 GO TO 530
730 PRINT #0;A$;
740 LET N$=N$+A$
745 BEEP 0.05,20
750 NEXT X
770 REM
780 REM "VERIFICACION"
790 REM
800 LET N=VAL (N$)
810 IF N<0 OR N>65535 THEN PRIN
T #0;AT 1,0;"Numero no valido.":
FOR X=1 TO 200: NEXT X: PRINT #
0;AT 1,0;" ": GO
TO 500
812 REM
814 REM "DECIMAL"
816 REM
818 PRINT #0;AT 1,0;"Espere un
momento por favor."
820 PRINT AT 19,8;S$
830 LET y=19
840 IF N<10 THEN LET x=14

```

26 MICROBASIC

"El autor de "y" es "

y como variables, los símbolos L\$ y N\$; éstos, respectivamen- te, almacenan el título y el autor del libro; en un momen- to determinado podrían tomar los valores L\$ = "EL QUIJO- TE" y N\$ = "CERVANTES"; en este caso el mensaje a imprim- ir sería:

El autor de EL QUIJOTE es
CERVANTES

Constantes numéricas

En el BASIC/Sinclair, los nú- meros se representan en cua- tro formatos:

- NOTACION ENTERA.
- NOTACION FRACCIONARIA.
- NOTACION EXPONENCIAL.
- NOTACION BINARIA.

Notación entera

En este formato se repre- sentan los números *enteros*, que por definición son aque- llos que no contienen parte decimal.

Los números enteros siguen las siguientes reglas:

- Pueden ser positivos y negativos.
- Precisión máxima de ocho dígitos.
- El signo "+" o espacio en blanco indica núme- ro positivo y el signo " — " negativo.

Por tanto, un número entero es- tá comprendido dentro del ran- go + 99999999 y — 99999999.

Ejemplos correctos:

- + 15934
- 823
- 9357
- 1
- + 12345678

Ejemplos incorrectos:

- 12.347 (Número con de- cimales).

```

850 IF N>=10 AND N<=99 THEN LET
x=13
860 IF N>=100 AND N<=999 THEN L
ET x=12
870 IF N>=1000 AND N<=9999 THEN
LET x=11
880 IF N>=10000 THEN LET x=10
890 PRINT AT y,x;N
900 REM
901 REM "BINARIO"
902 REM
910 DIM I(16)
915 LET Z=N
920 FOR Y=15 TO 0 STEP -1
930 IF Z>=INT (2↑Y) THEN LET Z=
Z-INT (2↑Y): LET I(Y+1)=1
940 NEXT Y
950 PRINT AT 10,6;S$
960 PRINT AT 10,7; OVER 1;" ";
970 FOR X=16 TO 1 STEP -1
980 PRINT OVER 1;I(X);
990 NEXT X
1000 PRINT #0;AT 1,0;"Desea otro
calculo (S/N)
1010 PAUSE 0
1020 LET A$=INKEY$
1030 IF A$="S" THEN BEEP 0.2,20:
GO TO 1055
1040 IF A$="N" THEN BEEP 0.2,20:
STOP
1050 GO TO 1010
1055 PRINT AT 10,7; OVER 1;" ";
1060 FOR X=16 TO 1 STEP -1
1070 PRINT OVER 1;I(X);
1080 NEXT X
1090 PRINT AT 10,6;" "
1100 PRINT AT 19,8;" "
1150 PRINT #0;AT 1,0;" "
1200 GO TO 500

```

935749103 (Tiene nueve dígitos).
 32G14 (Contiene un carácter no numérico).

Notación decimal

Sirve para representar aquellos números que contienen decimales. En BASIC, la "coma decimal" se sustituye por el "punto decimal", aunque su representación se denomine de "coma fija". Al igual que los números enteros, la precisión de los datos de salida es de ocho cifras significativas; sin embargo, los datos introducidos por teclado (entrada) pueden tener mayor longitud, ya

que el ordenador efectúa un redondeo.

El campo de los números decimales está comprendido entre + 0.00001 y + 99999999 / - 0.00001 y - 99999999. Cuando el dígito situado a la izquierda del punto decimal es cero, su representación puede ser omitida; es decir, la cifra .15 es la misma que 0.15.

Ejemplos de números decimales:

- 3.4569035
 + 5.47
 .37851
 + 3537591.2

Notación exponencial

Otra forma de expresar números es mediante la notación

exponencial, utilizando las potencias de base 10. Por ejemplo:

| NUMERO | NOTACION EXPONENCIAL |
|---------|----------------------|
| 43012 | 43.012×10^3 |
| 370000 | 37×10^4 |
| 93.23 | 9.323×10^2 |
| 93.23 | 9.323×10^1 |
| 1000000 | 1×10^6 |

El formato que utiliza el Spectrum para representar la notación exponencial, es el indicado en la fig. 1.

La *mantisa* es un número de ocho dígitos de precisión máxima.

La letra *e* (minúscula) o *E* (mayúscula) es un símbolo que indica que la base de potenciación es decimal (10).

El *exponente* es un número de dos dígitos de precisión máxima, que expresa la potencia a la que hay que elevar la base.

Tanto la mantisa como el exponente tienen sus propios signos "+" o "-".

En el ejemplo de la figura 1, podemos comprobar la notación + 3.7281234 E-14 es equivalente a $3.7281234 \times 10^{-14}$. Desplazando la coma (punto en BASIC) y rectificando por tanto el valor del exponente, podríamos tener muchas combinaciones y todas con el mismo valor, por ejemplo:

- 0.0003728 $\times 10^{10}$
- 0.0037281 $\times 10^{11}$
- 0.0372812 $\times 10^{12}$
- 0.3728123 $\times 10^{13}$
- 3.7281234 $\times 10^{14}$

De todas éstas, el Spectrum utiliza únicamente la 5, ya que proporciona el mayor número de cifras significativas en la

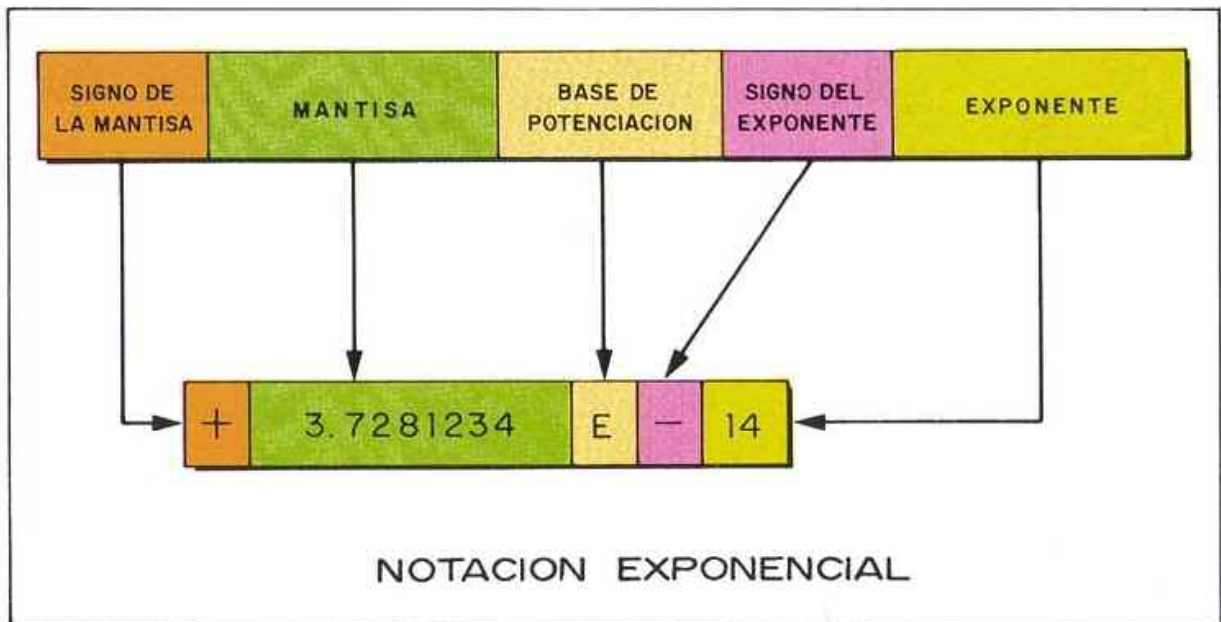


Fig. 1.

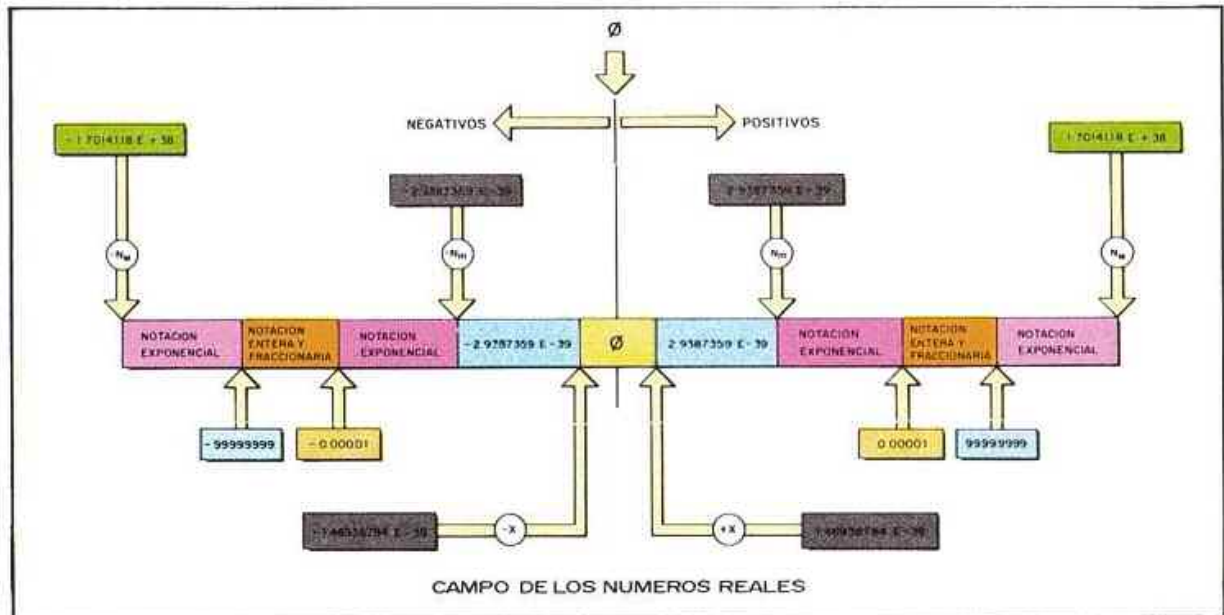


Fig. 2.

mantisa (un dígito distinto de cero, punto decimal y hasta siete dígitos más); este tipo de representación se denomina *coma flotante*.

La notación exponencial la utiliza el ordenador para representar números cuyo valor sea inferior a 0.000001 o superiores a 99999999, dentro del campo de los números positivos.

En la **figura 2**, se puede observar el campo de los núme-

ros reales; a la derecha del 0 se encuentran los números positivos y a la izquierda los negativos; ya que ambas partes son iguales, bastará con explicar una de ellas.

El mayor número positivo que se puede representar es "Nm", cuyo valor es igual a 1.7014118 E + 38; cuando el ordenador realiza una operación y el resultado es superior

a "Nm", se presenta en pantalla el mensaje:

6 Number too big, a : b

a = línea de programa donde se originó el error.

b = n.º de instrucción dentro de la línea.

El menor número positivo es "Nm" y su valor es 2.9387359E - 39; cualquier

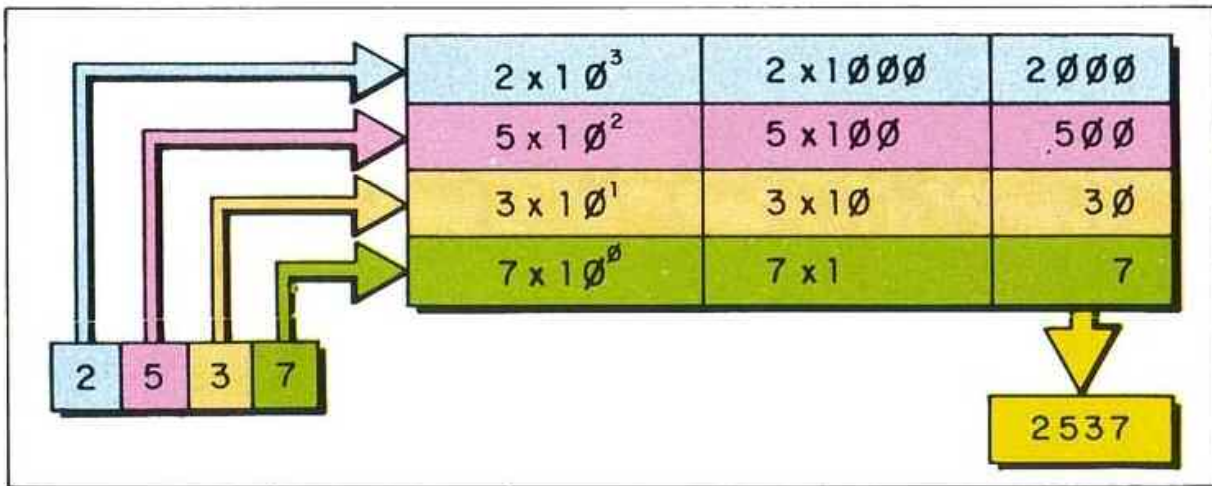


Fig. 3.

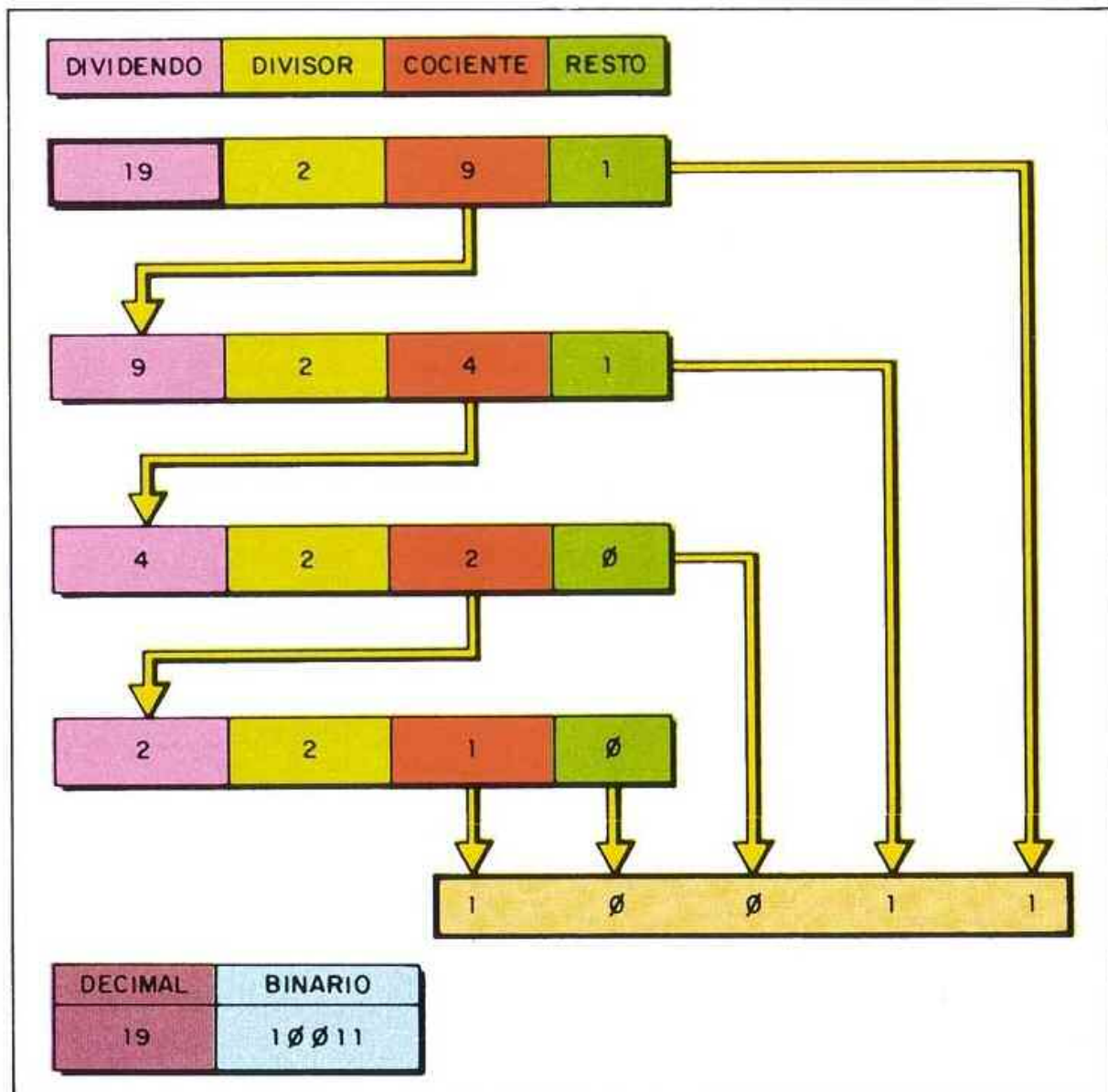


Fig. 4.

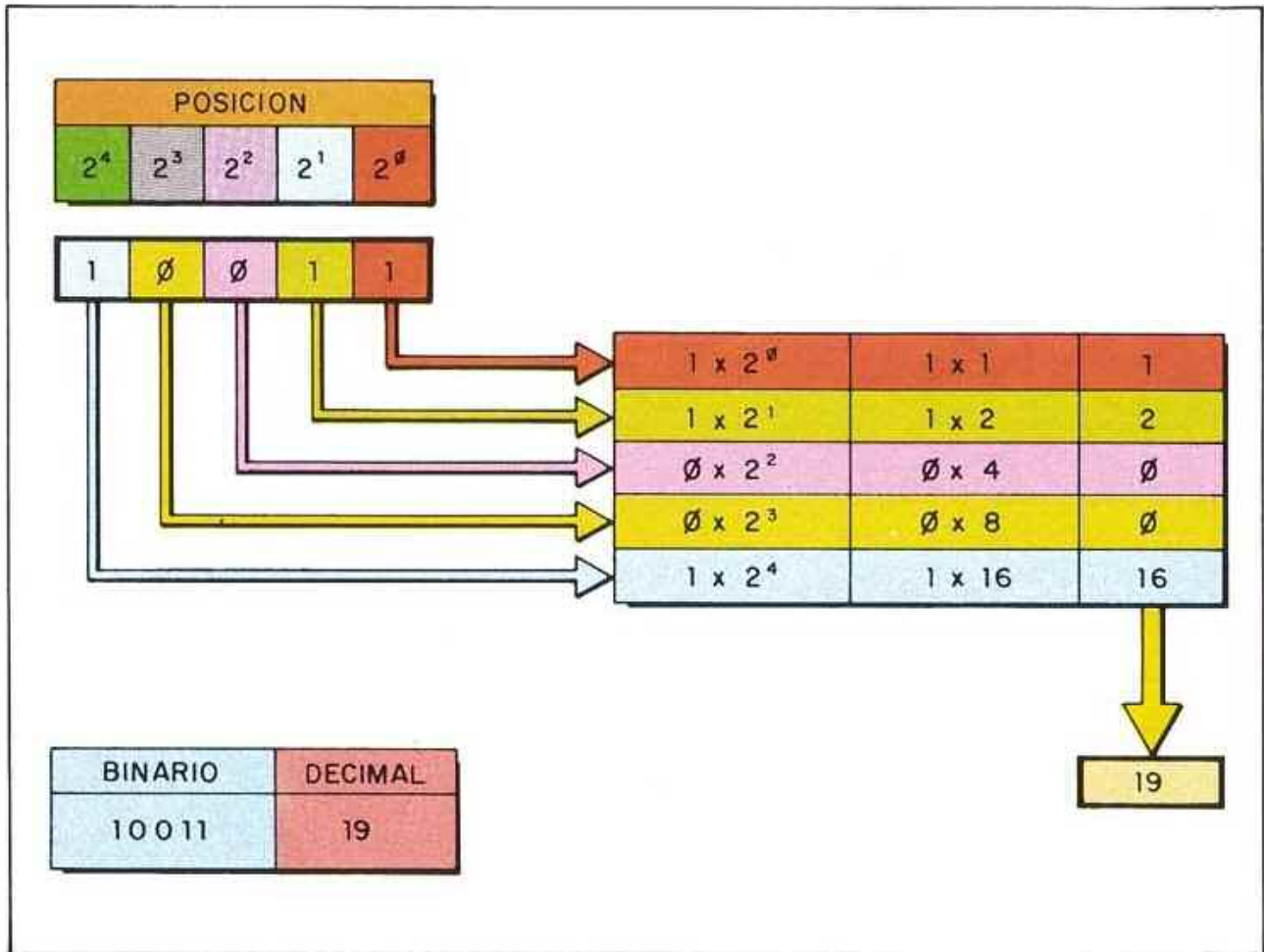


Fig. 5.

número inferior a éste y superior a "X" (1.46936794E - 39) será representado con el valor de "Nm". Valores inferiores a "X" serán tomados como 0.

Teclee el siguiente miniprograma y podrá comprobar cómo representa el Spectrum las constantes numéricas que le sean introducidas por el teclado.

```

10 INPUT "CONSTANTE NUMERICA?"
   "a
20 PRINT a
30 GO TO 10
  
```

Notación binaria

Para expresar un número, normalmente empleamos el sistema *decimal*, en el que se manejan diez símbolos (0, 1, 2, 3, 4, 5, 6, 7, 8 y 9) con los que se puede expresar cualquier

valor, utilizando las potencias de base 10 (fig. 3).

Cada número tiene un valor dependiendo de la posición que ocupa; por ejemplo, en el n.º 2537, el 2 ocupa la posición de las unidades de millar y tiene el valor 2000, el 5 las centenas (500), el 3 las decenas (30) y el 7 las unidades (7); sumando todos, obtenemos el valor 2537.

Otro sistema de numeración es el que utiliza internamente el ordenador y que se llama *binario*, ya que sólo utiliza dos símbolos, el 0 y el 1. Para expresar un valor en este sistema se utilizan las potencias de base 2; el valor de los símbolos también es opcional, ya que depende de la posición que ocupen.

En el Spectrum, los datos que deban ser introducidos en

notación binaria, tienen que ir precedidos de la palabra clave BIN, que se encuentra situada en la letra B. Los números binarios pueden ser positivos o negativos, pero en cualquier caso tienen que ser enteros, no se permiten números con parte decimal. El mayor valor expresado en BIN 1111111111111111, que equivale al valor 65535 en decimal, y el menor, lógicamente, - BIN 1111111111111111 (- 65535).

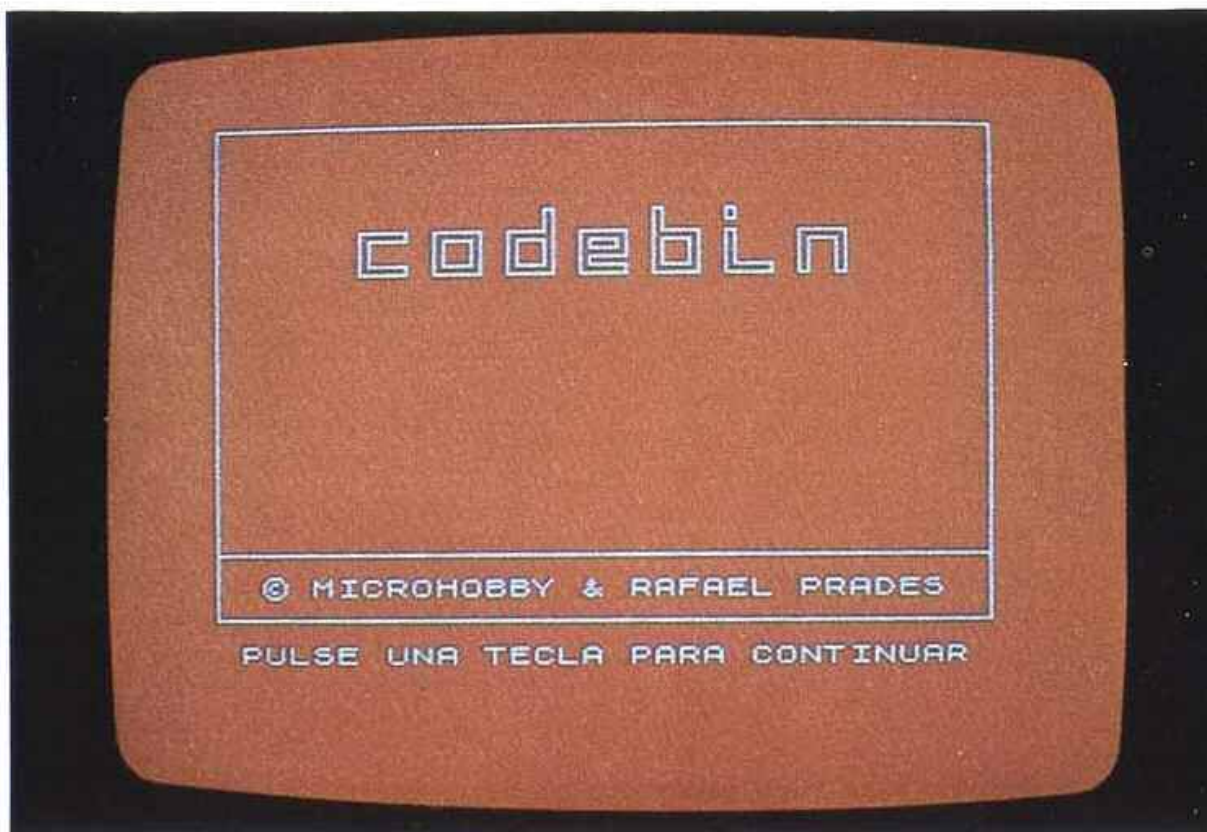
Ejemplos de notación binaria:

```

BIN 11010
-BIN 001110
BIN 10
BIN 1111
  
```

Decimal-binario

Para transformar un número decimal en binario, observe



Portada del programa CODEBIN.

el gráfico de la **figura 4** que lo explica con detalle. El n° a transformar es el 19, éste se divide entre dos, el cociente resultante se vuelve a dividir otra vez en dos, y así sucesivamente hasta que el último cociente sea 1; el número binario se forma con este cociente y con los restos obtenidos en las divisiones, tal como indica la figura.

Binario-decimal

Un número se transforma en su correspondiente decimal multiplicando cada símbolo (0 ó 1) por el valor de la posición que ocupa en base 2; sumando todos los valores se obtiene el decimal. La **figura 5** representa gráficamente esta transformación.

Ejercicio

Si desea convertir un número binario en decimal, sin te-

ner que hacer cálculos engorrosos, utilice el siguiente comando directo para números positivos:

```
PRINT BIN "notación binaria"
```

y para los negativos:

```
PRINT -BIN "notación binaria"
```

ejemplos:

```
PRINT BIN 1010
PRINT -BIN 111101
```

Si, por el contrario, lo que desea es pasar un número decimal a binario, edite el programa que le proponemos en este capítulo.

Cuando lo salve en cinta, hágalo de la forma:

```
SAVE "Codebin" LINE 10
```

De esta manera se autoejecutará al utilizar el comando LOAD y no hará falta teclear RUN una vez cargado.

Realizada la presentación del programa en pantalla, pul-

sando cualquier tecla, se pasa a la ejecución del programa principal. En la parte inferior aparece un prompt parpadeante " " indicando que el ordenador está preparado para la introducción del número decimal. Para teclear un negativo, es necesario que vaya precedido por el signo " - ". Si el valor del dato fuera superior a ± 65535 , el ordenador nos presentará un mensaje de error y se visualizará de nuevo el prompt. Cuando el dato es correcto, aparece un mensaje de espera, ya que el ordenador necesita un tiempo para realizar los cálculos; finalizado éste, es presentado de forma binaria el n° deseado.

Para realizar un nuevo cálculo, pulse la tecla "S" e introduzca el nuevo valor.

Constantes alfanuméricas

También son conocidas como *cadenas* o *strings*. Están

formadas por una secuencia de caracteres encerrados entre comillas; la estructura general es:

STRING
"cadena de caracteres"

Los caracteres pueden ser letras mayúsculas o minúsculas (a, z, L,...), números (8, 4, 0...), caracteres especiales (©, #, ↑,...) o comandos BASIC (VERIFY, BIN, OUT,...).

Ejemplos:

- "Curso BASIC/

SINCLAIR"
- "© MICROHOBBY
& RAFAEL PRADES"
- "Enero tiene 31 días"
- "3,1415927 es el valor de PI"

Cuando una cadena no tiene ningún carácter ("") se denomina *vacía* o *nula*, la cadena (" ") no se considera vacía, ya que contiene el carácter correspondiente al espacio.

Si dentro de una cadena ha de ir incluido el carácter de las comillas ("), éste deberá telearse por duplicado, como indicador de que no es el final de la cadena.

Ejemplos:

- "El ""Spectrum"" es un ordenador personal"
- "El significado de ""Anticuerpo"" es..."

Si visualizamos estas dos cadenas, anteponiendo el comando PRINT, observaremos que las comillas no aparecen repetidas.

Variables numéricas

Este tipo de variables está formada únicamente por letras y números, pudiendo estar constituida su longitud por distinta cantidad de caracteres. El primer carácter debe ser obligatoriamente una letra.

Ejemplos:

- Color
- V12
- puntuación
- xlp12

No existe ninguna diferencia entre las variables escritas en letras mayúsculas o minúsculas; los siguientes ejemplos se refieren a la misma variable escrita de distintas formas:

- FUERZA
- Fuerza
- fuerza
- FuErZa

En ocasiones, para facilitar una posterior interpretación del significado de la variable, puede introducirse el carácter *espacio* tantas veces como se quiera. No existe ninguna diferencia entre una variable escrita con espacios y otra que no lo esté.

Ejemplos:

- RESISTENCIA
DEL AIRE
- Resistenciadel Aire
- Resistenciadelaire

El valor asignado a una variable de tipo numérico debe ser una constante u otra variable, ambas, lógicamente, de tipo numérico.

Ejemplos:

- KILO = 12
- PESO = KILO
- Potencia = 30
- grados = 27
- Temperatura = grados

Variables alfanuméricas

Las variables alfanuméricas o de cadena, están constituidas por una sola letra, mayúsculas o minúsculas, indistintamente, seguida del símbolo *dólar* "\$".

Ejemplos:

- M\$
- J\$
- X\$
- T\$

El valor asignado a una variable alfanumérica debe ser, o una constante o una variable, ambas del tipo cadena alfanumérica.

Ejemplos:

- S\$ = "Producto"
- T\$ = S\$
- K\$ = "1024 Kbytes"
- N\$ = "3518E + 14"

OPERADORES

Los operadores son símbolos que expresan el tipo de operación que ha de realizarse, bien entre dos constantes, bien entre una variable y una constante, etc. Veamos unos ejemplos.

| OPERANDO | OPERADOR | OPERANDO |
|----------|----------|-----------|
| 3 | + | 12 |
| valor | = | 1 |
| A\$ | <> | «Antonio» |
| 527 | AND | 32 |

Existen tres tipos de operadores:

- ARITMETICOS.
- DE RELACION.
- LOGICOS.

Operadores aritméticos

Son aquellos que permiten ejecutar las operaciones aritméticas básicas: suma, resta, etc. Los símbolos utilizados son los indicados en la tabla.

| OPERACION | SIMBOLO |
|----------------|---------|
| Suma | + |
| Resta | - |
| Multiplicación | * |
| División | / |
| Potenciación | ↑ |

Ejemplos:
 $12 \cdot 7$ (12 por 7)
 $3 \uparrow 2$ (3 al cuadrado)
 $456 / 2$ (456 entre 2)

Expresiones aritméticas

Son conjuntos de constantes y variables unidas entre sí por operadores aritméticos.

Ejemplos:

$(3 + 5) / 7$
 $8 \uparrow 2 / 1 \cdot 5$
 $gr \cdot X \uparrow 3 / tz$
 $voltaje \uparrow 2 / resistencia \cdot tiempo$

Las variables utilizadas tienen que estar definidas previa-

mente, ya que de lo contrario, cuando el ordenador realice un cálculo y alguna no lo esté, enviará el siguiente mensaje de error:

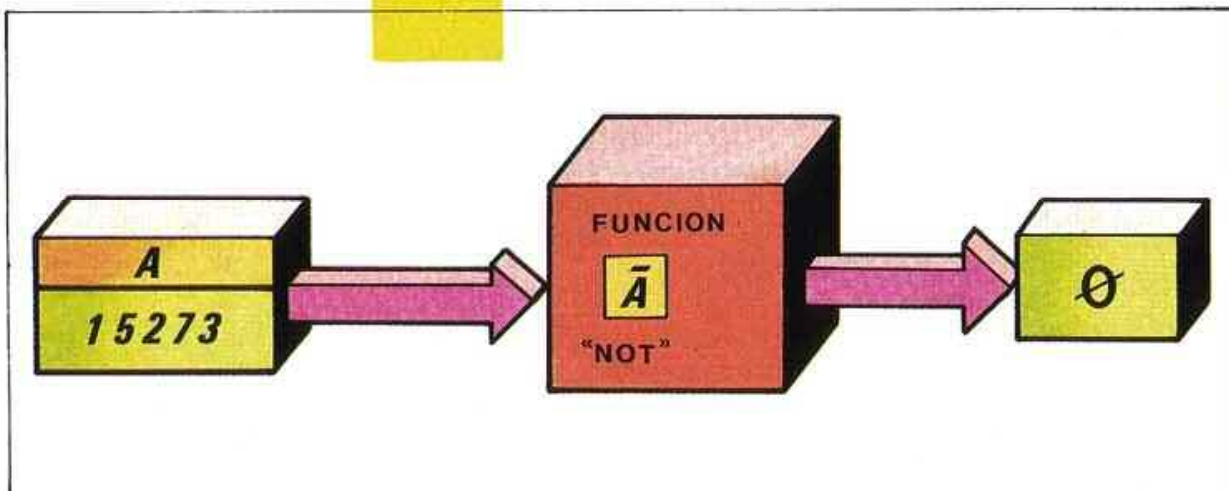
2 Variable not found

Al igual que en álgebra, se pueden utilizar los paréntesis, pero no las multiplicaciones implícitas, es decir, la expresión $(5 + X)(8 - Y)$, en BASIC, se escribe:

$(5 + X) * (8 - Y)$

Cálculo de expresiones

El ordenador cuando realiza un cálculo, lo hace siempre atendiendo al valor de *prioridad* que tenga cada operación. De las cinco operaciones aritméticas básicas, la potenciación tiene mayor prioridad que el resto, después la multiplicación y la división, ambas con el mismo valor, y por último, la suma y la resta, también con idéntica prioridad.



Función «AND». Estructura 1.

Cuando en una expresión hay dos operaciones de la misma prioridad, el ordenador efectúa los cálculos de izquierda a derecha.

Veamos por pasos cómo el ordenador calcula la siguiente expresión para $x = 3$ e $y = 2$:

- $16 + x + 3\phi - y / 2$
 a) $16 + 3 + 3\phi - 2\phi / 2$
 b) $16 + 9\phi - 2\phi / 2$
 c) $16 + 9\phi - 1\phi$
 d) $1\phi 6 - 1\phi$
 Resultado: 96

Utilizando los paréntesis se puede alterar el orden de evaluación de las operaciones, ya que éstos se calculan primero. Aprovechando la expresión anterior, vamos a observar que el resultado varía colocando los paréntesis en las operaciones de menos prioridad.

- $(16 + x) * (3\phi - y) / 2$
 a) $(16 + 3) * (3\phi - 2\phi) / 2$
 b) $18 * (3\phi - 2\phi) / 2$
 c) $18 * 1\phi / 2$
 d) $18\phi / 2$
 Resultado: 9ϕ

Operadores de relación

Permiten realizar las comparaciones entre operandos (constantes o variables) tanto numéricos como de cadena.

| OPERADOR | SIMBOLO |
|---------------|---------|
| Igual | = |
| Distinto | <> |
| Mayor | > |
| Menor | < |
| Mayor o igual | >= |
| Menor o igual | <= |

Las operaciones realizadas con estos operadores sólo tie-

nen dos resultados o soluciones, es decir:

— Si la condición impuesta por el operador se cumple, es decir, que es *verdadera* (true), el valor del resultado es «1».

— Si por el contrario, la condición no se cumple, es decir, que es *falsa* (false), el valor se vuelve «0».

Ejecute los siguientes comandos directos y comprobará lo explicado anteriormente.

| COMANDO | CONDICION | RESULTADO |
|---------------|-----------|-----------|
| PRINT 3 = 7 | Falsa | 0 |
| PRINT 10 > 99 | Verdadera | 1 |
| PRINT 8 <> 8 | Falsa | 0 |
| PRINT 10 >= 7 | Verdadera | 1 |

El símbolo «=» también sirve para asignar un valor a una variable.

Operadores lógicos

Se utilizan para realizar las operaciones lógicas a nivel de expresión.

| OPERACION | FUNCION |
|-----------------|---------|
| Producto lógico | AND |
| Suma lógica | OR |
| Negación | NOT |

En otras ocasiones, estos operadores se utilizan para relacionar dos expresiones mediante una condición, por ejemplo:

— Que el ordenador realice una determinada tarea si se cumplen varias relaciones.

| RELACION | OPERADOR | RELACION |
|-------------|----------|----------|
| $x >= \phi$ | AND | $x <= 9$ |

En este caso sólo se cumplirá la condición cuando la variable x sea mayor o igual a 0 y a la vez sea menor o igual a 9, es decir, que esté comprendida entre 0 y 9.

— Que se realice la tarea cuando simplemente alguna de las condiciones se cumpla.

| RELACION | OPERADOR | RELACION |
|-----------------|----------|----------|
| $a = 1\phi\phi$ | OR | $t > 1$ |

La condición se cumple, bien cuando el valor de la variable a sea $1\phi\phi$, bien cuando el valor de t sea mayor que 1 ó bien cuando se cumplan ambas condiciones.

Función «AND»

La estructura de esta función es la que se muestra a continuación, siendo a y b dos expresiones numéricas.

a AND b

Estas expresiones solamente pueden tomar los valores 0 o distinto de 0, este último le vamos a representar como «<>0». Teniendo en cuenta todas las posibles combinaciones que pueden tomar a y b , vamos a mostrar su *tabla de verdad*:

| a | b | RESULTADO |
|-----|-----|-----------|
| 0 | 0 | 0 |
| 0 | <>0 | $a(0)$ |
| <>0 | 0 | 0 |
| <>0 | <>0 | a |

de ésta se deduce que si el valor de b es igual a 0, el resultado de la función es 0, independientemente del valor de a . Si el valor de b es distinto de 0, la función asume el valor de a .

Ejemplos:

| OPERACION | RESULTADO |
|---------------------------------|---------------|
| 132 AND \emptyset | \emptyset |
| 8 \emptyset AND 1 \emptyset | 8 \emptyset |
| \emptyset AND 25 | \emptyset |
| 1 AND 9 \emptyset | 1 |

La función AND, también puede tomar la estructura:

a\$ AND b

en este caso el resultado de la función será una cadena vacía

las expresiones a y b sólo pueden tomar valores numéricos. Con todas las combinaciones se obtiene la siguiente tabla de verdad:

| a | b | Resultado |
|----------------|----------------|-------------------|
| \emptyset | \emptyset | a (\emptyset) |
| \emptyset | <> \emptyset | 1 |
| <> \emptyset | \emptyset a | a |
| <> \emptyset | <> \emptyset | 1 |

Mirando detenidamente la tabla se observa que si b to

Función «NOT»

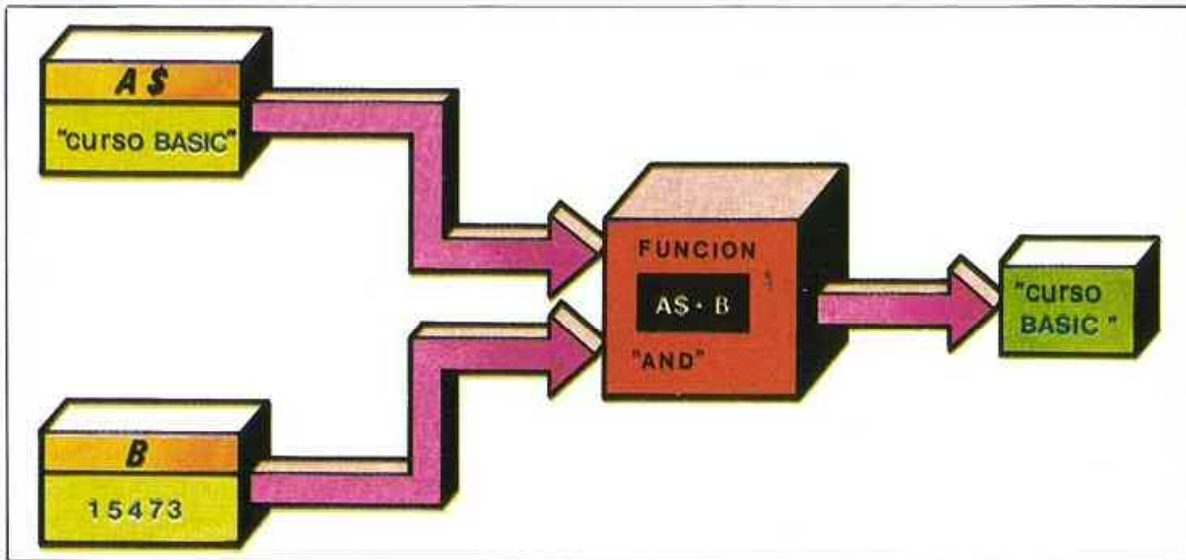
La función NOT afecta solamente a una expresión y su estructura es:

NOT a

su tabla de verdad es:

| a | RESULTADO |
|----------------|-------------|
| \emptyset | 1 |
| <> \emptyset | \emptyset |

Como se puede apreciar, el resultado de la función es la



Función «AND». Estructura 2.

cuando b sea \emptyset y será la cadena $a\$$ cuando b sea distinto de \emptyset .

Ejemplos:

| OPERACION | RESULTADO |
|---------------------------|-----------|
| «PEDRO» AND 3 \emptyset | «PEDRO» |
| «JUAN» AND \emptyset | «» |

Función «OR»

La estructura de la función OR es la siguiente:

a OR b

ma el valor \emptyset , la función resultante adquiere el de a , sin embargo, si el valor de b es diferente de \emptyset , el valor de la función es 1, independientemente del que tenga a .

Ejemplo:

| OPERACION | RESULTADO |
|------------------------------|---------------|
| \emptyset OR 3 \emptyset | 1 |
| 2 \emptyset OR \emptyset | 2 \emptyset |
| 55 OR 7 | 1 |
| \emptyset OR \emptyset | \emptyset |

negación de la expresión a , es decir, vale 1 si a es igual a \emptyset y \emptyset si a es distinto de \emptyset . Ejemplo:

| OPERACION | RESULTADO |
|-----------------|-------------|
| NOT 7 | \emptyset |
| NOT \emptyset | 1 |

Anteponiendo el signo «—» a la función NOT, el resultado cambia de signo.

| OPERACION | RESULTADO |
|-------------------|-------------|
| — NOT 4 | \emptyset |
| — NOT \emptyset | — 1 |

| CARACTERES ASCII | | | | | | | | | | | | | | | | | | | |
|------------------|-----|-----|-----|-----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
| C | u | r | s | o | | B | A | S | I | C | / | S | I | N | C | L | A | I | R |
| ↑ | ↑ | ↑ | ↑ | ↑ | ↑ | ↑ | ↑ | ↑ | ↑ | ↑ | ↑ | ↑ | ↑ | ↑ | ↑ | ↑ | ↑ | ↑ | ↑ |
| 67 | 117 | 114 | 115 | 111 | 32 | 66 | 65 | 83 | 73 | 67 | 47 | 83 | 73 | 78 | 67 | 76 | 65 | 73 | 82 |
| CODIGO DECIMAL | | | | | | | | | | | | | | | | | | | |

Función «OR».

Ejercicio

Estas funciones pueden en-
cadenarse para formar otra
más complicada. Haciendo
uso de los paréntesis «()» se
consigue facilitar su interpre-
tación; por ejemplo:

$$(X \text{ AND } Y) \text{ OR } (\text{NOT } X)$$

si asignamos a la variable X el
valor 3 y a Y el valor 5, veamos
cual es el resultado final resol-
viendo la función por pasos:

a) $(3 \text{ AND } 5) \text{ OR } (\text{NOT } 3)$

b) $3 \text{ OR } (\text{NOT } 3)$

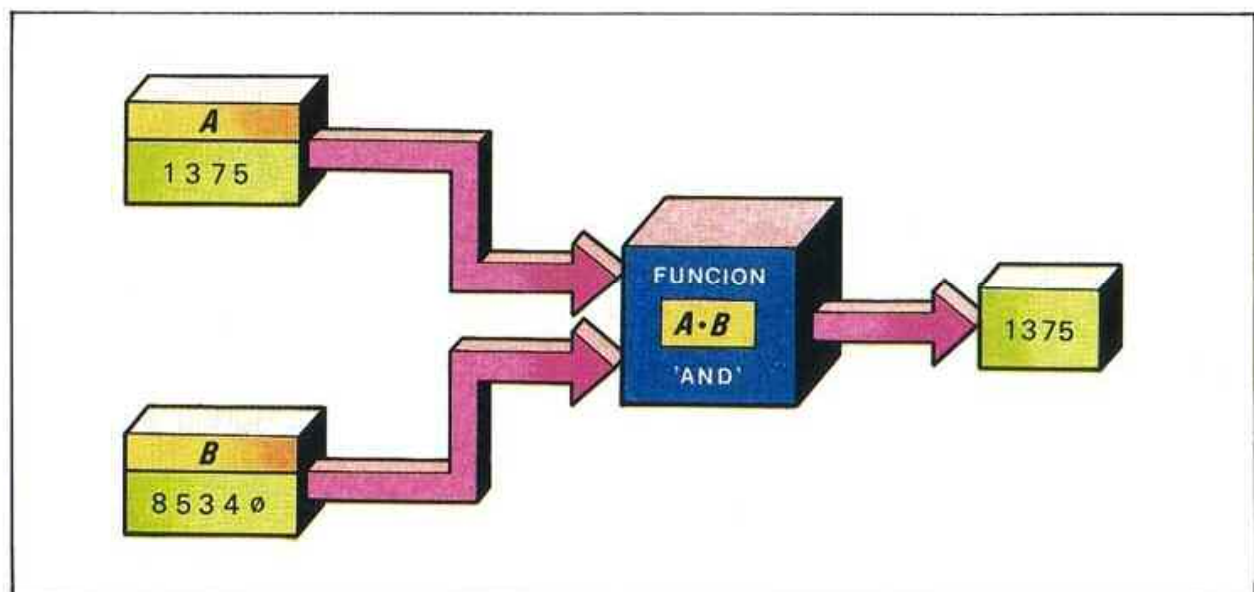
c) $3 \text{ OR } \phi$

Resultado = 3

Como ejercicio, intente re-
solver la siguiente función:

— $\text{NOT } ((X \text{ OR } Y) \text{ AND } (\text{NOT } Z))$

Para los valores: $X = 8$, $Y = 1\phi$ y $Z = 83\phi$. Si desea com-
probar el resultado de su cálculo
introduzca como comando
directo la función anterior pre-
cedida de la sentencia PRINT.



Función «NOT».

CODIGO ASCII

Para representar un carácter en la pantalla del televisor o en la impresora, el ordenador utiliza el *código ASCII*; éste permite la transferencia de datos entre el ordenador y los dispositivos conectados a él (periféricos). Cada número, letra o símbolo tiene su representación en este código.

ASCII es la abreviatura, en inglés, de «American Standard Code for Information Interchange» que, traducido al idioma español, significa «Código normalizado Americano para intercambio de información».

El ASCII completo consta de 256 caracteres, cuyo código está comprendido entre 0 y 255.

Edite el programa número «1» que, una vez ejecutado, visualiza en pantalla los caracteres ASCII usados por el ZX Spectrum y comprendidos entre el código 32 y el 255.

Manejo de la tabla

Para conocer el *código decimal* correspondiente a un determinado carácter ASCII, basta con sumar los números de *fila* y *columna* indicados en la tabla.

código = fila + columna

Ejemplos:

| CARACTER ASCII | NUMERO | | CODIGO |
|-------------------|--------|---------|--------|
| | FILA | COLUMNA | |
| ≠ | 30 | 5 | 35 |
| USR | 190 | 2 | 192 |
| M | 70 | 7 | 77 |
| 8 | 50 | 6 | 56 |

PROGRAMA 1

```

10 REM *****
*
*          CURSO          *
*        BASIC/SINCLAIR  *
*          "ASCII"       *
*
*****

20 BORDER 1: PAPER 1: INK 7: C
LS

30 REM *****
*
*  IMPRESION DE  *
*  CARACTERES   *
*
*****

40 FOR Y=32 TO 255 STEP 44
50 FOR X=Y TO Y+43
60 IF X<100 THEN PRINT " ";
70 PRINT X;" ";CHR$ X,
80 IF X=255 THEN GO TO 200
90 NEXT X
100 PRINT #0;AT 1,0;"Pulse una
tecla para continuar."
110 PAUSE 0: BEEP 0.05,20
115 CLS
120 NEXT Y
200 REM

*****
*
* CONTINUACION ? *
*
*****

210 POKE 23658,8
220 PRINT #0;AT 0,0;"Desea obtener un nuevo listado (S/N)"
230 PAUSE 0: LET D$=INKEY$
240 IF D$="S" THEN BEEP 0.05,20
: GO TO 10
250 IF D$="N" THEN BEEP 0.05,20
: CLS : STOP
260 BEEP 0.2,-15: GO TO 230

```

El programa «2» visualiza el código decimal correspondiente a la cadena ASCII introducida, de un máximo de 20 caracteres.

Organización del ASCII

Dentro del código ASCII, usado por el Spectrum, pueden encontrarse diversas zo-

nas, teniendo cada una de ellas unas características distintas:

- CODIGO TRANSPARENTE.
- CODIGO ASCII CONVENCIONAL.
- CODIGO ASCII ESPECIFICO.
- GRAFICOS PREDEFINIDOS.

- GRAFICOS DE USUARIO.
- TOKENS.

El denominado *código transparente* está formado por una serie de comandos y funciones de control, como por ejemplo, el control de los cursores, el del color, la función EDIT, etc. Estos caracteres ASCII de control son específicos

| OPERACION | SIMBOLO | PRIORIDAD | EJEMPLO |
|----------------------------|----------------------------------|-----------|-----------------|
| FRAGMENTACION | TO | 12 | "Juan" (1 TO 2) |
| POTENCIACION | ↑ | 10 | 10 2 |
| NEGACION | — | 9 | — 15 |
| MULTIPLICACION | * | 8 | 7 * 13 |
| DIVISION | / | | 152 / 2 |
| ADICION | + | 6 | 10 + 4 |
| SUBSTRACCION | — | | 8 — 5 |
| OPERADORES RELACIONALES | = > < < > > = < = | 5 | 10 4 |
| OPERADORES LOGICOS | NOT | 4 | NOT 5 |
| | AND | 3 | 10 AND 1 |
| | OR | 2 | 7 OR 0 |

del Spectrum, otros ordenadores disponen de otro juego distinto.

El código transparente está comprendido entre el ϕ y el 31, en decimal. Dentro de éste, existen caracteres que no son utilizados, cuando el Spectrum representa uno de estos códigos aparece una interrogación (?) en su lugar.

Los códigos comprendidos entre el 32 y el 127 forman el ASCII convencional de todo ordenador. En esta zona se encuentran los caracteres correspondientes a las 26 letras mayúsculas, a sus homólogas las minúsculas, a los diez dígitos (ϕ al 9), al carácter «espacio» y a una serie de símbolos (ω , &, &...) y signos ortográficos (", "!", "?"...).

A pesar de ser la zona convencional del ASCII, el Spectrum tiene dos caracteres particulares, los correspondientes a la «Libra» (£) y al «Copyright» (©), códigos 96 y 127, respectivamente.

Los gráficos predefinidos, es decir, los símbolos que están dibujados sobre las teclas con los números «1» a «8», y los complementarios, tienen un código comprendido entre el 128 y el 143. A continuación se encuentran los códigos correspondientes a los «DGU» (Gráficos Definidos por el Usuario), que como ya se indicó en el capítulo primero, éstos están asignados a las teclas con las letras de «A» y la «U» y su código está comprendido entre el 144 y el 164.

Al final del juego de caracteres se encuentran los *tokens*, término que simboliza a las 88 palabras clave (RUN, STOP, CLEAR...) una vez *codificadas*, éstas forman la lista de comandos, sentencias y funciones. También se encuentran en esta zona tres caracte-

PROGRAMA 2

```

10 REM *****
*
*          CURSO          *
*        BASIC/SINCLAIR  *
*        ASCII/DECIMAL   *
*
*****

20 BORDER 1: PAPER 1: INK 7: C
LS
22 LET FLAG52=23658
24 LET PIP=23609
30 REM

*****
*
*   ENTRADA DE           *
*   CARACTERES          *
*
*****

32 POKE PIP,100
34 POKE FLAG52,0
40 INPUT "CADENA: ";A$
50 IF LEN A$>20 THEN GO TO 40
60 POKE PIP,0
70 REM

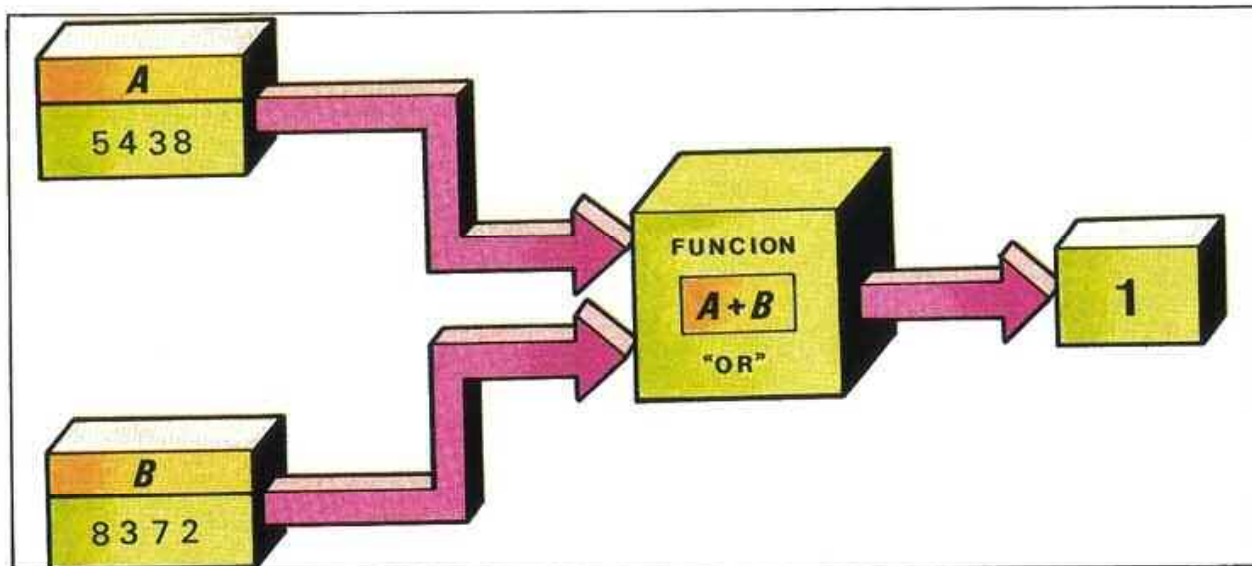
*****
*
*   VISUALIZACION      *
*
*****

80 FOR X=1 TO LEN A$
90 PRINT A$(X); " ";CODE A$(X)
)
100 NEXT X
110 REM

*****
*
*   CONTINUACION      *
*
*****

120 POKE FLAG52,8
130 PRINT #0;AT 1,0;"Quiere vol
ver a empezar (S/N)"
140 PAUSE 0: LET D$=INKEY$
150 IF D$="S" THEN BEEP 0.05,20
: GO TO 10
160 IF D$="N" THEN BEEP 0.05,20
: CLS : STOP
170 BEEP 0.2,-15: GO TO 140

```

terres que ocupan dos posiciones cada uno, estos son los correspondientes a los símbolos "<=", ">=", y "<>".

Transmisión del ASCII

En el lenguaje BASIC del Spectrum, la transmisión de caracteres ASCII puede efectuarse o directamente en este código o anteponiendo la sentencia "CHR \$" al correspondiente código decimal.

Ejemplos:

| ASCII | DECIMAL |
|-----------|----------------|
| PRINT «A» | PRINT CHR\$ 65 |
| PRINT «a» | PRINT CHR\$ 97 |

El primer método ofrece ciertas ventajas con respecto al segundo.

- Es legible directamente.
- Ocupa menos sentencias.
- Más rápido de ejecución.

si por el contrario todo lo que desea es semi-camufilar el mensaje, será conveniente utilizar el segundo.

Ejecute la siguiente instrucción directa y compare el resultado con el proporcionado por el programa número «3».

PRINT «Maese Pérez el organista»

40 MICROBASIC

PROGRAMA 3

```

10 REM *****
*          *
*   CURSO   *
*          *
* BASIC/SINCLAIR *
*          *
*   "CHR$"  *
*          *
*****

20 BORDER 7: PAPER 7: INK 0: C
LS

30 REM *****
*          *
* IMPRESION DE *
*          *
* CARACTERES  *
*          *
*****

35 RESTORE
40 FOR X=1 TO 24
50 READ codigo decimal
60 PRINT CHR$ codigo decimal;
70 NEXT x
75 PRINT
80 REM *****
*          *
*   TABLA DE  *
*          *
* CARACTERES  *
*          *
*****

90 DATA 77,97,101,115,101,32,8
0,101,114,101,122,32,101,108,32,
111,114,103,97,110,105,115,116,9
7

```

ORGANIZACION DEL JUEGO DE CARACTERES ASCII DEL SPECTRUM

ORGANIZACION DEL JUEGO DE CARACTERES "ASCII" USADO EN EL ZX SPECTRUM

| | | COLUMNAS | | | | | | | | | |
|-----|-----------------|---------------|------------|-------------|--------|-------|--------|-------------|---------------|---------------|----------------|
| | | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
| 0 | | | | | | | | PRINT coma | EDIT | cursor izda. | cursor dcha. |
| 10 | cursor abajo | cursor arriba | DELETE | ENTER | número | | | INK control | PAPER control | FLASH control | BRIGHT control |
| 20 | INVERSE control | OVER control | AT control | TAB control | | | | | | | |
| 30 | | | espacio | ! | " | # | \$ | % | & | ' | |
| 40 | (|) | * | + | , | - | . | / | 0 | 1 | |
| 50 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | : | ; | |
| 60 | < | = | > | ? | @ | A | B | C | D | E | |
| 70 | F | G | H | I | J | K | L | M | N | O | |
| 80 | P | Q | R | S | T | U | V | W | X | Y | |
| 90 | Z | [| \ |] | ↑ | - | £ | a | b | c | |
| 100 | d | e | f | g | h | i | j | k | l | m | |
| 110 | n | o | p | q | r | s | t | u | v | w | |
| 120 | x | y | z | | | | ~ | © | ☐ | ☐ | |
| 130 | ☐ | ☐ | ☐ | ☐ | ☐ | ☐ | ☐ | ☐ | ☐ | ☐ | |
| 140 | ☐ | ☐ | ☐ | ☐ | (A) | (B) | (C) | (D) | (E) | (F) | |
| 150 | (G) | (H) | (I) | (J) | (K) | (L) | (M) | (N) | (O) | (P) | |
| 160 | (Q) | (R) | (S) | (T) | (U) | RND | INKEYS | PI | FN | POINT | |
| 170 | SCREENS | ATTR | AT | TAB | VALS | CODE | VAL | LEN | SIN | COS | |
| 180 | TAN | ASN | ACS | ATN | LN | EXP | INT | SQR | SGN | ABS | |
| 190 | PEEK | IN | USR | STRS | CHRS | NOT | BIN | OR | AND | <= | |
| 200 | >= | <> | LINE | THEN | TO | STEP | DEF FN | CAT | FORMAT | MOVE | |
| 210 | ERASE | OPEN = | CLOSE = | MERGE | VERIFY | BEEP | CIRCLE | INK | PAPER | FLASH | |
| 220 | BRIGHT | INVERSE | OVER | OUT | LPRINT | LLIST | STOP | READ | DATA | RESTORE | |
| 230 | NEW | BORDER | CONT | DIM | REM | FOR | GOTO | GO SUB | INPUT | LOAD | |
| 240 | LIST | LET | PAUSE | NEXT | POKE | PRINT | PLOT | RUN | SAVE | RAND | |
| 250 | IF | CLS | DRAW | CLEAR | RETURN | COPY | | | | | |

☐ CODIGO TRANSPARENTE ESPECIFICO.

☐ CARACTERES "ASCII" ESPECIFICOS.

☐ GRAFICOS DE USUARIO.

☐ CODIGO NO UTILIZADO.

☐ CARACTERES "ASCII" CONVENCIONAL.

☐ GRAFICOS PREDEFINIDOS.

☐ JUEGO DE COMANDOS SENTENCIAS Y FUNCIONES.

OPERACIONES CON CADENAS

Concatenación de cadenas

Con este nombre se designa la operación de *encadenamiento* de un conjunto de constantes o variables alfanuméricas. La *concatenación* se realiza con el operador aritmético de la suma «+». Pueden tener diversas estructuras.

"Cons1" + "Cons2" + ... + "ConsN"
 Var 1\$ + Var2\$ + ... + VarN\$
 "Cons1" + Var1\$ + Var2\$ + ... +
 "ConsN"
 Cons = Constante de cadena
 Var = Variable de cadena

Ejemplos:

— El resultado de la concatenación de las constantes "Esta frase" + "e encade" + "na" + "da" + " " + "es muy larga" es la siguiente:

"Esta frase encadenada es muy larga"

— Si asignamos a la variable a\$ el valor "ZX SP" y a b\$ el valor "ECTRUM", efectuando la operación c\$ = a\$ + b\$, la variable c\$ tendrá el valor:

"ZX SPECTRUM"

— Siendo a\$ = "Programas" y b\$ = "aplicacion", el resultado de a\$ + "de" + b\$ será:

"Programas de aplicación"

Como ejercicio demostrativo edite el programa número «1». Cuando se ejecuta, el ordenador espera que le sean introducidas tres cadenas, con una longitud máxima de 10 caracteres cada una. Posterior-

PROGRAMA 1

```

10 REM *****
*
*          CURSO          *
*        BASIC/SINCLAIR  *
*        *****        *
*        CONCATENACION   *
*        *****        *
*
20 BORDER 1: PAPER 1: INK 7: C
LS
30 REM *****
*
*  INTRODUCCION          *
*    DE CADENAS          *
*        *****        *
*
100 INPUT "CADENA 1: ", A$
110 IF LEN A$ > 10 THEN GO TO 100
115 PRINT "CADENA 1: "; A$; ""
"
120 INPUT "CADENA 2: ", B$
130 IF LEN B$ > 10 THEN GO TO 120
135 PRINT "CADENA 2: "; B$; ""
"
140 INPUT "CADENA 3: ", C$
150 IF LEN C$ > 10 THEN GO TO 140
155 PRINT "CADENA 3: "; C$; ""
"
160 PRINT " "; A$+B$+C$; " "
170 PRINT " "; A$+C$+B$; " "
180 PRINT " "; B$+A$+C$; " "
190 PRINT " "; B$+C$+A$; " "
200 PRINT " "; C$+A$+B$; " "
210 PRINT " "; C$+B$+A$; " "
"
220 REM *****
*
*  CONTINUACION?        *
*        *****        *
*
230 POKE 23658,8
240 PRINT AT 21,0;"QUIERE CONTI
NUAR (S/N)"
250 PAUSE 0
260 LET D$=INKEY$
270 IF D$="S" THEN CLS : GO TO
100
280 IF D$="N" THEN CLS : STOP
290 GO TO 250

```

mente visualiza en la pantalla todas las posibles combinaciones de concatenación. Introduzca, por ejemplo, su nombre y dos apellidos, como parámetros y observe el resultado.

Subcadenas

Con el término de *subcadena* se designa a un grupo de *caracteres consecutivos* extraídos de una cadena. Por ejemplo "dicc" es una subcadena de "diccionario"; también lo son "onar" y "ario"; sin embargo "dcic", "noar" y "aroi" no lo son, ya que sus caracteres no han sido extraídos consecutivamente.

Otros ejemplos son:

| CADENAS | SUBCADENAS |
|------------------------|------------|
| "Coche de carreras " | "che de" |
| "Angel, Pepe, Luis" | "Pepe" |
| "lunesmartesmiércoles" | "lunes" |
| "Ordenador personal" | "Ordena" |

Fragmentación

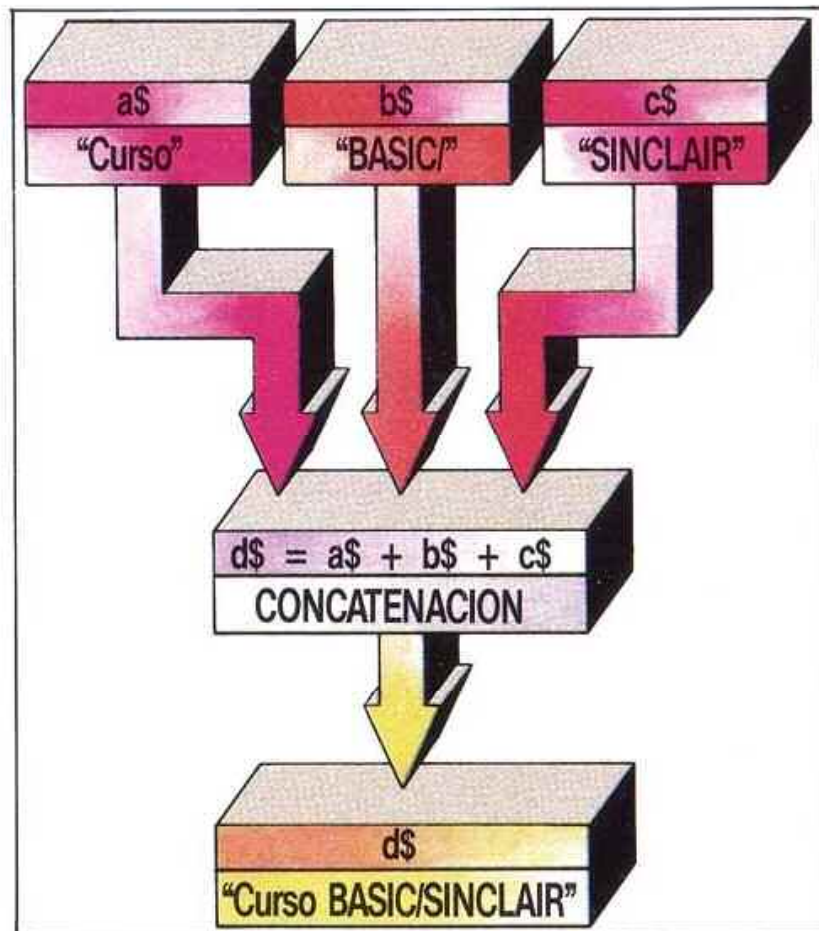
Para extraer una subcadena se utiliza la operación de *fragmentación*. La estructura general es:

| FRAGMENTACION |
|-----------------|
| cadena (a TO b) |

donde «a» es el número del primer carácter a extraer y «b» el último, entre ambos limitan la longitud de la subcadena; «a» y «b» pueden ser constantes o variables numéricas, pero siempre positivas, de lo contrario, el ordenador enviará el mensaje:

B Integer out of range

La cadena puede ser o bien



Concatenación.

una constante o bien una variable.

Ejemplos:

| FRAGMENTACION | RESULTADO |
|------------------------|-----------|
| "Curso BASIC" (1 TO 5) | "Curso" |
| "Monitor TV" (5 TO 9) | "tor T" |
| "MICROHOBBY" (6 TO 10) | "HOBBY" |
| "Cartucho" (4 TO 5) | "tu" |

Si «b» fuera menor que «a», el resultado de la fragmentación sería una cadena vacía.

Ejemplos:

| FRAGMENTACION | RESULTADO |
|-------------------|-------------------|
| "Juan" (2 TO 1) | Cadena vacía ("") |
| "Pepe" (20 TO 10) | Cadena vacía ("") |

Cuando «a» es \emptyset o «b» es mayor que el correspondiente a la longitud de la cadena, el

ordenador envía el siguiente mensaje:

3 Subscript wrong

Ejemplo:

- "Televisor" (\emptyset TO 5)
- "Teclado" (2 TO 3 \emptyset)

El programa número "2" permite la introducción de una cadena de quince caracteres como máximo, es necesario además introducir los límites de la fragmentación. La subcadena resultante es visualizada en la pantalla.

Fragmentación específica

Existen tres tipos de fragmentación que tienen una estructura distinta:

- Fragmentación de un solo carácter.

- Fragmentación izquierda.
- Fragmentación derecha.

Cuando se quiere extraer un solo carácter de una cadena, la estructura es la siguiente:

| |
|---------------|
| FRAGMENTACION |
| cadena (n) |

donde «n» es el número de carácter; es equivalente a escribir «cadena» (n TO n).

Ejemplos:

| FRAGMENTACION | RESULTADO |
|-------------------|-----------|
| "Cassette" (5) | "e" |
| "Cable" (2) | "a" |
| "Circular" (3) | "r" |
| "Cigarrillos" (8) | "l" |

La fragmentación izquierda consiste en la extracción de los «n» primeros caracteres de una cadena, su estructura es:

| |
|---------------|
| FRAGMENTACION |
| cadena (TO n) |

equivale a «cadena» (1 TO n)

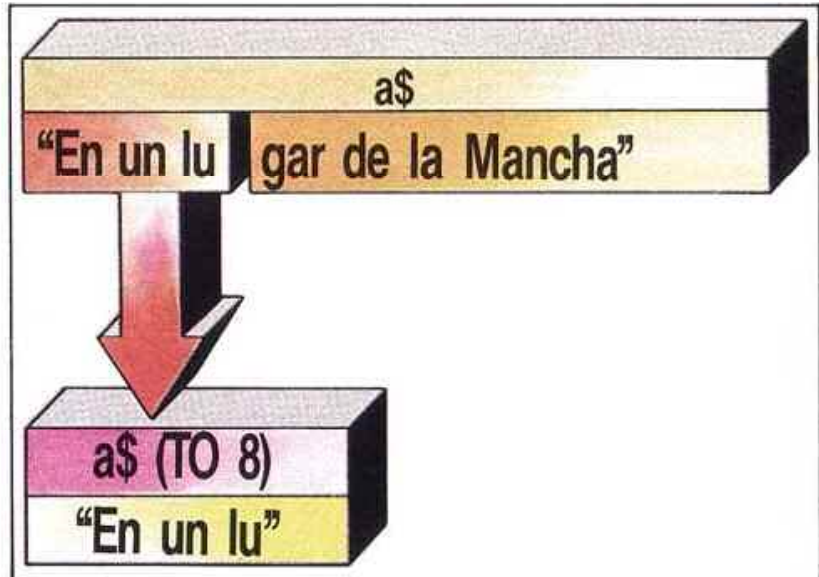
Ejemplos:

| FRAGMENTACION | RESULTADO |
|-------------------|-----------|
| "CARAMELO" (TO 3) | "CAR" |
| "coche" (TO 2) | "co" |
| "Botella" (TO 5) | "Botel" |
| "Corazon" (TO 4) | "Cora" |

La fragmentación derecha permite extraer los últimos caracteres de una cadena a partir de «n», su estructura es la siguiente:

| |
|---------------|
| FRAGMENTACION |
| cadena (n TO) |

es equivalente a «cadena» (n TO fin).



Fragmentación izquierda.

PROGRAMA 2

```

10 REM *****
   *          *
   *   CURSO   *
   * BASIC/SINCLAIR *
   *          *
   *          *
   * FRAGMENTACION *
   *          *
   *          *
   *          *
20 BORDER 1: PAPER 1: INK 7: C
L$
25 POKE 23658,8
30 REM *****
   *          *
   * INTRODUCCION *
   *          *
   *   CADENA   *
   *          *
   *          *
   *          *
100 INPUT "CADENA: ",A$
105 IF A$="" THEN GO TO 100
110 IF LEN A$>15 THEN GO TO 100
115 PRINT "CADENA :";A$;" "
120 LET longitud=LEN (A$)
125 PRINT "LONGITUD: ";longitud
130 REM *****
   *          *
   * INTRODUCCION *
   *          *
   *   LIMITES   *
   *          *
   *          *
   *          *
140 PRINT AT 21,0;"LIMITE INFER
IOR (1-";longitud;"):"
145 INPUT LINE I$
150 IF I$<"1" OR I$>"9" THEN GO
TO 145

```

```

160 IF VAL (I$)>longitud THEN G
0 TO 145
162 LET inferior=VAL (I$)
164 PRINT AT 4,0;"LIMITE INFERI
OR: ";inferior
170 PRINT AT 21,7;"SUP"
175 INPUT LINE S$
180 IF S$<"1" OR S$>"9" THEN GO
TO 175
190 IF VAL (S$)>longitud THEN G
0 TO 175
192 LET superior=VAL (S$)
194 PRINT AT 5,0;"LIMITE SUPERI
OR: ";superior
196 PRINT AT 21,0;";

200 REM *****
*
*   VISUALIZACION   *
*
*   SUBCADENA       *
*
* *****

210 PRINT AT 8,0;"SUBCADENA: ""
";A$(inferior TO superior);""""
220 REM *****
*
* CONTINUACION?    *
*
* *****

225 POKE 23658,8
230 PRINT AT 21,0;"QUIERE CONTI
NUAR (S/N)"
240 PAUSE 0
250 LET D$=INKEY$
260 IF D$="S" THEN CLS : GO TO
100
270 IF D$="N" THEN CLS : STOP
280 GO TO 240

```

Ejemplos:

| FRAGMENTACION | RESULTADO |
|-------------------|-----------|
| "Zumos" (3 TO) | "mos" |
| "pescados" (5 TO) | "ados" |
| "Puros" (2 TO) | "uros" |
| "A12557" (4 TO) | "557" |

Asignación de subcadenas

Una variable de cadena puede ser modificada parcialmente utilizando la asignación de subcadenas. Supongamos que la variable a\$ tiene asignado un valor.

"Evolucion de los anfibios" utilizando la expresión:
a\$ (18 TO 25) = "caballos"
la variable a\$ tendrá un nuevo valor

"Evolucion de los caballos"
Cuando a una subcadena se le asigna una longitud de caracteres mayor que los expresados, automáticamente los caracteres sobrantes, de la derecha, son recortados.

Ejemplo:
a\$ = "*****"
asignando

a\$ (2 TO 5) = "\$\$\$\$\$"
la variable a\$ tendrá el valor
"*\$\$\$\$*****"

Cuando por el contrario la longitud sea inferior, la subcadena se rellena con espacios.

Ejemplo:
a\$ = "*****"
asignando
a\$ (2 TO 5) = "\$\$"
a\$ tendrá el valor
"*\$\$ *****"

Este método de recortar o rellenar con espacios se llama «asignación procustea». Utilizando este método se puede asignar un nuevo valor a toda la variable manteniendo la longitud de ésta.

Ejemplo:

a\$ = "*****"

asignando

a\$ () = "\$\$\$"

el nuevo valor de a\$ será
"\$\$\$"

Comparación de cadenas

Las cadenas al igual que los números pueden compararse haciendo uso de los operadores relacionales. Para averiguar si una cadena es mayor, menor o igual a otra, se comparan los códigos "ASCII" correspondientes al primer carácter de cada una de ellas; si estos son diferentes, será mayor la que tenga un código más alto.

Ejemplo 1:



| LETRA | ASCII |
|-------|-------|
| P | 80 |
| G | 71 |

P < G

luego a\$ > b\$

Ejemplo 2:



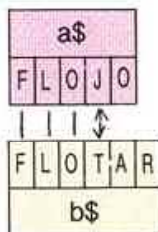
| LETRA | ASCII |
|-------|-------|
| B | 66 |
| b | 98 |

B < b

luego a\$ < b\$

Si fueran iguales los códigos del primer carácter, sería necesario pasar a comparar los siguientes hasta encontrar uno diferente.

Ejemplo:



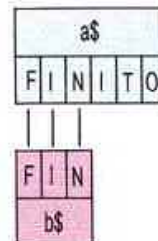
| LETRA | ASCII |
|-------|-------|
| J | 74 |
| T | 84 |

J < T

luego a\$ < b\$

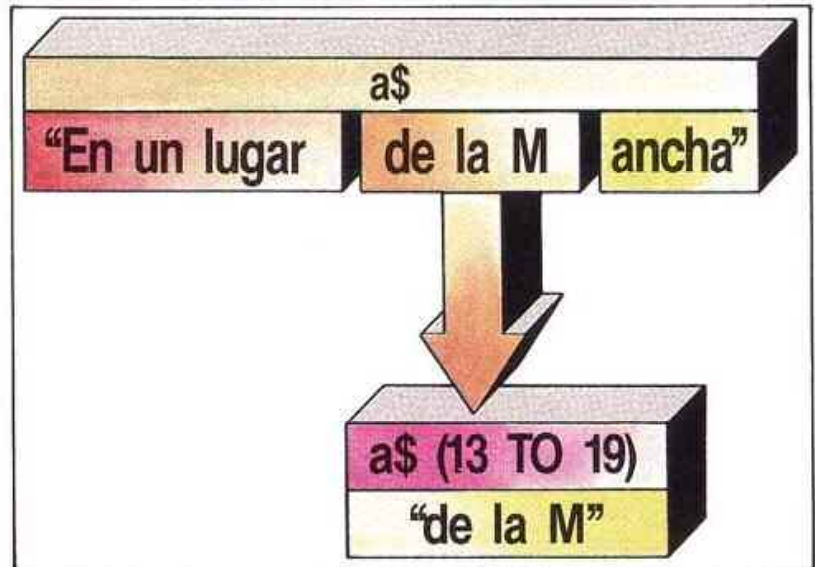
En este proceso de comparación puede ocurrir que los caracteres de una cadena se acaben antes que los de otra, en este caso, es mayor la que tenga mayor longitud.

Ejemplo:



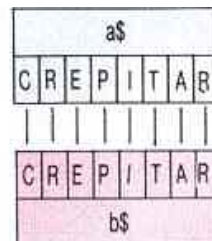
luego a\$ > b\$

Si las dos cadenas se acaban sin encontrar ningún carácter distinto, significa que ambas son iguales.



Fragmentación central.

Ejemplo:



luego a\$ = b\$

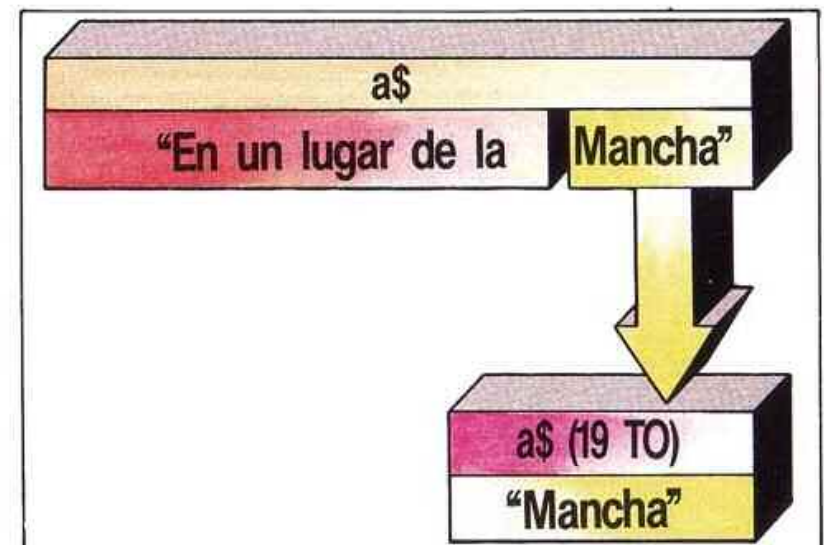
El carácter «espacio» también tiene su correspondiente código ASCII, por este motivo, al comparar cadenas alfanuméricas debe ser tenido en

cuenta. Aparentemente, en el siguiente ejemplo, las dos cadenas son iguales, sin embargo no lo son, ya que una de ellas contiene al final un espacio.

Ejemplo:



luego a\$ < b\$



Fragmentación derecha.

PROGRAMA 3

```

10 REM *****
*
*      CURSO      *
* BASIC/SINCLAIR *
* "COMPARACION"  *
*
* *****
LS 20 BORDER 1: PAPER 1: INK 7: C
50 REM *****
*
* ENTRADA DE DATOS *
*
* *****

100 INPUT "CADENA 1: ";A$
110 IF LEN A$>10 THEN GO TO 100
115 PRINT "CADENA 1: "A$;" "

"
120 INPUT "CADENA 2: ";B$
130 IF LEN B$>10 THEN GO TO 120
" 135 PRINT "CADENA 2: "B$;" "

138 REM *****
*
* COMPARACION *
*
* *****

140 IF A$>B$ THEN PRINT " "A$;"
" > " "B$;" "
150 IF A$<B$ THEN PRINT " "A$;"
" < " "B$;" "
160 IF A$=B$ THEN PRINT " "A$;"
" = " "B$;" "
200 REM *****
*
* CONTINUACION? *
*
* *****

210 POKE 23658,8
220 PRINT AT 21,0;"QUIERE CONTI
NUAR (S/N) "
230 PAUSE 0
240 LET D$=INKEY$
250 IF D$="S" THEN CLS : GO TO
100
260 IF D$="N" THEN CLS : STOP
270 GO TO 230

```

El programa número «3» compara dos cadenas cualesquiera, introducidas por el teclado, de un máximo de diez caracteres cada una, y visualiza en la pantalla el resultado de la comparación.

Ordenación de cadenas

Comparar cadenas puede tener utilidad, por ejemplo, en

la ordenación por orden alfabético de un fichero de nombres. Por sucesivas comparaciones de los elementos del fichero se puede llegar a la ordenación de las cadenas.

Veamos por pasos cómo se ordena un fichero que contiene los siguientes datos:

"PEPE", "JUAN", "ANTONIO",
"RODRIGO" y "DAVID"

como se puede observar, desordenados alfabéticamente.

PRIMER PASO:

El primer dato leído es "PEPE", este lo colocamos provisionalmente en la primera posición.

SEGUNDO PASO:

El dato "JUAN" se compara con "PEPE".

"JUAN" < "PEPE"

"PEPE" se transfiere a la segunda posición.

TERCER PASO:

"ANTONIO" es el tercer dato leído y se compara con "JUAN" y "PEPE".

"ANTONIO" < "JUAN" < "PEPE"

Las cadenas "JUAN" y "PEPE" se desplazan para que "ANTONIO" ocupe la primera posición.

CUARTO PASO:

El código correspondiente a "RODRIGO" es superior al de los demás.

"RODRIGO" > "PEPE" >
"JUAN" > "ANTONIO"

"RODRIGO" se coloca en la cuarta posición, a continuación de los demás.

QUINTO PASO:

El último dato leído es "DAVID" y tiene un código superior al de "ANTONIO" e inferior al resto.

"DAVID" > "ANTONIO"
y
"DAVID" < "JUAN" < "PEPE" < "RODRIGO"

"JUAN", "PEPE" y "RODRIGO" tienen que desplazarse para que "DAVID" ocupe la segunda posición.

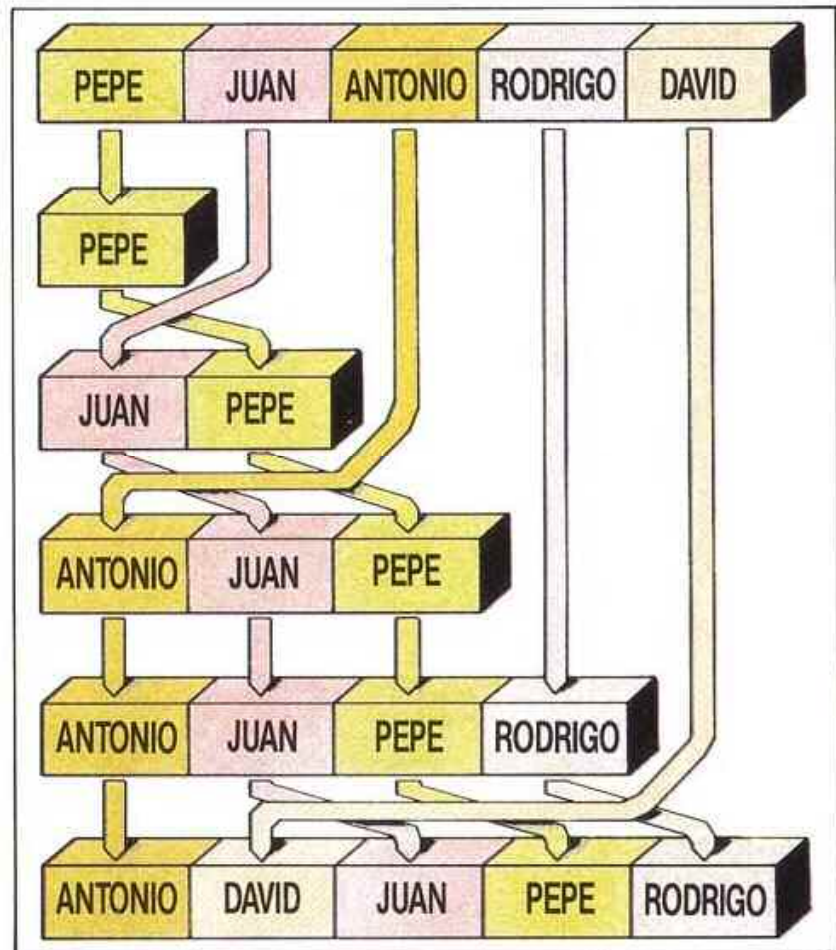
Después de este último paso al fichero queda ordenado de la siguiente manera:

"ANTONIO", "DAVID", "JUAN", "PEPE" y "RODRIGO"

El programa «4» permite la introducción de 20 nombres, de quince caracteres como máximo, y ordenarlos según se van introduciendo.

Prioridades

En el capítulo dedicado al cálculo de expresiones, se mencionó que al calcular el ordenador las operaciones aritméticas, tenía en cuenta la *prioridad* que éstas tenían. Una vez repasadas todas las operaciones (aritméticas, lógicas, de cadena, etc...) se ofrece en la página 38 un resumen de lo revisado. El ordenador asigna las prioridades con un número comprendido entre 1 y 16.



Metodología de ordenación de cadenas.

PROGRAMA 4

```

10 REM .....
    CURS0
    BASIC/SINCLAIR
    "ORDENA"
    .....

15 BORDER 2 PAPER 2 INK 7: C
LS
20 DRAW 255,0 DRAW 0,175 DRA
U -255,0 DRAW 0,-175
25 PRINT AT 5,7:"PROGRAMA "OR
DENA"
30 PRINT FLASH 1,AT 12,0:"PARE
LA CINTA"
35 PLOT 0,23 DRAW 255,0
40 PRINT AT 20,1:"
42 RESTORE
45 FOR X=1 TO 25
50 READ LOG0
55 PRINT CHR$(LOG0) BEEP 0,05
LOG0/2
60 NEXT X GO SUB 90
65 DATA 127,32,77,73,67,82,79,
72,79,66,66,69,32,35,32,82,65,70

65,69,76,32,80,82,65,68,69,80
70 PRINT AT 12,9:"
75 PRINT #0,AT 1,1:"Pulse una
tecla para continuar"
80 PAUSE 0 GO TO 100
82 REM
    .....
    SUB. TEMPORIZACION
    .....

90 FOR X=1 TO 300 NEXT X: RET
URN
100 BORDER 1: PAPER 1: INK 6: C
LS
110 REM
    .....
    DEFINICION DE
    VARIABLES
    .....

150 REM .....
    NUMERACION CADENAS
    .....

160 FOR Y=1 TO 20
170 PRINT AT Y,0:"CADENA ",Y,
180 IF Y=10 THEN PRINT " "
190 PRINT "
195 NEXT Y
200 REM
    .....
    ENTRADA
    PRIMERA CADENA
    .....

210 INPUT "CADENA ",D$
220 IF LEN (D$) < longitud THEN G
O TO 210
230 LET N$(1)=D$
240 PRINT AT 1,11,N$(1)
250 REM
    .....
    ENTRADA CADENAS
    .....

```

```

*****
260 FOR X=2 TO orden
270 INPUT "CADENA: ";D$
280 IF LEN (D$)>longitud THEN G
O TO 270
290 REM
*****
*
* RELLENO ESPACIOS *
*
*****
300 FOR Y=1 TO longitud-LEN (D$
)
310 LET D$=D$+" "
320 NEXT Y
325 REM
*****
*
* COMPARA CADENAS *
*
*****
330 IF D$<N$(X-1) THEN GO SUB 1
000: GO TO 360
340 LET N$(X)=D$
350 PRINT AT X,11,D$
360 NEXT X
370 REM
*****
*
* CONTINUACION? *
*
*****
380 POKE 23658,8
390 PRINT #0,AT 0,0;"QUIERE CON
TINUAR (S/N)"
400 PAUSE 0
410 LET D$=INKEY$
420 IF D$="S" THEN GO TO 500
430 IF D$="N" THEN CLS : STOP
440 GO TO 400

500 REM
*****
*
* BORRADO CADENAS *
*
*****
510 FOR Y=1 TO 20
520 PRINT AT Y,11;"
525 LET N$(Y)=""
530 NEXT Y
540 GO TO 210
1000 REM
*****
*
* SUB. DETECCION *
*
*****
1010 FOR Z=X-1 TO 1 STEP -1
1020 IF D$>N$(Z) THEN GO TO 1050
1030 NEXT Z
1040 REM
*****
*
* ORDENA *
*
*****
1050 FOR Y=orden-1 TO Z+1 STEP -
1
1060 LET N$(Y+1)=N$(Y)
1070 PRINT AT Y+1,11,N$(Y)
1080 NEXT Y
1090 REM
*****
*
* INSERTA *
*
*****
1100 LET N$(Y+1)=D$
1110 PRINT AT Y+1,11,D$
1120 RETURN

```

Elaboración de programas

Antes de introducir las instrucciones necesarias para elaborar un programa, es necesario haber efectuado antes una serie de pasos. Existen diversas técnicas para ayudar al diseñador de programas a estructurar en alguna medida su trabajo y así poder conseguir unos resultados satisfactorios.

Básicamente todas las técnicas tienden a examinar globalmente el programa desde una perspectiva superior, e ir descendiendo, poco a poco, hasta llegar a elaborarlo por completo. En este proceso se pueden distinguir tres fases:

- Análisis del programa
- Síntesis.
- Representación gráfica.

Análisis

El primer paso es analizar el problema y determinar las funciones de nuestro programa, es decir, lo que deseamos que haga. Una vez conocido esto, podremos averiguar cuáles son los datos que debemos proporcionar al programa. Veamos un ejemplo.

Supongamos que deseamos confeccionar un programa que visualice en pantalla o nos saque por impresora un listado ordenado de nuestra agenda telefónica, ¿cuáles van a ser los datos que vamos a necesitar?, en principio los «nombres» y los «teléfonos» de nuestras amistades, con esta información se confecciona un boceto similar al de la figura.



Simbología utilizada en los diagramas de flujo.

En un paso siguiente podremos determinar los programas que vamos a necesitar; siguiendo con nuestro ejemplo necesitaremos:

- Un programa o rutina que nos permita la introducción de los datos por primera vez.
- Otro para poder introducir o modificar nuevos teléfonos.
- Un programa que ordene por orden alfabético los nombres y los visualice.
- Y ya por último, un programa que nos permita seleccionar cualquiera de los anteriores.

Síntesis

Las funciones de cada uno de los programas mencionados anteriormente, pueden ser esquematizados usando la técnica desarrollada por la compañía IBM y denominada HIPO, iniciales de «Hierarchy Input/Process/Output», que traducido al español significa Jerarquía Entrada/Proceso/Salida. Esta técnica consiste en definir un programa mediante tres bloques principales.

- Entrada: definición de los datos a utilizar.
- Proceso: descripción esquematizada de los procesos o cálculos a realizar.
- Salida: especificación de los datos a imprimir o visualizar.

En fases posteriores cada bloque se desarrolla en otros del mismo tipo, más completos, pero todavía sin llegar al detalle.

Representación gráfica

Es el último paso antes de la edición de un programa. Es-

PROGRAMA 1

```

10 REM *****
*
*          CURSO          *
*
* BASIC/SINCLAIR        *
*
*          LISTIN        *
*
*****

12 BORDER 2: PAPER 2: INK 6: C
LS
14 PRINT FLASH 1; AT 7,10; "PARE
LA CINTA"
16 PLOT 104,56: DRAW 16,31: DR
AW 15,0: DRAW 16,-31: DRAW -47,0
18 CIRCLE 128,71,7
20 PLOT 96,80: DRAW 15,0: DRAW
7,11: DRAW 19,0: DRAW 7,-11: DR
AW 15,0: DRAW -17,17: DRAW -30,0
: DRAW -17,-17
22 PLOT 144,72: DRAW 24,0: DRA
W 0,-41: DRAW -80,0
24 PRINT INVERSE 1; AT 16,13; "L
ISTIN"
26 PLOT 0,0: DRAW 0,175: DRAW
255,0: DRAW 0,-175: DRAW -255,0
28 PLOT 0,152: DRAW 255,0
29 RESTORE
30 PRINT AT 1,1; " ";
31 FOR X=1 TO 28
32 READ chr
34 PRINT CHR$ chr; : BEEP 0.05,
chr/2
36 NEXT x
38 DATA 127,32,77,73,67,82,79,
72,79,66,66,89,32,38,32,82,65,70,
65,69,76,32,80,82,65,68,69,83
40 PRINT AT 7,10; "
"
42 PRINT #0; AT 1,1; "Pulse una
tecla para continuar"
44 PAUSE 0
46 BEEP 0.2,20

100 REM *****
*
* DEFINICIONES *
*
*****

110 BORDER 1: PAPER 1: INK 7: C
LS
120 DIM N$(20,22)
130 LET PIP=23609
140 LET FLAGS2=23658
145 LET datos=0
150 LET ordena=0
160 GO TO 500
200 REM

```

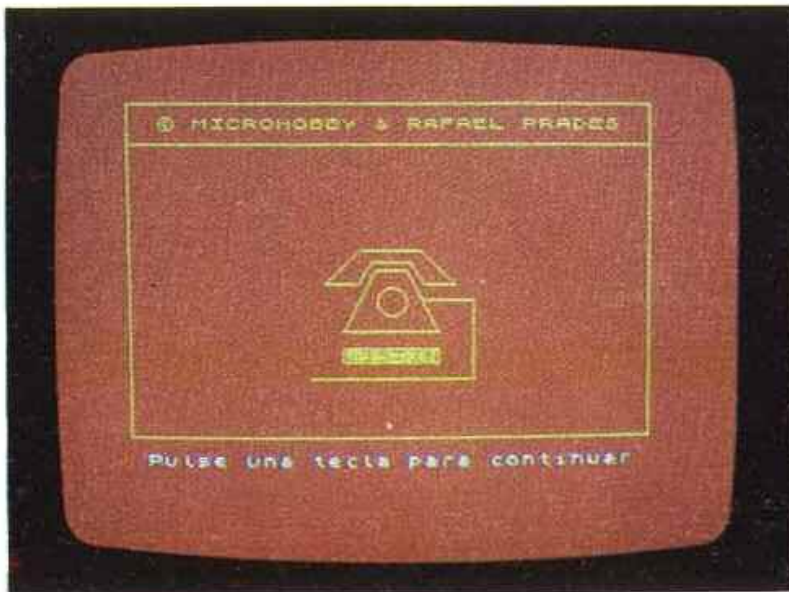
```

*****
*
*  NUEVO LISTIN  *
*
*****

206 CLS
208 DIM N$(20,22)
210 FOR Y=1 TO 20
220 GO SUB 900
230 IF D$="FIN" THEN GO TO 350
250 LET N$(Y,1 TO 15)=D$
260 PRINT D$,
270 GO SUB 920
320 LET N$(Y,16 TO 22)=D$
330 PRINT D$
340 NEXT Y
350 GO SUB 5000
355 CLS
360 GO SUB 5200
370 LET datos=1
375 LET ordena=0
380 GO TO 500
500 REM

*****
*
*  OPCIONES  *
*
*****

```



Carátula del listin.

ta representación utiliza una serie de símbolos standard para facilitar su análisis por otras personas. El gráfico resultante se denomina «Diagrama de flujos», traducción de la palabra inglesa «Flow chart».

Los símbolos utilizados indican las operaciones que de-

be realizar el ordenador en cada momento.

Hay un símbolo que indica cuál es el comienzo o final de un programa. Otro representa a las operaciones genéricas, como por ejemplo, un cálculo, una asignación de valores, etc. El símbolo de *decisión* (rombo

invertido) realiza una comparación entre dos o más valores y, en función del resultado, elige una de las dos salidas posibles. La operación manual se refiere a toda tarea que el ordenador no pueda efectuar y que se necesite de una persona para realizarla, por ejemplo, poner en funcionamiento una impresora, introducir una cinta para su lectura o grabación, etc.

La operación genérica entrada/salida representa a cualquier función de lectura, grabación, impresión, etc., es decir, de entrada o salida de datos. Cuando el diagrama es bastante grande o complejo y se dibuja en varias hojas, la unión de éstas se realiza con el símbolo denominado *conector*, dentro de éste debe figurar un número para utilizarlo como referencia.

Dentro de los símbolos debe incluirse una descripción corta y clara de la operación, cálculo, decisión, etc. que realicen. Los símbolos deben unirse mediante líneas con unas flechas que indiquen la dirección de ejecución del programa.

Programa

El programa "LISTIN" (número 4) realiza las funciones expresadas en los diagramas de flujo que acompañan este artículo. Una vez editado gráfico en cinta con el siguiente comando directo:

```
SAVE "LISTIN" LINE 10
```

de esta forma se autoejecuta al cargarlo.

Después de la presentación aparece un menú con diversas opciones. La número 1 permite crear un fichero con los

```

*****
502 POKE PIP,100
505 POKE FLAG52,8
506 CLS
510 PRINT AT 4,10;"OPCIONES"
520 PRINT AT 8,5;"1.- NUEVO LIS
TIN"
530 PRINT AT 12,5;"2.- MODIF. L
ISTIN"
540 PRINT AT 16,5;"3.- LISTIN"
545 PRINT AT 20,5;"4.- FIN"
550 PAUSE 0: LET P%=INKEY$
560 IF P%="1" THEN BEEP 0.05,20
GO TO 200
570 IF P%="2" THEN BEEP 0.05,20
GO TO 650
580 IF P%="3" THEN BEEP 0.05,20
GO TO 1000
585 IF P%="4" THEN BEEP 0.05,20
POKE PIP,0: CLS: STOP
590 BEEP 0.2,-15: GO TO 550

650 REM
*****
*
* MODIFICACION *
*
*
*****

660 IF datos=1 THEN GO TO 690
670 GO SUB 5500
680 LET datos=1
690 CLS
695 LET ordena=0
700 GO SUB 2000
740 FOR X=1 TO ultimo
745 IF X<10 THEN PRINT " ";
750 PRINT X; " ";N$(X,1 TO 15);"
";N$(X,16 TO 22)
760 NEXT X
770 PRINT #0;AT 0,0;"M=Modifica
/I=Inserta / F=Fin"
780 PAUSE 0: LET P%=INKEY$
782 IF P%="M" THEN BEEP 0.05,20
GO TO 800
784 IF P%="I" THEN BEEP 0.05,20
GO TO 830
786 IF P%="F" THEN BEEP 0.05,20
GO TO 890
788 BEEP 0.2,-15: GO TO 780
800 INPUT "Que linea ? "; LINE
P$
802 IF P%="" THEN GO TO 800
804 IF LEN P$>2 THEN BEEP 0.2,-
15: GO TO 800
806 FOR X=1 TO LEN P$
808 IF P$(X)<"0" OR P$(X)>"9" T
HEN BEEP 0.2,-15: GO TO 800
810 NEXT X
812 LET linea=VAL (P$)
814 IF P$<"1" OR linea>ultimo T
HEN BEEP 0.2,-15: GO TO 800
816 GO SUB 900
818 LET N$(linea,1 TO 15)=D$
820 PRINT AT linea-1,3;N$(linea
,1 TO 15);
822 GO SUB 920
824 LET N$(linea,16 TO 22)=D$
826 PRINT " ";N$(linea,16 TO 22
)
828 GO TO 770
830 IF ultimo=20 THEN GO TO 850
831 LET ultimo=ultimo+1
832 PRINT #0;AT 0,0;"
834 GO SUB 900
836 LET N$(ultimo,1 TO 15)=D$
837 IF ultimo<10 THEN PRINT AT
ultimo-1,0;" ";ultimo;" "; GO T
O 839
838 PRINT AT ultimo-1,0;ultimo;
" ";
839 PRINT N$(ultimo,1 TO 15);"
";
840 GO SUB 920
842 LET N$(ultimo,16 TO 22)=D$
844 PRINT N$(ultimo,16 TO 22)
848 GO TO 770

```

```

850 BEEP 0.2,-15
855 PRINT #0;AT 0,0;"No puede i
nserir mas nombres"
860 FOR T=1 TO 200: NEXT T
870 GO TO 770
890 PRINT #0;AT 0,0;"
891 GO SUB 5000
892 CLS
894 GO SUB 5200
896 GO TO 500
900 REM
*****
*
* ENTRADA NOMBRE *
*
*
*****

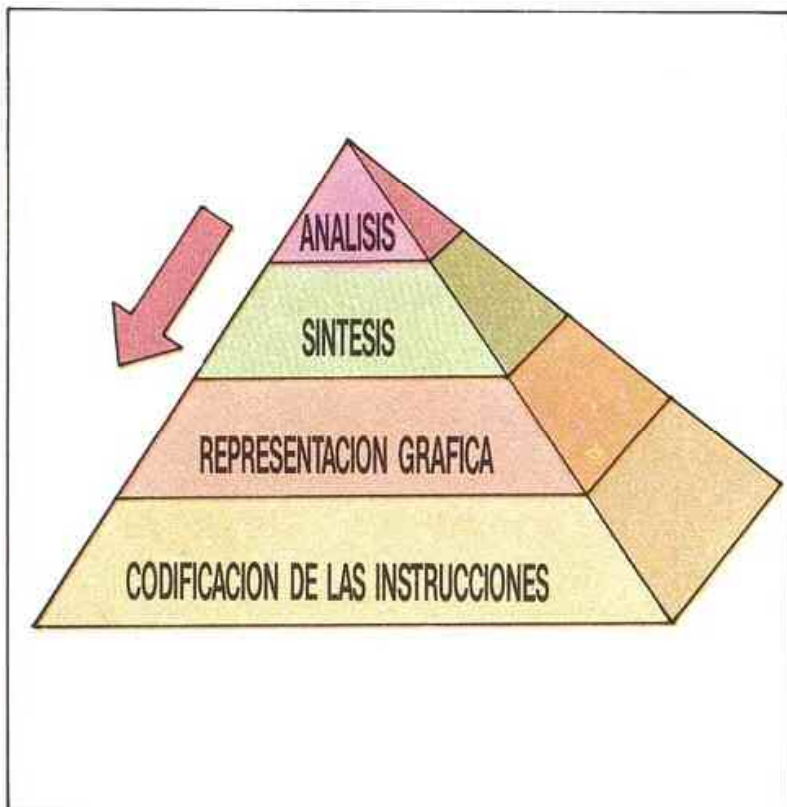
901 INPUT "NOMBRE: "; LINE D$
902 IF D$="" THEN GO TO 900
904 IF LEN D$>15 THEN BEEP 0.2,
-15: GO TO 900
906 FOR X=1 TO LEN D$
908 IF D$(X)<" " AND (D$(X)<"A
" OR D$(X)>"Z") THEN BEEP 0.2,-1
5: GO TO 900
910 NEXT X
912 RETURN
920 REM
*****
*
* ENTRADA TELEF. *
*
*
*****

921 INPUT "TELEFONO: "; LINE D$
922 IF D$="" THEN GO TO 920
924 IF LEN D$>7 THEN BEEP 0.2,-
15: GO TO 920
926 FOR X=1 TO LEN D$
928 IF D$(X)<"0" OR D$(X)>"9" T
HEN BEEP 0.2,-15: GO TO 920
930 NEXT X
932 RETURN

1000 REM
*****
*
* LISTIN *
*
*
*****

1002 IF datos=1 THEN GO TO 1015
1010 GO SUB 5500
1012 LET datos=1
1015 CLS
1016 IF ordena=1 THEN GO TO 1060
1018 PRINT AT 10,8;"ESPERE UN MO
MENTO"
1020 GO SUB 2000
1030 DIM C$(20,22)
1032 LET C$(1)=N$(1)
1034 FOR X=2 TO ultimo
1036 IF N$(X)<C$(X-1) THEN GO TO
1040
1038 LET C$(X)=N$(X): GO TO 1054
1040 FOR Z=X-1 TO 1 STEP -1
1042 IF N$(X)>C$(Z) THEN GO TO 1
046
1044 NEXT Z
1046 FOR Y=(ultimo-1) TO Z+1 STE
P -1
1048 LET C$(Y+1)=C$(Y)
1050 NEXT Y
1052 LET C$(Y+1)=N$(X)
1054 NEXT X
1056 CLS
1058 LET ordena=1
1060 FOR X=1 TO ultimo
1062 PRINT C$(X,1 TO 15);" ";C$(
X,16 TO 22)
1064 NEXT X
1100 GO SUB 5000
1110 GO TO 500
2000 REM
*****
*
* ULTIMO *
*
*
*****

```



Jerarquía «top-down» o descendente.



Fase de análisis.

nombres y teléfonos que deseamos almacenar.

Una vez elegido, el programa nos pide la introducción del primer nombre, y a continuación el teléfono; recuerde que debe pulsar «ENTER» cada vez que introduzca un dato. El nombre debe constar como máximo de 15 caracteres, estos sólo pueden ser o letras mayúsculas o espacios, y el teléfono debe estar constituido como máximo por 7 cifras.

Una vez introducidos, el programa nos pedirá los siguientes y así hasta que se complete el listín, que puede estar formado por 20 nombres como máximo. Si no desea rellenar todo el listín, cuando no tenga más nombres para introducir teclee la palabra FIN.

Posteriormente se nos presenta la opción de sacar un listado por impresora y, a continuación, el programa da las instrucciones necesarias para grabar y verificar una cinta con los datos editados.

Con la opción 2 se puede modificar una cinta ya editada. Lo primero que hace es leer la cinta con los datos. A continuación se nos presenta otro menú con tres opciones, una para poder modificar un nombre ya existente, para lo cual nos pide el número de línea que deseamos modificar. Otra opción permite introducir nuevos nombres, y por último, la tercera imprime un listado y graba una cinta de la misma manera que la opción 1.

La opción 3 es la encargada de ordenar los nombres alfabéticamente y presentarlos en pantalla. Los datos son leídos de la cinta editada. También permite sacar un listado ordenado en impresora.

Para salir del programa seleccione la opción 4.

| ENTRADA | PROCESO | SALIDA |
|-----------------------------------|---------------------------|--------------------|
| INTRODUCCION DE DATOS POR TECLADO | ALMACENAMIENTO EN MEMORIA | GRABACION EN CINTA |

EDITAR UN NUEVO «LISTIN TELEFONICO

| ENTRADA | PROCESO | SALIDA |
|---------------------------|--------------|-----------------|
| LECTURA DE DATOS EN CINTA | MODIFICACION | NUEVA GRABACION |

MODIFICACION DEL «LISTIN»

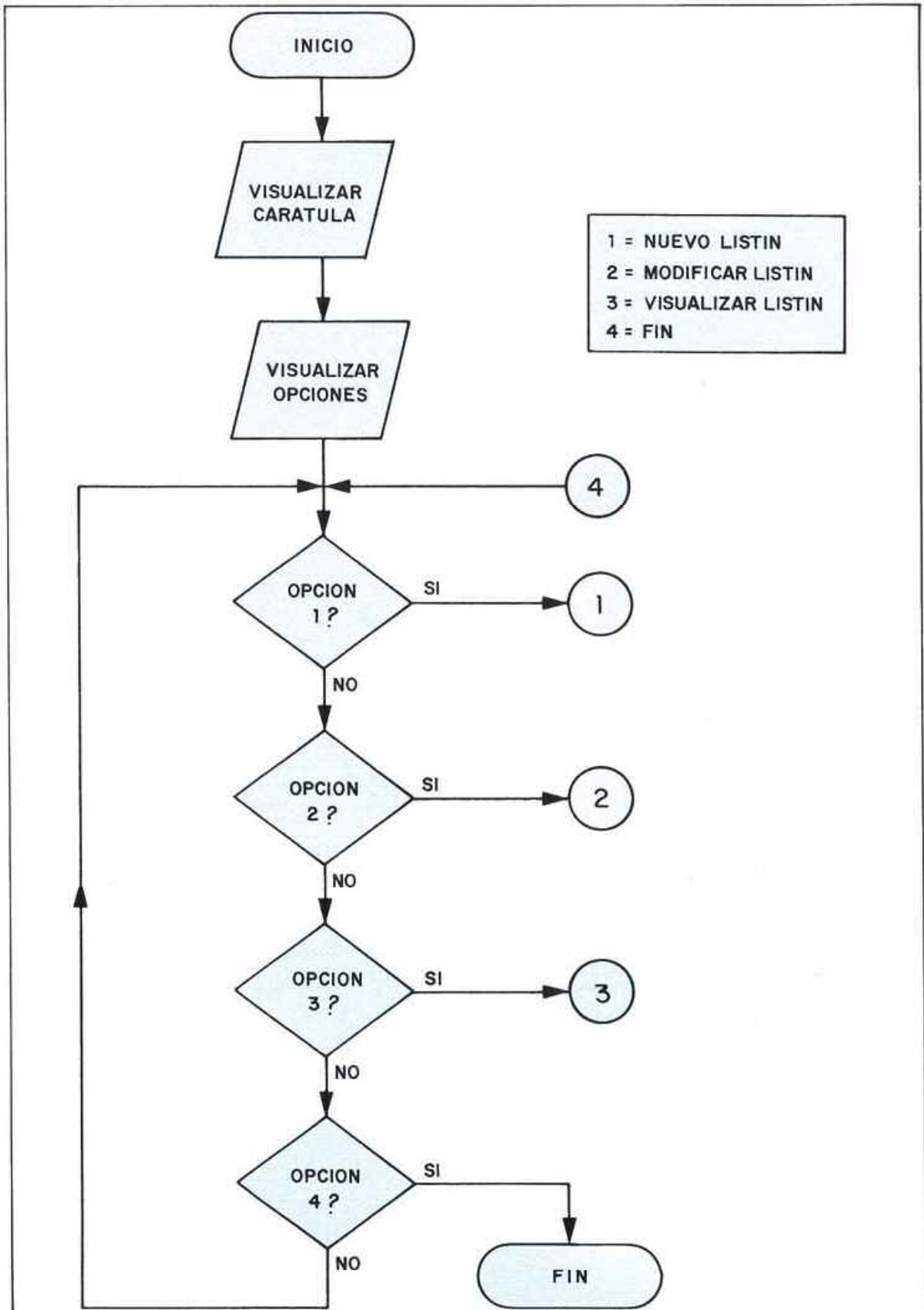
| ENTRADA | PROCESO | SALIDA |
|---------------------------|---------------------|---------------------------|
| LECTURA DE DATOS EN CINTA | ORDENACION DE DATOS | VISUALIZACION O IMPRESION |

VISUALIZACION DEL «LISTIN»

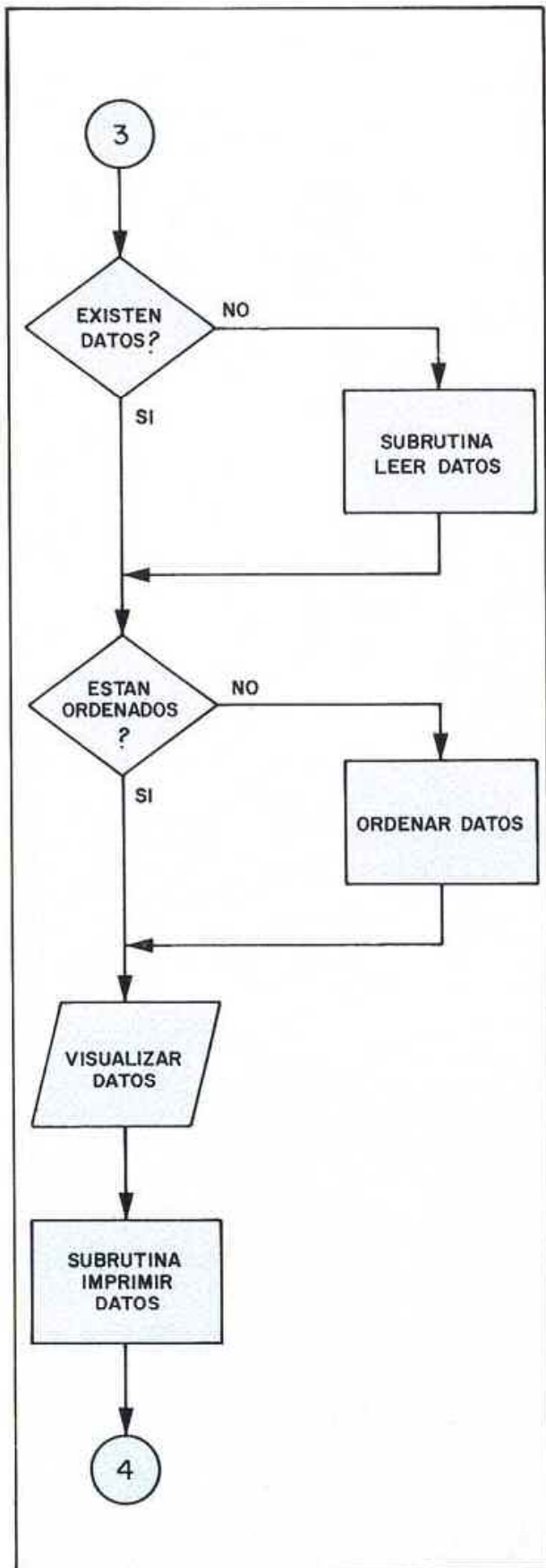
| ENTRADA | PROCESO | SALIDA |
|---------------------------|-----------|---------------------------|
| INTRODUCCION DE LA OPCION | SELECCION | PRESENTACION DE LA OPCION |

MENU DE OPCIONES

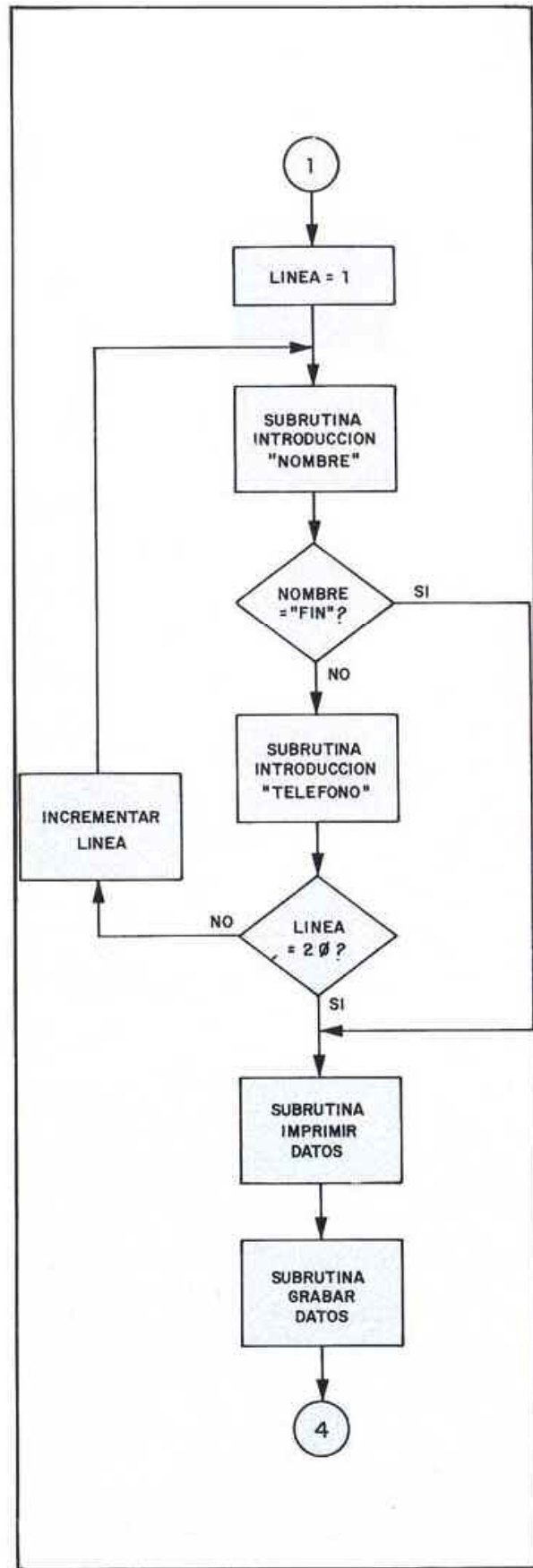
Sintesis.



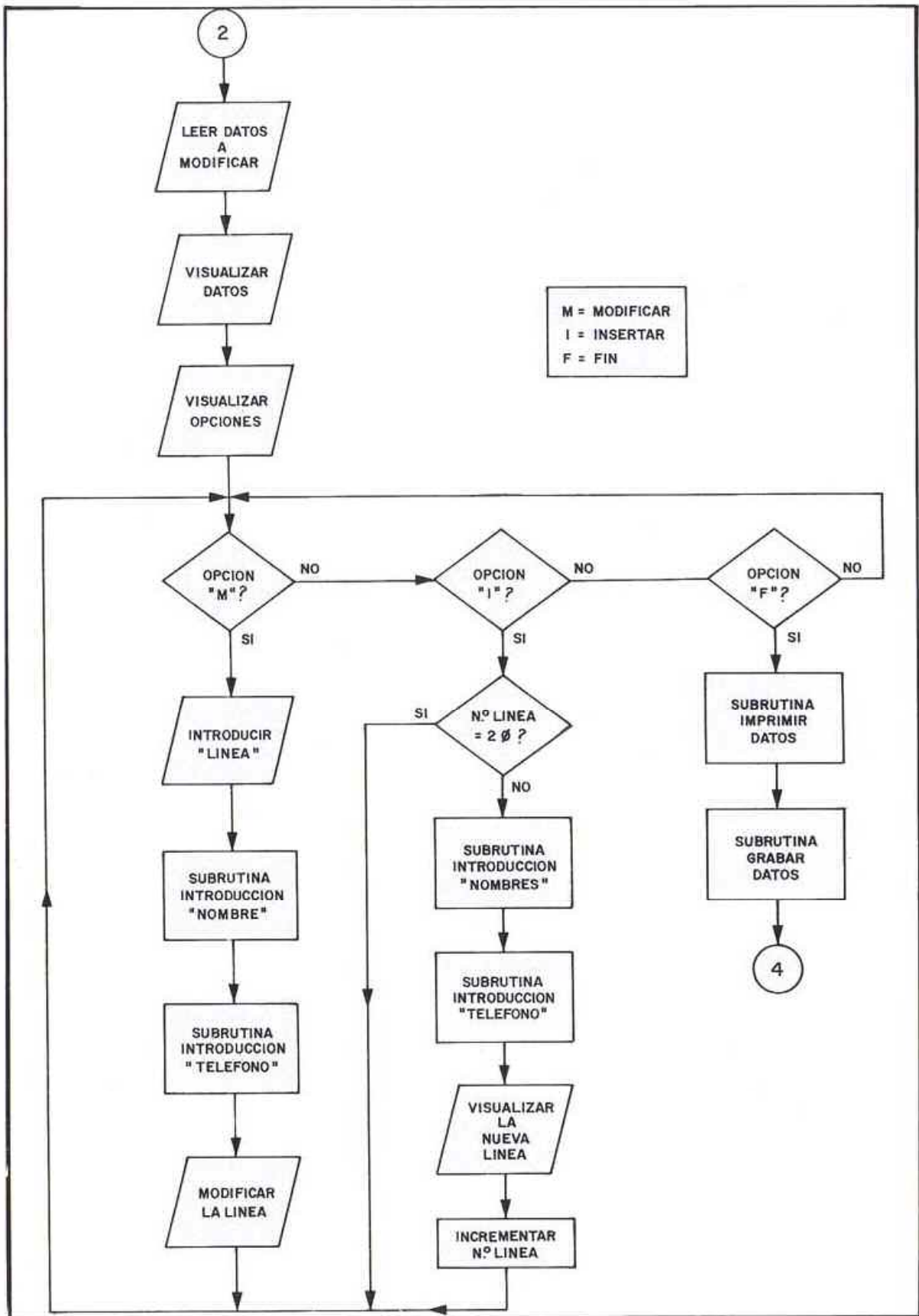
Menú de opciones.



Visualización del «listín telefónico» ordenado.



Editar un nuevo «listín telefónico».



Modificar un «listin telefónico» ya existente.

```

*****
2010 FOR X=1 TO 20
2020 IF N$(X)="
    THEN GO TO 2040
2030 NEXT X
2040 LET ultimo=x-1
2050 RETURN
5000 REM
*****
*
* LISTADO POR *
*
* IMPRESORA *
*
*****
5005 POKE FLAG52,8
5010 PRINT #0;AT 0,0;"Desea un l
istado (S/N)"
5020 PAUSE 0: LET P$=INKEY$
5030 IF P$="S" THEN BEEP 0.05,20
: GO TO 5060
5040 IF P$="N" THEN BEEP 0.05,20
: RETURN
5050 BEEP 0.2,-15: GO TO 5020
5060 PRINT #0;AT 0,0;"Ponga en f
uncionamiento la impresora y pul
se una tecla."
5070 PAUSE 0: BEEP 0.05,20
5080 COPY
5090 PRINT #0;AT 0,0;"Impresion
terminada."
"
5100 FOR x=1 TO 200: NEXT x
5110 GO TO 5010
5200 REM
*****
*
* GRABACION EN *
*
* CINTA *
*
*****
5210 PRINT #0;AT 0,0;"Coloque un
a cinta en el cassettey grabe lo
s datos."
5220 FOR x=1 TO 400: NEXT x
5225 PRINT #0;AT 0,0;"
"
5230 SAVE "telefono" DATA N$(
5240 PRINT #0;AT 0,0;"Grabacion
terminada."
5250 FOR x=1 TO 200: NEXT x
5260 PRINT #0;AT 0,0;"Rebobine l
a cinta (verificacion)"
5270 VERIFY "telefono" DATA N$(
5280 CLS
5290 PRINT #0;AT 0,0;"Grabacion
correcta"
5300 FOR x=1 TO 200: NEXT x
5305 POKE FLAG52,8
5310 PRINT #0;AT 0,0;"Desea grab
ar otra cinta (S/N)"
5320 PAUSE 0: LET P$=INKEY$
5330 IF P$="S" THEN BEEP 0.05,20
: GO TO 5200
5340 IF P$="N" THEN BEEP 0.05,20
: CLS : RETURN
5350 BEEP 0.2,-15: GO TO 5320
5500 REM
*****
*
* LECTURA DE *
*
* DATOS *
*
*****
5505 CLS
5510 PRINT #0;AT 0,0;"Coloque la
cinta con los datos."
5520 LOAD "telefono" DATA N$(
5530 CLS
5540 PRINT #0;AT 0,0;"Lectura co
rrecta."
5550 FOR x=1 TO 200: NEXT x
5560 RETURN

```

El juego de sentencias

Después de haber repasado en capítulos anteriores los conocimientos esenciales del lenguaje de programación BASIC, empezamos en éste a dar una visión global del juego de sentencias del Spectrum; posteriormente irán siendo estudiadas una a una.

Clasificación

El juego de comandos, sentencias y funciones, consta de 88 palabras claves, éstas pueden clasificarse, según su funcionalidad, de la siguiente forma:

- Comandos de control.
- Comandos de programación.
- Comandos de entrada/salida.
- Manejo de cadenas.
- Funciones aritméticas.
- Funciones lógicas.
- Comandos de dibujo.
- Comandos de control de color e impresión.
- Sonido.
- Manejo impresora.
- Manejo periférico (Interface - 1).
- Auxiliares.

Comandos de control

Esta serie de comandos normalmente son introducidos de forma directa en el ordenador, es decir, que no forman parte de las instrucciones de un programa, aunque en algunas ocasiones sí pueden formar.



Algunas de las funciones de estos comandos son:

- Ejecución de un programa.
- Listado por pantalla de las instrucciones.
- Grabación de programas.
- Carga de programas.
- Borrado de pantalla.
- Borrado de memoria.
- Etc...

Comandos de programación

La función principal de estas sentencias es, manejar la información interna, realizar cálculos, tomar decisiones, etc. Algunas de las funciones más concretas son:

- Asignación de variables.
- Dimensionado de Matrices.
- Parada en la ejecución de un programa.
- Temporizaciones.
- Control de bucles.

- Decisiones.
- Salto a una línea determinada.
- Salto a subrutina.
- Etc...

Comandos de entrada/salida

Son todos aquellos que indican al ordenador qué tipo de operación de lectura o escritura de datos debe realizar, como por ejemplo:

- Visualización en pantalla.
- Entrada de datos por teclado.
- Reconocimiento de la última tecla pulsada.
- Lectura de los datos de una tabla.
- Escritura en una posición de memoria.
- Lectura de una posición de memoria.
- Entrada o salida de datos por el conector trasero.

Manejo de las cadenas

útiles en el manejo de cadenas como:

- Conocer la longitud de una cadena.
- Asignar a una variable numérica el valor de una cadena.
- Pasar a decimal el valor de un carácter ASCII.
- Pasar a Código ASCII un número decimal.
- Etc...

Funciones aritméticas

Con esta serie de funciones, el Spectrum se convierte en una útil calculadora científica, ya que permite efectuar las siguientes operaciones:

- Raíz cuadrada.
- Logaritmos naturales.
- Seno.
- Coseno.
- Tangente.
- Arco seno.
- Arco coseno.
- Arco tangente.
- Generación de números aleatorios.
- Conversión de binario a decimal.
- Conocer el signo de una expresión.
- Averiguar su valor absoluto o su parte entera.
- Operar con las constantes matemáticas «pi» y «e».

Funciones lógicas

Al estudiar los operadores lógicos, fueron repasadas estas funciones:

- AND.
- OR.
- NOT.

Comandos de dibujo

Con estas sentencias se puede realizar cualquier tipo de gráfico en *alta resolución*. Se puede dibujar mediante:

PROGRAMA 1

```
10 REM *****
*          CURSO          *
*          BASIC/SINCLAIR *
*          *****
*          "COLOREAR"     *
*          *****
20 BORDER 1: PAPER 1: CLS
30 PRINT #0; AT 1,0; "CURSO BASI
C/SINCLAIR MICROHOBBY"
40 FOR y=0 TO 21
50 FOR x=0 TO 31
60 LET grafico=INT (RND*15)+12
9
70 LET color papel=RND*7
80 LET color tinta=RND*7
90 PRINT AT y,x; PAPER color p
apel; INK color tinta; CHR$ grafi
co
100 NEXT x
110 LET color borde=RND*7
120 BORDER color borde
130 NEXT y
140 PRINT AT 10,4; PAPER 9; INK
8; "EL COLOR EN EL SPECTRUM"
150 PAUSE 0
160 BORDER 7: PAPER 7: INK 0: C
LS
```



- Puntos.
 - Líneas.
 - Circunferencias.
 - Arcos de circunferencia.
- Los argumentos usuales que hay que indicar en este tipo de sentencias son las coor-

denadas «X» e «Y» de la posición a dibujar.

También existe la posibilidad de realizar gráficos, utilizando los *predefinidos*, que incorpora el Spectrum, y los definidos por el usuario (GDU).

PROGRAMA 2

```
10 REM *****
*
*          CURSO          *
*
* BASIC/SINCLAIR *
*
* *****
*          "GRAFICAS"    *
*
* *****

15 BORDER 2: PAPER 2: INK 7: C
LS
20 DRAW 255,0: DRAW 0,175: DRA
W -255,0: DRAW 0,-175
25 PRINT AT 5,7;"PROGRAMA ""GR
AFICAS""
30 PRINT FLASH 1;AT 12,9;"PARE
LA CINTA"
35 PLOT 0,23: DRAW 255,0
40 PRINT AT 20,1;" ";
42 RESTORE
45 FOR X=1 TO 28
50 READ logo
55 PRINT CHR$ logo;: BEEP 0.05
, logo/2
60 NEXT x: GO SUB 750
65 DATA 127,32,77,73,67,82,79,
72,79,66,66,89,32,38,32,82,65,70
,65,69,76,32,80,82,65,68,69,83
70 PRINT AT 12,9;"
75 PRINT #0;AT 1,1;"Pulse una
tecla para continuar"
80 PAUSE 0
85 REM
*****
*
* DIBUJO *
*
* *****

90 BORDER 1: PAPER 1: INK 6: C
LS
100 FOR x=-127 TO 128 STEP 2
110 PLOT OVER 1;127,0
120 DRAW OVER 1;x,175
130 NEXT x
140 GO SUB 750
200 FOR x=-127 TO 128 STEP 2
210 PLOT OVER 1;127,175
220 DRAW OVER 1;-x,-175
230 NEXT x
240 GO SUB 750
300 FOR y=88 TO -87 STEP -2
310 PLOT OVER 1;0,87
320 DRAW OVER 1;255,y
330 NEXT y
340 GO SUB 750
400 FOR y=88 TO -87 STEP -2
410 PLOT OVER 1;255,87
420 DRAW OVER 1;-255,y
430 NEXT y
440 GO SUB 750
```

Comandos de control de color

Pueden definirse hasta ocho colores básicos con estos comandos, aunque pueden simularse hasta 56 más. Con ellos se puede alterar el color de:

- El *borde* de la pantalla (zona donde no se imprimen caracteres).
- El *fondo*, también conocido como *papel* (zona en la cual se visualizan los caracteres o los gráficos).
- El color del propio carácter o gráfico, también conocido como *tinta*.

Estos colores pueden ser controlados en dos gamas de brillo.

La impresión se puede variar con otra serie de comandos que controlan:

- El parpadeo de los caracteres.
- La inversión del color de *tinta* por el de *papel*.
- La sobreimpresión de caracteres.

Sonido

El Spectrum tiene, con el empleo de una sola instrucción, capacidad para producir una amplia variedad de notas musicales; esta posibilidad es bastante interesante, ya que nos permite, con su uso, amenizar los programas. Estos sonidos son escuchados a través del altavoz interno, aunque también es posible enchufar un amplificador a los conectores EAR o MIC para incrementar su volumen.

Manejo impresora

Puede incorporarse una impresora ZX u otra, con el *interface* adecuado, al conector de

```

500 FOR y=175 TO 0 STEP -2
510 PLOT OVER 1;0,y
520 DRAW OVER 1;255,0
530 NEXT y
540 GO SUB 750
600 FOR x=0 TO 255 STEP 1
610 PLOT OVER 1;x,0
620 DRAW OVER 1;0,175
630 NEXT x
640 GO SUB 750
700 GO TO 100
740 REM

```

```

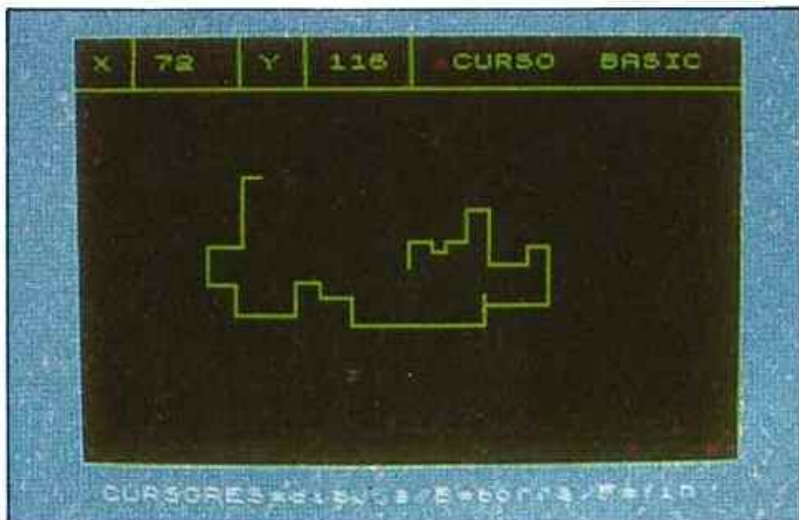
*****
*
* RETARDO *
*
*****

```

```

750 FOR z=0 TO 300
760 NEXT z
770 RETURN

```



expansión trasero. En ésta se pueden obtener:

- Listado de programas.
- Copia de los gráficos presentes en pantalla.
- Mensajes.

INTERFACE - 1

El INTERFACE - 1 es un dispositivo que también se adapta al conector de expansión del Spectrum. Con el manejo de ciertas instrucciones y a través de este interface se puede controlar:

- Un máximo de ocho *Microdrive* (unidades de almacenamiento).
- La *red local* (mediante la cual se pueden atender hasta 64 Spectrum).
- Salida *RS-232* (para conectar periféricos que utilicen este sistema de transmisión de datos en serie).

También prolonga el conector de expansión de manera que se puedan añadir más periféricos, aunque esté conectado el INTERFACE - 1.

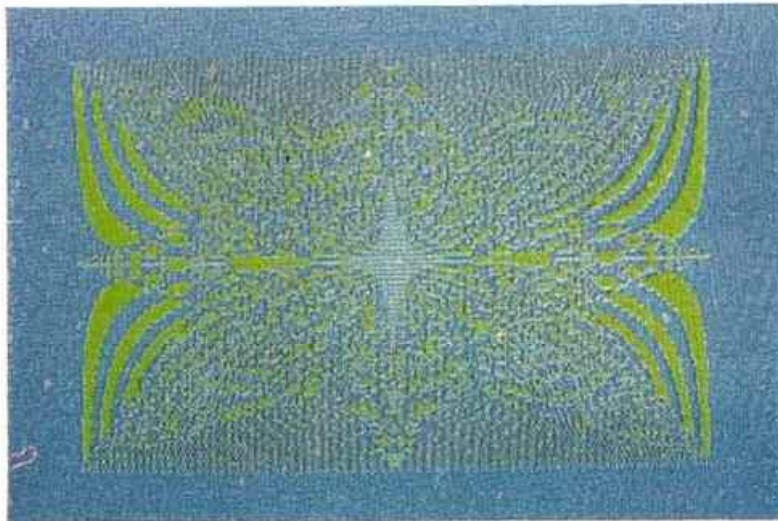
Manejo Microdrive

A través del INTERFACE - 1 se pueden realizar las siguientes operaciones con los Microdrive:

- Formatear o inicializar cartuchos.
- Almacenar y verificar programas en cartucho.
- Cargar programas.
- Borrar ficheros de datos y programas.
- Combinar programas residentes en memoria con los archivados en cartucho.
- Catalogar cartuchos, es decir, obtener un directorio o listado de los programas contenidos en él.
- Abrir y cerrar ficheros de datos.

PROGRAMA 3

```
10 REM *****
*          CURSO          *
* BASIC/SINCLAIR        *
* *****                *
*          G.D.U.        *
* *****                *
20 BORDER 4: PAPER 4: INK 0: C
LS
30 PRINT AT 10,7;"Espere un mo
mento"
40 REM *****
*          GENERACION    *
* *****                *
50 FOR n=USR "A" TO USR "U"+7
60 LET GDU=INT (RND*256)
70 POKE n,GDU
80 NEXT n
90 REM *****
*          VISUALIZACION *
* *****                *
100 CLS
110 FOR n=0 TO 9
120 PRINT CHR$(144+n);" ....."
";CHR$(65+n),CHR$(154+n);" ..
..";CHR$(75+n)
130 NEXT n
140 PRINT AT 21,8;CHR$ 164;" ..
..";CHR$ 85
150 PAUSE 0
```



- Grabar y leer datos de un fichero.
- Mover datos entre ficheros.

Auxiliares

Con estas sentencias se puede controlar el formato de

impresión de alguno de los comandos de entrada/salida; también, por ejemplo, se puede conocer si un punto determinado de la pantalla tiene color de *tinta* o *papel*.

Programas

Los párrafos siguientes son una breve descripción de los programas que acompañan este capítulo, y que vienen a mostrar la capacidad del Spectrum para generar colores, gráficos y sonido.

Programa «COLOREAR»

Este programa muestra en pantalla la capacidad del Spectrum para producir colores en *baja resolución*. Los gráficos predefinidos son generados aleatoriamente, al igual que los colores de borde, papel y tinta.

Una vez ejecutado, pulse cualquier tecla para terminar y aparecerá el mensaje:

Ø OK,16Ø : 4

Cada vez que se ejecuta se obtienen resultados distintos.

Programa «GRAFICAS»

Visualiza en pantalla unas gráficas realizadas en *alta resolución*, empleando la técnica de *sobreimpresión*. Una vez representadas, el programa las vuelve a generar; para salir de este bucle, apriete simultáneamente las teclas «CAPS SHIFT» y «SPACE», el ordenador nos presentará el mensaje:

L BREAK into program

El programa debe ser grabado de la forma:

SAVE "GRAFICAS" LINE 1Ø

PROGRAMA 4

```

10 REM *****
*          CURSO          *
*          BASIC/SINCLAIR *
*****
*          DIBUJANDO      *
*****

20 BORDER 1: PAPER 0: INK 7: C
LS
30 RESTORE 60
40 FOR X=1 TO 26
50 READ character
60 PRINT AT 20,(1+x);CHR$ cara
cter: BEEP 0.05,character/2
70 NEXT x
80 DATA 127,32,77,73,67,82,79,
72,79,66,66,89,32,36,32,82,65,70
,65,69,76,32,80,82,65,68,69,83
110 REM *****
*          PIZARRA        *
*****

120 DRAW 0,175: DRAW 255,0: DRA
W 0,-175: DRAW -255,0
130 REM *****
*          TIZA           *
*****

132 PRINT AT 3,11;"DIBUJANDO"
134 PRINT AT 4,11;"
140 PRINT AT 8,10; INK 4;"U.- U
ERDE"
150 PRINT AT 10,10; INK 5;"C.-
CYAN"
160 PRINT AT 12,10; INK 6;"A.-
AMARILLO"
170 PRINT AT 14,10; INK 7;"B.-
BLANCO"
180 PRINT #0;AT 1,0;"Color de l
a tiza ?"
185 POKE 23658,8
190 PAUSE 0
200 LET A%=INKEY$
210 IF A%="U" THEN LET color=4:
GO TO 260
220 IF A%="C" THEN LET color=5:
GO TO 260
230 IF A%="A" THEN LET color=6:
GO TO 260
240 IF A%="B" THEN LET color=7:
GO TO 260
250 BEEP 0.2,-15: GO TO 190
260 REM *****
*          POSICION      *
*          INICIAL       *
*****

270 BEEP 0.2,20
280 INK color: CLS
290 DRAW 0,175: DRAW 255,0: DRA
W 0,-175: DRAW -255,0

300 PLOT 0,152: DRAW 255,0
310 PLOT 24,152: DRAW 0,22
320 PLOT 54,152: DRAW 0,22
330 PLOT 88,152: DRAW 0,22
340 PLOT 128,152: DRAW 0,22
350 PRINT AT 1,1;"X";AT 1,9;"Y"
360 PRINT AT 1,18;"CURSO BASIC"

370 LET pos x=127
380 LET pos y=80
390 PRINT AT 1,4;pos x;AT 1,12;
pos y
392 PRINT #0;" CURSORES=dibuja/
B=borra/F=fin"
400 REM *****
*          DIBUJAR CON   *
*          CURSORES     *
*****

410 PLOT pos x,pos y
420 IF INKEY$="5" THEN LET pos
x=pos x-1: LET movimiento=1: GO
TO 490
430 IF INKEY$="6" THEN LET pos
y=pos y-1: LET movimiento=0: GO
TO 490
440 IF INKEY$="7" THEN LET pos
y=pos y+1: LET movimiento=0: GO
TO 490
450 IF INKEY$="8" THEN LET pos
x=pos x+1: LET movimiento=1: GO
TO 490
460 IF INKEY$="B" THEN BEEP 0.0
5,20: CLS: GO TO 260
470 IF INKEY$="F" THEN BEEP 0.5
,20: STOP
480 GO TO 420
490 GO SUB 700
500 IF borde=1 THEN GO TO 420
510 GO SUB 1000: GO TO 410
700 REM *****
*          VERIFICACION  *
*****

710 IF pos x<1 THEN LET pos x=p
os x+1: BEEP 0.05,20: GO TO 760
720 IF pos x>254 THEN LET pos x
=pos x-1: BEEP 0.05,20: GO TO 76
0
730 IF pos y<1 THEN LET pos y=p
os y+1: BEEP 0.05,20: GO TO 760
740 IF pos y>151 THEN LET pos y
=pos y-1: BEEP 0.05,20: GO TO 76
0
750 LET borde=0: RETURN
760 LET borde=1: RETURN
1000 REM *****
*          VISUALIZACION *
*****

1010 IF movimiento=1 THEN PRINT
AT 1,4;" " : PRINT AT 1,4,pos x
: RETURN
1020 PRINT AT 1,12;" " : PRINT
AT 1,12,pos y: RETURN
1030 RETURN

```

Programa "GDU"

Con este programa se generan 21 gráficos definidos por usuario (GDU), de una forma aleatoria. Al lado de cada gráfico aparece la letra a la que queda asignado.

Una vez ejecutado pulse cualquier tecla para terminar y aparecerá el mensaje:

⌀ OK, 15⌀ : 1

Pase a modo **G** y pulsando una tecla de la «A» a la «U», aparecerá en pantalla el gráfico correspondiente.

Programa "DIBUJANDO"

Este programa permite realizar cualquier tipo de dibujo

en la pantalla, utilizando los cursores de movimiento situados en las teclas «5», «6», «7» y «8». Al ser ejecutado, lo primero que debemos hacer es pulsar una de las cuatro teclas que indican el color de la «tiza» (V = verde, C = cyan, A = amarillo, B = blanco).

En la parte superior izquierda de la pantalla aparece en

PROGRAMA 5

```

10 REM *****
*          *
*   CURSO   *
*          *
* BASIC/SINCLAIR *
*          *
*****
*   "BIPBIP" *
*          *
*****

15 BORDER 2: PAPER 2: INK 6: C
LS
20 DRAW 255,0: DRAW 0,175: DRA
W -255,0: DRAW 0,-175
25 PRINT AT 5,7;"PROGRAMA ""BI
PBIP""
30 PRINT FLASH 1;AT 12,9;"PARE
LA CINTA"
35 PLOT 0,23: DRAW 255,0
40 PRINT AT 20,1;" "
42 RESTORE
45 FOR X=1 TO 28
50 READ logo
55 PRINT CHR$ logo;: BEEP 0.05
,logo/2
60 NEXT X: GO SUB 90
65 DATA 127,32,77,73,67,82,79,
72,79,66,66,89,32,38,32,82,65,70
,65,69,76,32,80,82,65,66,69,83
" 70 PRINT AT 12,9;"
72 GO TO 100
84 REM
*****
*          *
* TEMPORIZACION *
*          *
*****

90 FOR X=1 TO 300: NEXT X: RET
URN
100 GO SUB 120: GO TO 400

120 REM
*****
*          *
* TECLADO *
*          *
*****

140 FOR Y=12 TO 14
150 FOR X=12 TO 18
160 PRINT AT Y,X; PAPER 7;" "
170 NEXT X
180 NEXT Y
185 INK 0
190 PLOT 95,79
200 DRAW 56,0: DRAW 0,-23: DRAW
-56,0: DRAW 0,23
210 FOR X=103 TO 143 STEP 8
220 PLOT X,79: DRAW 0,-22
230 NEXT X
240 LET Y=79
250 LET X=100: GO SUB 350
260 LET X=108: GO SUB 350

270 LET X=124: GO SUB 350
280 LET X=132: GO SUB 350
290 LET X=140: GO SUB 350
340 RETURN
350 FOR X=X TO X+6
360 PLOT X,Y: DRAW 0,-15
370 NEXT X
380 RETURN
400 REM
*****
*          *
* MUSICA *
*          *
*****

430 LET clave de sol=12
440 LET duracion=0.2
450 FOR x=1 TO 22
460 READ nota,pausa
480 IF nota=0 THEN LET posicion
=12
490 IF nota=2 THEN LET posicion
=13
500 IF nota=4 THEN LET posicion
=14
510 IF nota=5 THEN LET posicion
=15
520 IF nota=7 THEN LET posicion
=16
530 PRINT AT 15,posicion;"↑"
532 BEEP duracion,nota+clave de
sol
534 PRINT AT 15,12;" "
540 PAUSE pausa
550 NEXT X
560 DATA 4,2,4,2,4,10,4,2,4,2,4
,10,4,2,7,2,0,2,2,2,4,10,5,2,5,2
,5,10,4,2,4,2,4,10,7,2,5,2,4,2,2
,2,0,10
600 PRINT #0;AT 1,1;"Pulse una
tecla para continuar"
610 PAUSE 0
615 REM
*****
*          *
* TECLADO MUSICAL *
*          *
*****

620 BORDER 1: PAPER 1: INK 6: C
LS
630 PRINT " A partir de este
momento su Spectrum se ha conve
rtido en un instrumento musical.
Obtendra una nota
diferente cada vez que pulse u
na tecla. Simultaneamente pued
e utilizar la tecla ""CAPS SHIF
T"" o ""SYMBOL SHIFT""
640 PRINT #0;"Pulse ""CAPS SHIF
T"" + ""SPACE"" pa ra terminar."
650 GO SUB 120
670 LET A$=INKEY$
680 IF A$="" THEN GO TO 670
700 LET nota=CODE A$/4
710 BEEP duracion,nota
720 GO TO 670

```

CLASIFICACION DE LAS SENTENCIAS SEGUN SU FUNCION

| | | |
|---|---|--|
| <p>COMANDOS DE CONTROL</p> <p>RUN CONT LIST LOAD SAVE VERIFY MERGE CLEAR CLS NEW</p> | <p>MANEJO DE CADENAS</p> <p>VAL VALS LEN STRS CHRS CODE</p> | <p>COMANDOS DE CONTROL DE COLOR E IMPRESION</p> <p>BORDER PAPER INK BRIGHT INVERSE FLASH OVER</p> |
| <p>COMANDOS DE PROGRAMACION</p> <p>REM LET DIM DEF EN PAUSE STOP FOR-TO-STEP NEXT IF-THEN GO TO GO SUB RETURN RESTORE RAND USR</p> | <p>FUNCIONES ARITMETICAS</p> <p>SGN ABS INT LN EXP SQR FN RND SIN COS TAN ASN ACS ATN PI BIN</p> | <p>AUXILIARES</p> <p>LINE TAB AT POINT ATTR SCREEN\$</p> |
| <p>COMANDOS DE ENTRADA/SALIDA</p> <p>PRINT INPUT INKEY\$ READ DATA PEEK POKE IN OUT</p> | <p>FUNCIONES LOGICAS</p> <p>AND OR NOT</p> | <p>SONIDO</p> <p>BEEP</p> |
| | <p>COMANDOS DE DIBUJO</p> <p>PLOT DRAW CIRCLE</p> | <p>MANEJO IMPRESORA</p> <p>LLIST LPRINT COPY</p> |
| | | <p>MANEJO PERIFERICOS (INTERFACE-1)</p> <p>FORMAT CAT ERASE OPEN CLOSE MOVE</p> |

tiempo real las coordenadas de la «tiza». La opción «B» borra toda la pantalla y posiciona la «tiza» en las coordenadas iniciales $X = 127$ e $Y = 80$. La opción «F» permite salir del programa, apareciendo el mensaje:

9 STOP statement, 470 : 3

Para que se autoejecute, grábelo de la forma:

SAVE "DIBUJANDO" LINE 10



Programa "BIPBIP"

El último programa presentado convierte a su Spectrum en un instrumento musical.

Después de la presentación se genera una melodía conocida por todos. Las instrucciones de manejo se encuentran en el propio programa.

Sávelo de la siguiente manera:

SAVE "BIPBIP" LINE 10



COMANDOS BASICOS

REM

Acceso al teclado



MODO **K**

Tipo de sentencia

Comando de programación

Concepto

Si durante la ejecución de un programa el ordenador detecta una sentencia REM, automáticamente analiza la siguiente instrucción y no ejecuta la correspondiente a REM. ¿Para qué sirve entonces una sentencia que no se ejecuta?, simplemente para poder introducir líneas de comentario (REMARK en inglés) dentro de un programa.

Estos comentarios sirven, por ejemplo, para indicar al principio de un programa el título de éste, el nombre del programador, la fecha y la edición, este último dato es bastante importante ya que nos permite comparar a simple vista, cuál es la versión actualizada de un mismo programa.

Dentro del programa, los comentarios sirven para indicarnos las funciones que realizan las distintas rutinas de que se compone o para aclarar el sig-

nificado de alguna de las variables utilizadas.

La estructura de esta sentencia es la siguiente:

| SENTENCIA | ARGUMENTO |
|-----------|--------------------|
| REM | Cualquier carácter |

Ejemplos:

— Título de programas.

```

10 REM ***** MANHATTAN *****
20 REM
30 REM          @ Pablo Otero
40 REM          Agosto 84
50 REM
60 REM          Edición ... 5
70 REM
80 REM
90 REM *****
    
```

— Función de una rutina.

```
260 REM ** CALCULO VARIABLES **
```

— Significado de variables.

```

200 REM *****
201 REM          valx=Coordenada x
202 REM
203 REM          valy=Coordenada y
204 REM
205 REM          ns = Nombre
206 REM
207 REM *****
    
```

Las sentencias REM ayudan a que un programa tenga claridad y limpieza, ya que si al cabo de un tiempo debemos realizar una modificación, será más fácil realizarla en un programa que esté documentado con comentarios, que en otro que no lo esté.

Consideraciones

En la edición de sentencias REM es necesario tener en cuenta los siguientes puntos:

— Para localizar con facilidad las distintas rutinas de un programa conviene que es-

tas sentencias resalten sobre las demás. Se puede, por ejemplo, enmarcar los mensajes con asteriscos (*) u otro símbolo, también puede utilizarse la función de vídeo invertido (INV. VIDEO), ésta será explicada posteriormente.

— No es necesario encerrar el argumento entre comillas (" "), aunque esté formado por una cadena alfanumérica.

— Pueden ir en líneas independientes o al final de una cadena de sentencias.

Ejemplo:

```

200 PRINT " - MICROHOBBY - " : REM
*** Esto es un ejemplo ***
    
```

ADVERTENCIA

A continuación de este tipo de sentencias, no debe editarse otra instrucción en la misma línea. Como el símbolo utilizado como separador de instrucciones (;) también puede formar parte del argumento de una sentencia REM, la sentencia editada a continuación no sería ejecutada.

Ejemplo:

```

200 REM *** Esto es un ejemplo
*** PRINT " - MICROHOBBY - "
    
```

Si ejecutamos esta instrucción, resulta que la cadena «—MICROHOBBY—» no será visualizada, ya que el ordenador interpreta que forma par-

te de la sentencia REM y por lo tanto no se ejecuta.

- Un argumento largo puede ser incluido en una sola línea.

Ejemplo:

```

10 REM ***** MANHATTAN *****
    © Pablo Otero
    Agosto 84
    Edición ... 5
  
```

Las sentencias REM ocupan parte de la memoria del ordenador, aunque como no son procesadas tienen la ventaja de no retardar la ejecución de un programa.

Funciones de Video



La función que el Spectrum presenta por defecto es la de *video normal* (TRUE VIDEO), es decir, que los caracteres se visualizan en el color de la tinta (INK) y el fondo en el del papel (PAPER). Si desea que estos dos colores se intercambien para destacar un fragmento de programa o algún texto, como, por ejemplo, el argumento de una sentencia REM, es necesario utilizar la función de *video invertido* (INV. VIDEO).

A esta función se accede,

en la edición de instrucciones, después de haber introducido el número de línea, pulsando la tecla CAPS SHIFT simultáneamente con la tecla correspondiente al número 4. A partir de este momento los caracteres tendrán el color del papel y el fondo el color de la tinta.

Para retornar a la visualización en video normal es necesario pulsar CAPS SHIFT y la tecla número 3, simultáneamente.

ADVERTENCIA

Para que este retorno tenga validez es necesario realizarlo dentro de una instrucción, es decir, después de haber introducido el número de línea y antes de pulsar ENTER, bien en la instrucción en la que se insertó la función invertida de video, bien en otra posterior.

Un ejemplo del manejo de estas funciones es el siguiente:

```

10 REM ***** MANHATTAN *****
    © PABLO OTERO
    AGOSTO 84
20 REM
30 REM ***** EDICIÓN ... 5 *****
  
```

LET

Acceso al teclado



Tipo de sentencia

Comando de programación.

Definición

También es conocido como comando de *asignación* ya que a una variable (numérica o de cadena) le asigna un valor; este puede ser una constante o variable (numérica o de cadena), el resultado de una expresión matemática, una operación de cadena o una función VAL, esta última será vista en otro capítulo.

Deben tenerse en cuenta las siguientes consideraciones:

- Una variable o constante numérica y una expresión matemática, sólo pueden ser asignadas a una variable numérica.
- Una variable o constante alfanumérica, una operación de cadena (concatenación, fragmentación) o una función VAL pueden asignarse solamente a las funciones de cadena.

Las estructuras de esta sentencia son:

| a) | SENTENCIA | ARGUMENTO |
|----|-----------|----------------------|
| | LET | Variable = expresión |

Ejemplos:

```

10 LET dato=123
20 LET a=(12+5)*7
30 LET resultado=dato+a
40 LET fin=resultado*7
  
```

| b) | SENTENCIA | ARGUMENTO |
|----|-----------|-----------------------------------|
| | LET | Variable \$ = expresión de cadena |

Ejemplos:

```

10 LET a$="MICROHOBBY"
20 LET b$="SEMANAL"
30 LET c$=a$b$
40 LET s$="Curso BASIC"
50 LET a$(2)="U"
  
```

Para utilizar una variable dentro de una expresión, debe estar asignada previamente por una sentencia LET, es decir, si las variables «dato» y «a» no hubieran sido asignadas no se podría haber utilizado la sentencia:

```
3Ø LET resultado = dato + a
```

ya que hubiera dado el siguiente error:

```
2 Variable not found
```

lo mismo ocurre con la sentencia:

```
3Ø LET c$ = a$ + b$
```

Veamos, por pasos, otro ejemplo de utilización de la sentencia LET. Supongamos que se desea implementar la fórmula que calcula el área de un círculo conociendo su diámetro ($d = 1Ø$).

$$S = \pi r^2, \text{ donde } r = \frac{d}{2}$$

1.º Asignar a la variable «diámetro» el valor $1Ø$.

```
LET diametro = 1Ø
```

2.º Calcular el radio ($r \times d/2$).

```
LET radio = diametro/2
```

3.º Asignación del valor «pi» (π). Podríamos hacer:

```
LET pi = 3.141592
```

pero como el Spectrum dispone de esta constante numérica, no es necesario asignarla. PI se encuentra en la tecla «M».

4.º Cálculo del área.

```
LET superficie = pi * radio ^ 2
```

5.º El programa completo sería:

```
1Ø LET diametro=1Ø
2Ø LET radio=diámetro/2
3Ø LET superficie=PI*radio^2
```

Después de ejecutar este programa la variable «superficie» tendrá almacenado el valor del área de un círculo de diámetro $1Ø$; para visualizar este valor incluya la siguiente instrucción y ejecútelo de nuevo.

```
4Ø PRINT superficie
```

Modificando el valor de la línea $1Ø$, obtendrá diversos valores para la variable «superficie».

Representación gráfica

La visualización de caracteres no ocupa la totalidad de la pantalla del televisor o monitor, ésta sólo se realiza en la zona central, dejando un espacio o borde alrededor (BORDER). La zona de representación permite visualizar 24 líneas de 32 caracteres, las líneas o filas están numeradas de la $Ø$ a la 23 y las columnas de la $Ø$ a la 31, esta disposición permite representar 768 caracteres y se denomina de *baja resolución*, ya que no permite hacer gráficos con calidad aceptable de definición.

Las líneas $Ø$ a 21 son las utilizadas por el usuario y la 22 y 23 las utiliza el ordenador para enviarnos los mensajes y para la introducción de comandos directos y edición de programas; estas líneas también pueden ser utilizadas por el usuario accediendo de un modo especial.

Cada carácter de la zona de usuario puede dividirse en una matriz de 8 por 8 puntos, los cuales se denominan *pixel* (picture element o elementos de imagen), con esta nueva división se obtiene una retícula de 176 por 256 puntos, lo que nos da un total de 45.056 pixel,

esta modalidad se denomina *alta resolución* y permite hacer gráficos con una calidad aceptable de definición.

```
PRINT
```

Acceso al teclado



Tipo de sentencia

Comando de salida.

Definición

Este comando permite visualizar en la pantalla el valor de las constantes, variables, expresiones o textos indicados en el argumento.

Veamos los distintos tipos de estructura que adopta esta sentencia.

a) Visualizar una constante numérica.

| SENTENCIA | ARGUMENTO |
|-----------|-----------|
| PRINT | constante |

Ejemplos:

```
1Ø PRINT 2Ø
2Ø PRINT 2.2372
3Ø PRINT 1Ø
4Ø PRINT 0.89
```

b) Visualizar una constante alfanumérica.

| SENTENCIA | ARGUMENTO |
|-----------|-----------|
| PRINT | "Cadena" |

Ejemplos:

```
10 PRINT "Programa ""SENIOR""""
20 PRINT "ARCHIVO DE DATOS"
30 PRINT "Menu de opciones"
40 PRINT "Pulse una tecla para
continuar"
```

- c) Visualizar variables numéricas previamente asignadas.

| SENTENCIA | ARGUMENTO |
|-----------|-----------|
| PRINT | Variable |

Ejemplos:

```
10 LET numero=20+3
20 PRINT numero
30 LET grados=(27 AND 1)+2
40 PRINT grados
50 LET valor=7
60 LET total=(valor=5)+10
70 PRINT total
```

- d) Visualizar variables de cadena asignadas previamente.

| SENTENCIA | ARGUMENTO |
|-----------|-------------|
| PRINT | variable \$ |

Ejemplos:

```
10 LET a$="123.47 pts."
20 PRINT a$
30 LET l$="total"
40 PRINT l$
50 LET g$="*****"
60 PRINT g$
```

- e) Visualizar expresiones aritméticas.

| SENTENCIA | ARGUMENTO |
|-----------|-----------|
| PRINT | expresión |

Ejemplos:

```
10 LET a=20
20 LET b=7+3
30 PRINT a/10+b
```

Con esta sentencia y utilizándola como comando directo, es decir, sin atribuirle un número de línea se puede manejar el Spectrum como si fuera una calculadora.

72 MICROBASIC

Ejemplos:

```
PRINT (7+5)+8
PRINT 2+2+5
PRINT (100-10*5)+2+8)/2
```

- f) Visualizar operaciones con cadenas.

| SENTENCIA | ARGUMENTO |
|-----------|--------------|
| PRINT | expresión \$ |

Ejemplos:

```
10 LET a$="Almacenamiento"
20 LET b$="memoria"
30 PRINT a$+b$
40 PRINT a$(2 TO 6)
50 PRINT b$(5)
```

Desplazamientos

Cuando se ejecuta una sentencia del tipo PRINT, el argumento se visualiza al principio de cada línea, en el momento que se completan las 22 líneas y antes de visualizar un nuevo valor, el ordenador nos presenta en la parte inferior de la pantalla, el mensaje:

scroll?

con esta pregunta el ordenador queda a la espera de realizar un «scroll» o desplazamiento del texto hacia la parte superior o no. Pulsando cualquiera de las teclas «N», «BREAK» (SPACE) o «STOP» (A) la ejecución del programa se detiene y nos presenta el mensaje:

D BREAK - CONT repeats

si se pulsa cualquier otra tecla, el desplazamiento o *scrolling* se realiza.

El siguiente miniprograma visualiza cien veces, utilizando esta particularidad, la cadena «MICROHOBBY».

```
10 FOR n=1 TO 100
20 PRINT n,"MICROHOBBY"
30 NEXT n
```

Formatos

Por razones de estética o para una presentación de datos más ordenada, se pueden utilizar diversos formatos de visualización.

- a) Visualizar una línea en blanco.

SENTENCIA

PRINT

Ejemplo:

```
10 PRINT "Programas"
20 PRINT "PRINT" PRINT
30 PRINT "Educativos"
```

La instrucción 20 deja tres espacios en blanco entre el mensaje de la línea 10 y el de la 30.

- b) Para visualizar variables o constantes seguidas dentro de una misma línea se utiliza el signo ortográfico del *punto y coma* (;).

Ejemplos:

```
10 LET a$="La suma de "
20 LET total=7+5
30 PRINT a$;7;"+";5;" es ";tot
```

La instrucción 30 visualiza en pantalla el mensaje:

La suma de 7 + 5 es 12

```
10 LET base=30
20 LET altura=20
30 LET area=base*altura
40 PRINT "AREA DE UN RECTANGULO"
50 PRINT "
"
60 PRINT "
"
70 PRINT "BASE: ";base;" y ALTURA: ";altura
80 PRINT "
"
90 PRINT "AREA: ";area
```

c) Utilizando el signo ortográfico de la *coma* (,) se consigue una tabulación, de forma automática, de los elementos del argumento.

Con este formato, la pantalla queda dividida, verticalmente en dos zonas o campos; los elementos se van visualizando alternativamente en las columnas ϕ y 16.

Cuando la longitud del elemento a visualizar en el primer campo es superior a 15 caracteres, el siguiente se visualiza al comienzo de la línea inferior.

Ejemplo:

```
10 LET a$="FRUTAS"
20 LET b$="PIE."
30 PRINT a$,b$
40 PRINT " "
50 PRINT "Manzanas",.60
60 PRINT "Peras",.90
70 PRINT "Platanos",.115
80 PRINT "Naranjas",.70
90 PRINT "Uvas",.150
100 PRINT
```

d) En lugar de utilizar la sentencia PRINT, sin argumento, para dejar líneas en blanco, también puede utilizarse el signo ortográfico del *apóstrofe* ('), situado en la tecla con el número 7.

El número de espacios en blanco es igual al número de apóstrofes menos uno, ya que el primero sirve para retornar a la línea siguiente.

Ejemplo:

```
10 PRINT "Manzanas"'"Peras"'"
Platanos"'"Naranjas"'"Uvas"
```

TAB

Acceso al teclado

TAB



MODO **E**



Tipo de sentencia

Auxiliar.

Concepto

Esta *palabra clave* se utiliza conjuntamente con PRINT y tabula la salida de datos al valor deseado. Su estructura es la siguiente:

PRINT TAB expresión; valor

una vez evaluada la *expresión*, nos indica ésta el número de columna a partir del cual debe visualizarse el *valor* (numérico o de cadena).

Ejemplos:

```
10 PRINT TAB 10;"Directorio"
20 PRINT TAB 7;.125
30 LET a=10: LET b=100*a
40 PRINT TAB 30;a,b
50 LET a$="casino"
60 PRINT TAB 20;a$
```

Cuando el resultado de la expresión es un número negativo, el ordenador presenta el mensaje:

B Integer out of range

Los números decimales son redondeados de manera que TAB 7.5 es igual que TAB 8 y TAB 7.4 es equivalente a TAB 7.

Podríamos suponer en un principio que los posibles valores de la expresión tendrían que estar comprendidos entre ϕ y 31, ya que éstos son los números de columna existentes, pues bien, esto no es así, ya que podemos introducir cualquier número comprendido entre ϕ y 65535. ¿Cómo tabula el ordenador una sentencia del tipo TAB 75 ϕ ? la respuesta es: reduciendo a «módulo 32», es decir, divide la expresión entre 32 sin obtener decimales y el *resto* de la división lo interpreta como número de columna.

Observe cómo las siguientes instrucciones, que simulan la forma en que el ordenador calcula el número de columna, realizan la misma función que

PRINT TAB 75 ϕ ; «MICROHOBBY»

O

PRINT TAB 14; «MICROHOBBY»

```
10 REM *** Modulo 32 ***
20 REM
30 LET numero=750
40 LET cociente=INT (numero/32)
50 LET resto=numero-cociente*32
60 PRINT TAB resto;"MICROHOBBY"
```

esta sentencia auxiliar puede combinarse con los signos ortográficos (", " y "; ") que también determinan el formato de salida de datos.

Ejemplos:

```
10 PRINT TAB 2;"a"
20 PRINT TAB 8;"b"
30 PRINT TAB 14;"c"
40 PRINT TAB 20;"d"
50 PRINT TAB 26;"e"
```

sustituya el *punto y coma* del final por una *coma* y observe el nuevo resultado. Las cinco sentencias anteriores podrían editarse en una sola:

```
10 PRINT TAB 2;"a";TAB 8;"b";TAB 14;"c";TAB 20;"d";TAB 26;"e";AT
```

AT

Acceso al teclado

CODE



IN



Tipo de sentencia

Auxiliar.

Concepto

Esta sentencia visualiza a partir del número de línea y columna especificado, un valor. Su estructura es la siguiente:

PRINT AT línea, columna; valor

Ejemplos:

```
10 PRINT AT 4,11;"Version"
20 PRINT AT 6,10;"Microdrive"
30 PRINT AT 10,3;"Programas".A
T
10,18;"Longitud"
40 LET linea=11
50 LET col1=3 LET col2=10
60 LET a$=""
70 PRINT AT linea,col1;a$+"_"
AT linea,col2;a$
```

Cuando el valor de *línea* es superior a 22 ó 31 en el caso de *columna*, el ordenador presenta el mensaje:

B Integer out of range

En cambio, cuando se quiere imprimir en la primera línea de la zona destinada a los mensajes (22), presenta otro distinto:

5 Out of screen

Los números negativos los interpreta como si fueran positivos, por tanto dará igual editar:

10 PRINT AT 10, 10; «Hola»

o

10 PRINT AT -10, -10; «Hola»

aunque la primera forma es la más correcta.

PROGRAMA 1

```
10 REM *****
* CURSO BASIC *
* *****
* "GRANJA" *
* *****
20 BORDER 1: PAPER 1: INK 7: C
L5
30 REM *****
* INTRODUCCION *
* *****
40 PRINT AT 3,3;"El programa A
NIMAL calcula el""numero total
de animales que con""viven en
una granja, asi como la""suma
de sus patas."
50 PRINT AT 12,3;"Introduzca l
os datos a medida""que el orde
nador se los vaya pi-""diendo."
60 PRINT #0;AT 0,2;"Pulse una
tecla para seguir"
70 PAUSE 0
80 CLS
90 REM *****
* ENTRADA DE DATOS *
* *****
100 INPUT "Numero de patos? ";p
ato
110 INPUT "Numero de gallinas?
";gallina
120 INPUT "Numero de conejos? "
;conejo
130 INPUT "Numero de palomas? "
;paloma
140 INPUT "Numero de cerdos? ";
cerdo
150 REM *****
* PRINT DATOS *
* *****
160 PRINT AT 2,0;"Patos ..... "
;pato
170 PRINT AT 4,0;"Gallinas .. "
;gallina
180 PRINT AT 6,0;"Conejos ... "
;conejo
190 PRINT AT 8,0;"Palomas ... "
;paloma
200 PRINT AT 10,0;"Cerdos .....
";cerdo
210 REM *****
* CALCULO TOTAL *
* *****
```

```

220 LET total=pato+gallina+cone
jo+paloma+cerdo
230 PRINT AT 15,6;"Total .....
";total
240 REM *****
* CALCULO PATAS *
*****
250 LET pat=pato*2
260 LET ga=gallina*2
270 LET co=conejo*4
280 LET pal=paloma*2
290 LET ce=cerdo*4
300 LET patas=pat+ga+co+pal+ce
400 PRINT AT 17,6;"Patás .....
";patas

```

PROGRAMA 2

```

10 REM *****
* CURSO BASIC *
*****
* ECUAC. 2 Gr *
*****
20 BORDER 1: PAPER 1: INK 7: C
LS
30 REM *****
* INTRODUCCION *
*****
40 PRINT " Este programa cal
cula las ra-ices de una ecuacion
de segundo grado."
50 PRINT "" La formula gene
ral de este tipo de ecuaciones e
s:
60 PRINT "" ax2 + bx +
c = 0"
70 PRINT "" De donde se ded
uce la incognita:
80 PRINT ""
-----
90 PRINT " -b +- V b2
- 4ac"
100 PRINT " x = -----
-----
110 PRINT "" 2a
"
120 PRINT #0;" Pulse una tecl
a para seguir"
130 PAUSE 0
140 CLS
150 PRINT " El programa le ir
a pidiendo""el valor de ""a"",
""b"" y ""c"""
160 PRINT "" Posteriormente
realizara el""calculo y seran
visualizadas las""dos raices.
"
170 PRINT #0;" Pulse una tecla
para comenar"

```

Canales de Comunicación

El Spectrum dispone de una serie de canales de comunicación o «streams», por los que el ordenador mantiene el intercambio de información con sus periféricos. Estos canales están numerados del «0» al «15» y precedidos por el signo del sostenido (#).

Pueden ser de *entrada* (INPUT) si los datos llegan al ordenador procedentes de alguno de sus periféricos, de *salida* (OUTPUT), si el ordenador es el que los envía y de *entrada/salida* (I/O), si el canal sirve tanto para enviar como para recibir datos.

Los canales «0» al «3» tienen una asignación fija de periféricos y el resto puede ser utilizado por cualquiera de ellos, previa definición por el usuario.

A través del canal # 2 se pueden visualizar valores numéricos o cadenas en la zona de pantalla destinada al usuario, por tanto es indiferente utilizar el comando «PRINT # 2» o «PRINT», ya que este último lleva asignado, implícitamente el canal de comunicación # 2.

El canal # es el asignado al periférico conocido como *impresora*, por tanto será de salida. Una instrucción del tipo:

```
PRINT # 3;"MICROHOBBY"
```

imprimirá la misma cadena que la sentencia específica de la impresora (LPRINT):

```
LPRINT "MICROHOBBY"
```

Los canales # 0 y # 1 son de entrada/salida y están relacionados con las dos últimas líneas de la pantalla, la 22 y 23. Estas líneas están destinadas para la introducción de sentencias en la edición de pro-

```

180 PAUSE 0
190 CLS
200 REM *****
* ENTRADA DE DATOS *
*
210 INPUT AT 0,0;"Valor de ""a""
";a;AT 1,0;"Valor de ""b""? ";
;b;AT 2,0;"Valor de ""c""? ";c
220 REM *****
* VISUALIZACION *
*
230 PRINT ""a"" = ";a
240 PRINT ""b"" = ";b
250 PRINT ""c"" = ";c
260 REM *****
* CALCULO DE RAICES *
*
270 LET raiz=SQR (b*b-(4*a*c))
280 LET divisor=2*a
290 LET raiz1=(-b+raiz)/divisor
300 LET raiz2=(-b-raiz)/divisor
400 REM *****
* VISUALIZACION DE *
*
410 PRINT ""Primera raiz ....
";raiz1
420 PRINT ""Segunda raiz ....
";raiz2

```

gramas, a la de datos cuando se utiliza la sentencia «INPUT» y para la visualización de los informes del ordenador.

Como ya vimos, con la sentencia PRINT no se podía visualizar ningún texto en estas dos líneas, sin embargo utilizando cualquiera de estos dos canales (0 ó 1) esto es posible. Ejecute las siguientes instrucciones y lo comprobará:

```

10 FOR a=0 TO 31
20 PRINT AT 0,0;"LINEA ";a
30 NEXT a
40 PRINT #0;AT 0,0;"LINEA 32"
50 PRINT #0;AT 1,0;"LINEA 33"
60 PAUSE 0

```

Al pulsar cualquier tecla, observará que las líneas 22 y

23 desaparecen, ¿por qué ocurre esto? Como hemos dicho anteriormente, estas líneas también las utiliza el ordenador para enviarnos sus mensajes, por tanto al visualizar el informe de fin de programa:

OK, 60:1

estas desaparecen.

Para aprovechar al máximo la capacidad de la zona de visualización destinada al usuario, los mensajes que tengan que enviarse en un programa, pueden hacerse a través de estos canales.

Ejemplos:

```

10 PRINT #0;"Pulse una tecla p
ara continuar"
20 PAUSE 0
30 BEEP 0,05,20

```

```

10 PRINT #1;"Para grabar pulse
una tecla"
20 PAUSE 0
30 BEEP 0,05,20

```

La instrucción «PAUSE 0» detiene la ejecución de un programa hasta que se pulsa una tecla, y la sentencia «BEEP 0,05,20» hace que suene el altavoz interno del Spectrum durante 0,05 sg. y con un tono de valor 20.

La impresión a través de los canales #0 y #1 puede combinarse con las sentencias auxiliares TAB y AT, teniendo en cuenta que para esta última, la línea 22 se convierte en la 0 y la 23 en la 1.

Ejemplos:

```

10 PRINT #1,TAB 14;"hola"
20 PAUSE 0
30 BEEP 0,5,5

```

```

10 PRINT #1;AT 0,6;"Conecte la
impresora"
20 PRINT #1;AT 1,2;"y pulse un
a tecla para seguir"
30 PAUSE 0
40 BEEP 0,2,30

```

Cuando se utiliza un canal que no está activado, aparece el siguiente mensaje:

O invalid stream

Al especificar una operación de entrada en un canal destinado a salida el informe de error es:

J Invalid I/O device

INPUT**Acceso al teclado****CODE**MODO **IN****Tipo de sentencia**

Comando de entrada.

Concepto

Esta sentencia permite introducir por teclado, durante la ejecución de un programa, datos, tanto numéricos como de cadena y se asignan a la variable indicadas en el argumento. Los datos introducidos se visualizan en las dos líneas inferiores de la pantalla, si cometemos algún error, podemos corregirlo con la función «DELETE». Para que los datos sean aceptados por el ordenador y continúe, por tanto, la ejecución del programa, debe pulsarse la tecla «ENTER».

Las estructuras básicas de esta sentencia son:

a)

| SENTENCIA | ARGUMENTO |
|-----------|---------------|
| INPUT | Var. numérica |

Ejemplos:

```

13 INPUT radio
14 PRINT radio
15 INPUT numero
16 PRINT numero
17 INPUT i
18 PRINT i

```

Si se introduce un valor no numérico, el intérprete del

PROGRAMA 3

```

10 REM *****
*   CURSO BASIC   *
* *****
* INTERES SIMPLE *
* *****
11 BORDER 1: PAPER 1: INK 7: C
LS
12 REM *****
*   INTRODUCCION *
* *****

13 PRINT AT 2,3;"Este programa
calcula el inte"
14 PRINT AT 4,0;"res simple de
acuerdo con la for"
15 PRINT AT 6,0;"mula:"
16 PRINT AT 10,11;"C * R * T"
17 PRINT AT 11,7;"I = _____"

18 PRINT AT 13,14;100: PAUSE 3
00: CLS
19 REM *****
*   ENTRADA DE   *
*   DATOS       *
* *****

20 INPUT "CAPITAL: ";capital
30 PRINT "CAPITAL .... ";capit
al;" pesetas"
40 INPUT "REDITOS en %: ";redi
tos
50 PRINT "REDITOS .... ";redi
tos;" % anual"
60 INPUT "TIEMPO: ";tiempo
70 PRINT "TIEMPO ..... ";tiem
po;" (anualidad)""
80 REM *****
*   CALCULO     *
* *****

90 LET interes=capital*reditos
*tiempo/100
100 REM *****
*   RESULTADO   *
* *****

110 PRINT "INTERESES .. ";inter
es;" pesetas"

```

Spectrum entenderá que es el nombre de una variable numérica, si esta no existe como tal, aparecerá el mensaje de error:

2 Variable not found

b)

| SENTENCIA | ARGUMENTO |
|-----------|-------------|
| INPUT | Var. cadena |

Ejemplos:

```

10 INPUT a$
20 PRINT a$
30 INPUT b$
40 PRINT b$
50 INPUT c$
60 PRINT c$

```

Observará que cuando la variable es del tipo alfanumérico o de cadena aparecen automáticamente las comillas en la zona destinada a la introducción de datos.

Dentro de la sentencia «INPUT» pueden especificarse más de una variable, éstas deben ir separadas por cualquiera de los signos ortográficos («», «;» o «,») con el mismo significado que con la sentencia «PRINT».

Ejemplos:

```

10 INPUT a$, b$, c$
20 PRINT a$, b$, c$
30 INPUT j$, equipo$, puntuacion$
40 PRINT j$, equipo$, puntuacion$
50 INPUT programa$, precio$
60 PRINT programa$, precio$

```

Otra posibilidad de la sentencia «INPUT» es la de presentar en pantalla un mensaje informativo, indicando qué tipo de dato debe introducir el usuario, este sistema es más eficaz ya que aclara las posibles dudas al respecto. El mensaje, al ser alfanumérico, debe ir entrecorrido.

PROGRAMA 4

```

10 REM *****
*   CURSO BASIC   *
* *****        *
*   GRADOS       *
* *****        *
20 BORDER 1: PAPER 1: INK 7: C
LS
30 REM *****
*   INTRODUCCION *
* *****        *

40 PRINT "   El programa GRADO
5   convierte"" primeramente los
   grados Centigra"" dos en Fahren
   heit y posteriormen"" te realiz
   a la transformacion in-"" versa
"
50 PRINT ""
60 PRINT ""          F = 9"
32" PRINT ""          * C +
70 PRINT ""          5"
80 PRINT ""          (F - 32
) * 5"
90 PRINT ""          C = _____
100 PRINT ""
110 PRINT #0;" Pulse una tecla
para seguir"
120 PAUSE 0
130 CLS
140 REM *****
*   ENTRADA     *
* *****        *
*   CENTIGRADOS *
* *****        *

150 INPUT "Grados Centigrados?
"; C
160 PRINT "" C;" grados Centigr
ados: "
170 REM *****
*   CALCULO "F" *
* *****        *

180 LET fahrenheit=9*c/5+32
190 PRINT "" igual a = "; fahrenheit; " F"
200 REM *****
*   ENTRADA     *
* *****        *

```

```

210 INPUT "Grados Fahrenheit? "
; f
220 PRINT " f;" " grados Fahren
heit:"
230 REM
*****
*
* CALCULO "C" *
*
*****

240 LET centigrados=(f-32)*5/9
250 PRINT " igual a = ";centigr
ados;" C"

```

PROGRAMA 5

```

10 REM *****
* CURSO BASIC *
* *****
* FICHA *
* *****

20 BORDER 1: PAPER 1: INK 6: C
LS
30 REM *****
* DIBUJO FICHA *
* *****

50 LET a$=" "
60 LET b$=" "

70 PRINT " " + a$ + " "
80 FOR n=1 TO 20
90 PRINT AT n,0;" | ";AT n,31;" | "
100 NEXT n
110 PRINT " " + b$ + " "
120 REM *****
* ROTULOS FICHA *
* *****

130 PRINT AT 2,3;"NOMBRE:"
140 PRINT AT 5,11;"APELLIDOS"
150 PRINT AT 6,1;" "

160 PRINT AT 8,3;"1:";AT 8,16;"
2:"
170 PRINT AT 11,3;"CALLE:";AT 1
1,23;"N."
180 PRINT AT 13,3;"PISO :";AT 1
3,18;"PUERTA:"

```

Ejemplos:

```

10 INPUT "Nombre? " ; n$
20 INPUT "Primer apellido? " ; a
$
30 INPUT "Segundo apellido? " ;
b$
40 INPUT "Calle? " ; c$
50 INPUT "Numero? " ; n
60 INPUT "Telefono? " ; t
70 INPUT "Poblacion? " ; p$
80 PRINT n$ " " + a$ " " + b$ " "
90 PRINT c$ " " ; n " "
100 PRINT t " " ; p$

```

Cuando entre los comentarios del mensaje deba figurar el contenido de una variable, esta deberá ir encerrada entre paréntesis, bien ella sola, bien todo el mensaje, ya que de lo contrario, el intérprete BASIC la tomará como variable a introducir.

Ejemplos:

```

10 INPUT "Nombre? " ; n$
20 INPUT "No. de casa? " ; n1$
calle vives
30 INPUT "En que numero de la
calle? " ; n2$
40 PRINT "Te llamas " ; n$ " y v
ives en el numero " ; n1$ " de la ca
lle " ; c$

```

INPUT TAB y AT

Las sentencias auxiliares «TAB» y «AT» también pueden utilizarse conjuntamente con «INPUT». La palabra clave «TAB» se utiliza de forma similar que cuando acompaña a la sentencia «PRINT».

Ejemplos:

```

10 INPUT TAB 20;"Nombre: " ; b$
20 INPUT TAB 7;"Nombre: " ; b$
30 INPUT TAB 2;"Tabulación: " ;
a$
40 INPUT TAB 10;"Memoria en K
bytes: " ; w

```

«AT» tiene un tratamiento ligeramente distinto. Todos los «INPUT» a utilizar con «AT» deben estar incluidos en la misma instrucción, separados por «;». Independientemente de las coordenadas del primer «AT», la entrada de datos del primer «INPUT» se realiza en la zona inferior de la pantalla. Los siguientes se van a introducir en las coordenadas indicadas en los «AT», pero tomando como línea de referencia la especificada en el primer «AT».


```

190 PRINT AT 16,3;"TELEFONO : "
200 PRINT AT 19,3;"POBLACION:"
210 PRINT #0;" Rellene los camp
os de la ficha"
220 PAUSE 200
230 REM

```

```

*****
*
* ENTRADA DE DATOS *
*
*****

```

```

240 PRINT AT 2,11; FLASH 1;"<"
250 INPUT "? "; LINE n$
260 PRINT AT 2,11;n$
270 PRINT AT 8,6; FLASH 1;"<"
280 INPUT "? "; LINE a$
290 PRINT AT 8,6;a$
300 PRINT AT 8,19; FLASH 1;"<"
310 INPUT "? "; LINE b$
320 PRINT AT 8,19;b$
330 PRINT AT 11,10; FLASH 1;"<"
340 INPUT "? "; LINE c$
350 PRINT AT 11,10;c$
360 PRINT AT 11,26; FLASH 1;"<"
370 INPUT "? "; nu
380 PRINT AT 11,26; nu
390 PRINT AT 13,10; FLASH 1;"<"
400 INPUT "? "; pi
410 PRINT AT 13,10; pi
420 PRINT AT 13,26; FLASH 1;"<"
430 INPUT "? "; d$
440 PRINT AT 13,26; d$
450 PRINT AT 16,14; FLASH 1;"<"
460 INPUT "? "; te
470 PRINT AT 16,14; te
480 PRINT AT 19,14; FLASH 1;"<"
490 INPUT "? "; LINE p$
500 PRINT AT 19,14; p$
510 PRINT #0; AT 1,10;"Fin de ed
icion"
520 PAUSE 200

```

Otra aplicación

Existe una aplicación de la sentencia «INPUT» no especificada en el manual y que posiblemente sea consecuencia de los «efectos laterales» de la programación. Cuando la sentencia «INPUT» va acompañada de cualquiera de los siguientes argumentos:

```

INPUT
INPUT
INPUT
INPUT cadena alfanumerica
INPUT expresión numerica

```

se borran las dos últimas líneas de la pantalla, la 22 y la 23.

¿Qué utilidad puede tener esto? Cuando se utilizan los canales de comunicación #0 y #1 para enviar mensajes a veces es necesario incluir una instrucción que borre estas líneas para que no se mezclen los mensajes, esto ocurre cuando el segundo es más corto que el primero, como por ejemplo:

```

10 PRINT #0; AT 0,0;"Espere un
momento por favor"
20 PAUSE 100
30 PRINT #0; AT 0,0;"Verifique
la grabación"
40 PAUSE 0

```

Veamos algunos ejemplos:

```

10 INPUT AT 0,0;"Nombre? ";a$
AT 1,0;"Donde naciste? ";a$
11 b$ AT 3,0;"¿A que provincia pe
rtenece? ";b$; AT 6,0;"¿C
uando se encuentra en la Comunidad Aut
ónoma de? ";c$
20 PRINT b$;" se encuentra en
la provincia de ";c$;" y pertene
ce a la Comunidad Autónoma de ";
d$

```

```

10 INPUT AT 10,0;"Linea 10 "
a$; AT 2,0;"Linea 2 " ;a$; AT 4,0;"
Linea 4 " ;a$; AT 15,0;"Linea 15
";a$; AT 12,0;"Linea 12 " ;a$; AT
4,0;"Otra vez la línea 4 " ;a$; AT
21,0;"Linea 21 " ;a$

```

INPUT LINE

La sentencia «INPUT» también puede combinarse con la palabra clave «LINE». Debe uti-

lizarse únicamente con las variables de cadena y el resultado obtenido es el que el ordenador no visualiza las comillas durante la introducción de datos alfanuméricos.

La estructura es:

| SENTENCIA | ARGUMENTO |
|-----------|------------------|
| INPUT | LINE var. cadena |

Ejemplos:

```

10 INPUT LINE a$
20 PRINT a$
30 INPUT "Población? "; LINE b
$
40 PRINT b$
50 INPUT "Número de vueltas? "
LINE c$
60 PRINT c$

```

Para subsanar esta alteración del segundo mensaje, podríamos incluir la instrucción:

```
25 PRINT #0; AT 0,0;" "
```

o bien

```
25 PRINT #0; AT 0,22;" "
```

pero resulta más cómodo hacerlo de la forma:

```
25 INPUT 0
```

y además tiene la ventaja de que ocupa menos memoria.

Programas de repaso

Como colofón al estudio de las sentencias básicas de programación:

```
REM  
LET  
PRINT  
INPUT
```

se analizan en este capítulo cinco programas realizados con este tipo de instrucciones.

Estos programas, numerados del uno al cinco, son los siguientes:

1. GRANJA
2. ECUACION
3. INTERES
4. GRADOS
5. FICHA

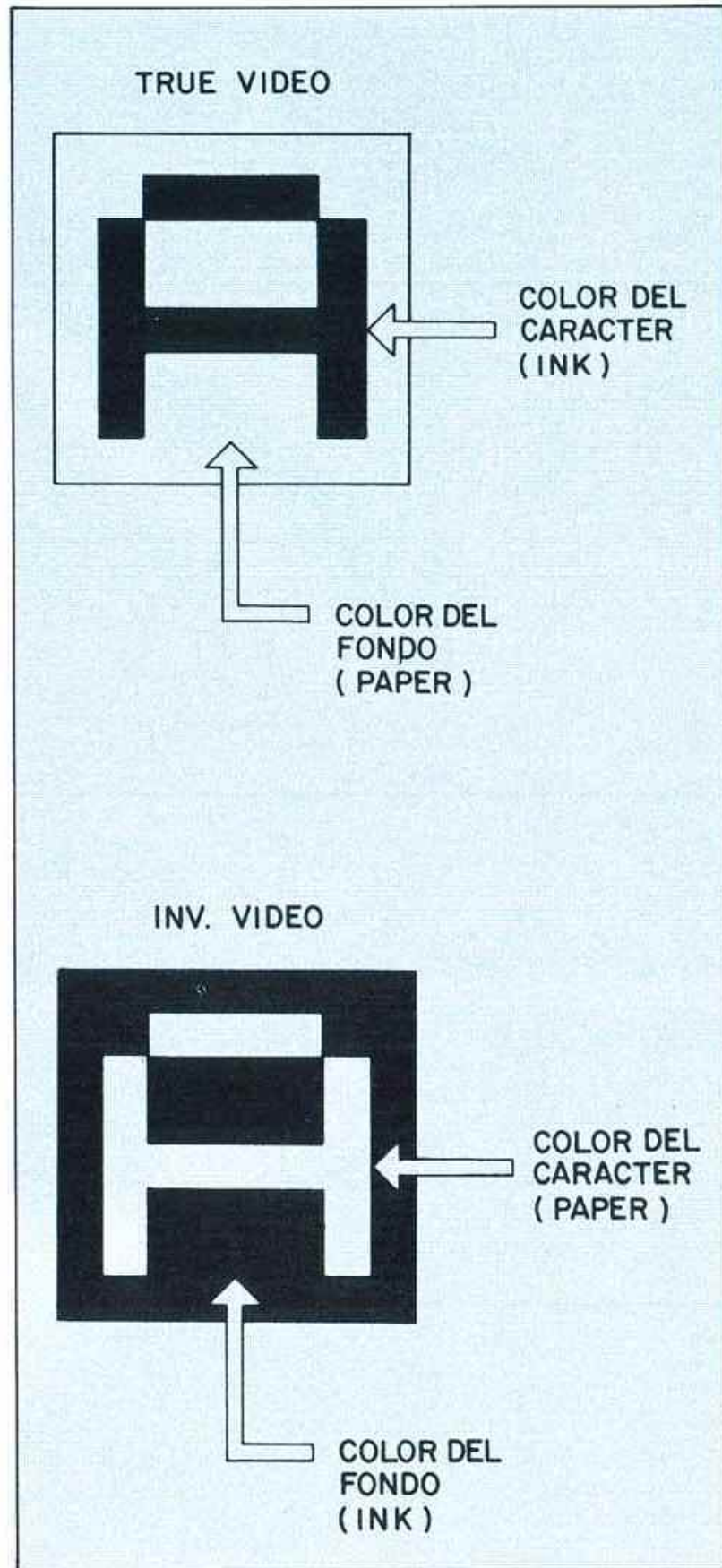
Programa «GRANJA»

Para almacenarlo en cinta una vez editado hágalo, por ejemplo, de la forma:

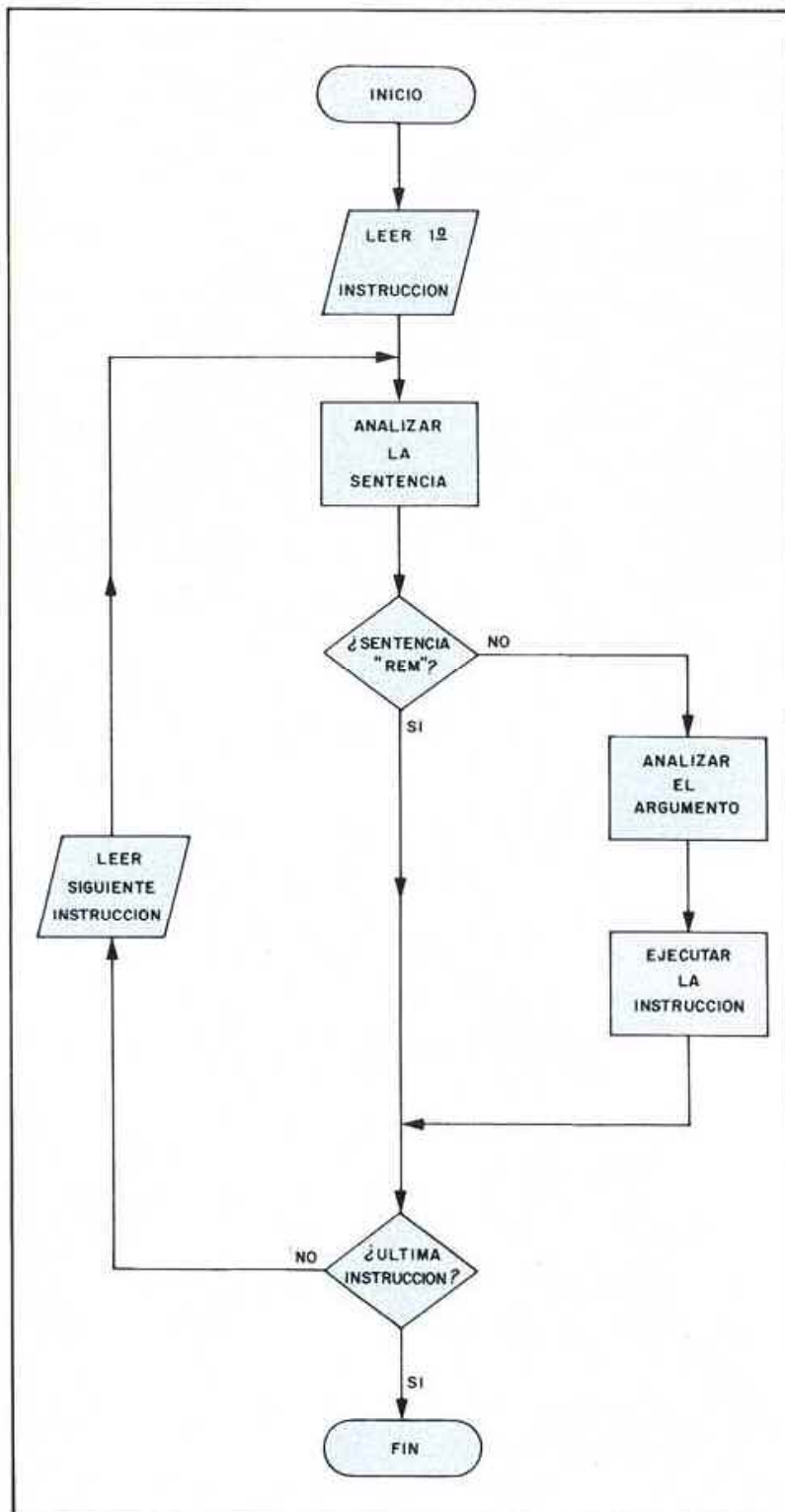
```
SAVE "granja"
```

Este programa es bastante sencillo, ya que calcula el número total de animales que hay en una granja, a partir de los datos que le son introducidos por teclado; también calcula el número total de patas.

La sentencia 1 ϕ es la presentación del programa. En la 2 ϕ hay cuatro sentencias, esto es posible ya que se utiliza el signo separador «;»; éstas no se han explicado todavía, pero vamos a ver unas pequeñas nociones sobre su funcionamiento. En conjunto realizan la tarea de colorear la pantalla. La sentencia «BORDER 1» asigna el color azul al borde de la pantalla, «PAPER 1» asigna el color azul al fondo, e «INK 7» tiene la misión de que el color de los caracteres sea



Funciones de video.



Análisis sentencia "REM".

blanco. Por último, «CLS» es una sentencia que borra la pantalla y asigna de inmediato los colores especificados en los comandos anteriores.

Las sentencias 4φ y 5φ son

del tipo «PRINT AT» y sirven para dar al usuario una pequeña información sobre la finalidad del programa.

La instrucción 6φ utiliza el canal de comunicación φ

para visualizar el mensaje de espera.

«PAUSE φ» situada en la línea 7φ es una instrucción que detiene la ejecución del programa hasta que se pulsa una tecla.

La sentencia 8φ borra la información visualizada en la pantalla.

De la línea 1φφ a la 14φ se encuentran los «INPUT» necesarios para la entrada de datos. La variable numérica utilizada para almacenar el número total de patos es «pato», para los restantes animales se han utilizado: «gallina», «conejo», «paloma» y «cerdo».

Después de la introducción de datos, se realiza la visualización detallada de éstos, los «PRINT AT» de las líneas 16φ a 2φφ, se encargan de ello.

El cálculo del número total de animales se realiza en la línea 22φ, se asigna a la variable «total» la suma de las variables «pato», «gallina», «conejo», «paloma» y «cerdo». La línea 23φ se encarga de visualizar este valor.

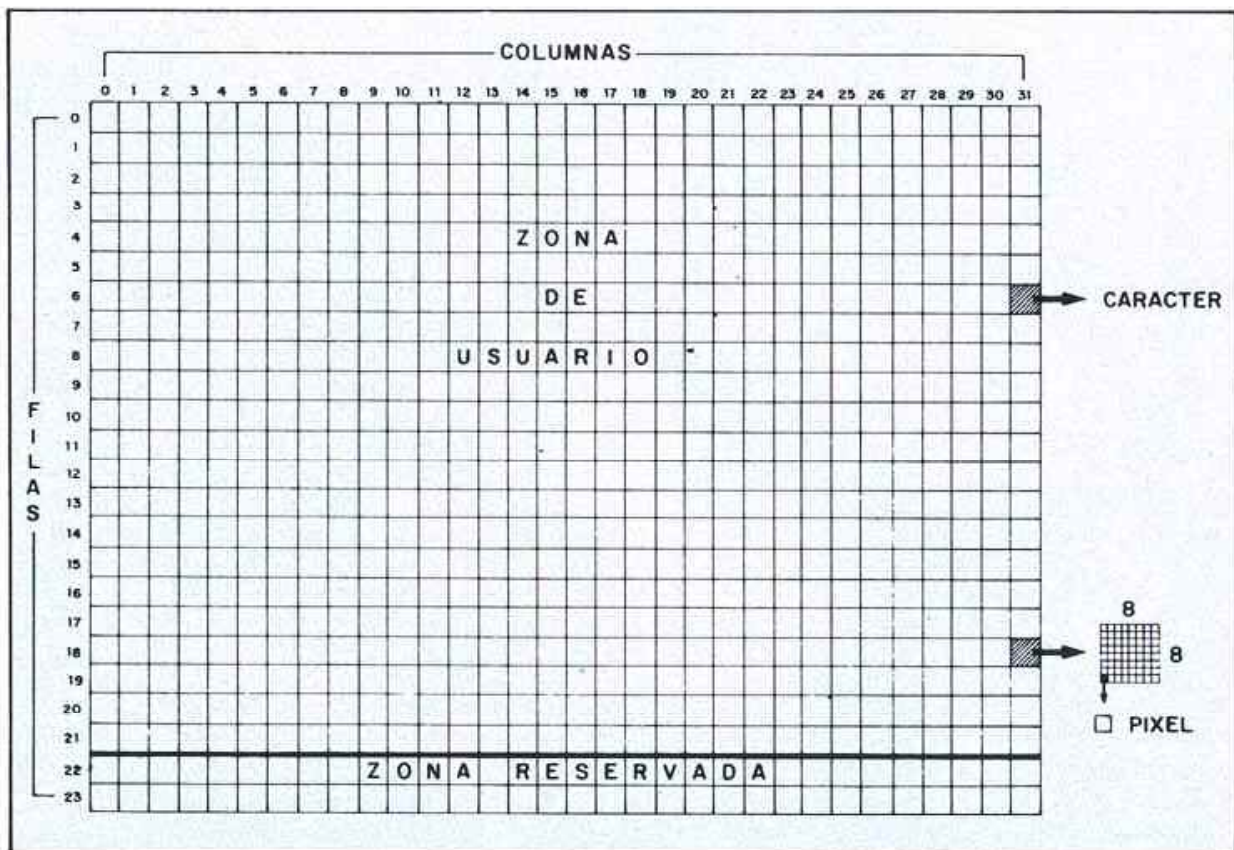
Las líneas 25φ a 3φφ se encargan de calcular el número total de patas. Primero se asignan a las variables «pat», «gat», «co», «pal» y «ce» los valores totales por especie.

Ejemplo: como los conejos tiene cuatro patas, será necesario multiplicar este número por el número total de conejos, valor especificado en la variable «conejo»:

LET co = conejo * 4

Posteriormente se asigna a la variable «patas» la suma de las variables «pat», «ga», «co», «pal» y «ce».

Por último el valor de la variable «patas» se visualiza con el «PRINT AT» de la línea 4φφ.



Zonas de visualización.

Programa «ECUACION»

Salvar el programa de la forma:

SAVE "ecuación"

Este programa calcula las dos raíces de una ecuación de segundo grado del tipo:

$$ax^2 + bx + c = 0$$

Los dos valores de «x» que cumplen esta ecuación se calculan con la fórmula:

$$x = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$$

Los valores que deben introducirse para que el programa calcule dichas ecuaciones son los correspondientes a las variables «a», «b» y «c».

Las funciones de las sentencias que componen dicho programa son las siguientes:

- 1 ϕ : Comentario con el nombre del programa.
- 2 ϕ : Asignación de los colores de borde, papel y tinta.
- 4 ϕ – 14 ϕ : Primera pantalla de información.
- 15 ϕ – 19 ϕ : Segunda pantalla de información.
- 21 ϕ : Introducción de los valores de las variables «a», «b» y «c».
- 23 ϕ – 25 ϕ : Visualización detallada de las variables.
- 27 ϕ – 30 ϕ : Cálculo de las dos raíces. En este programa se ha utilizado la sentencia «SQR» que calcula la raíz cua-

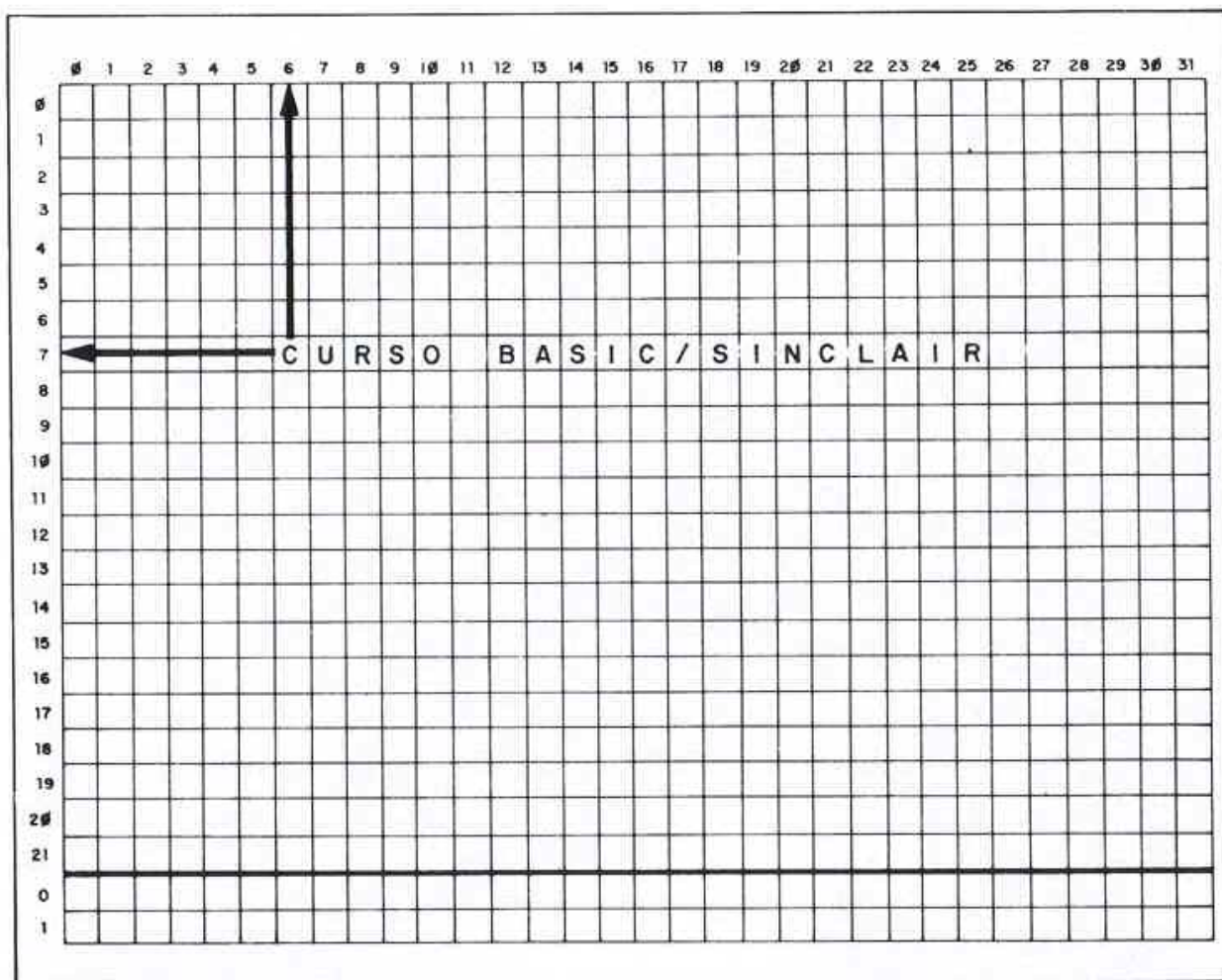
drada del argumento que va entre paréntesis. El cálculo se ha realizado en varias etapas, primeramente se han evaluado las partes comunes; a la variable «raíz» se le ha asignado el resultado de:

$$\sqrt{b^2 - 4ac}$$

y a la variable «divisor»

$$2a$$

posteriormente y a partir de estas dos variables se han obtenido las dos



Print AT 7,6.

raíces, «raíz 1» y «raíz 2».
 41φ — 42φ : Visualización de los resultados.

Programa «INTERÉS»

Grábelo en cinta de la forma:

SAVE "interés"

Este programa calcula el *interés simple* de un capital colocado en un banco durante cierto número de años. La fórmula del interés simple implementada en el programa es:

$$I = \frac{C \cdot R \cdot T}{100}$$

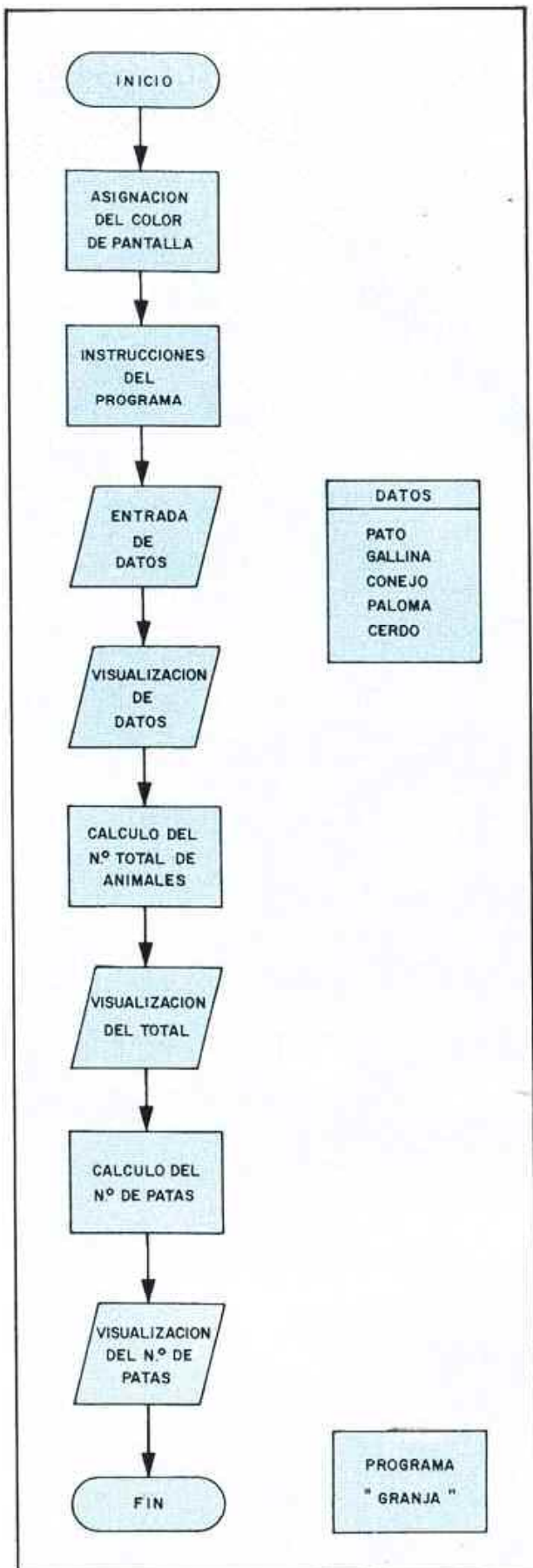
El valor asignado a la variable «capital» debe estar expresado en *pesetas*, el de la variable «reditos» en %, es decir, si el banco proporciona unos intereses al 3%, el valor a introducir deberá ser «3», y por último el asignado a la variable «tiempo» deberá ser expresado en *años*.

La estructura del programa es la siguiente:

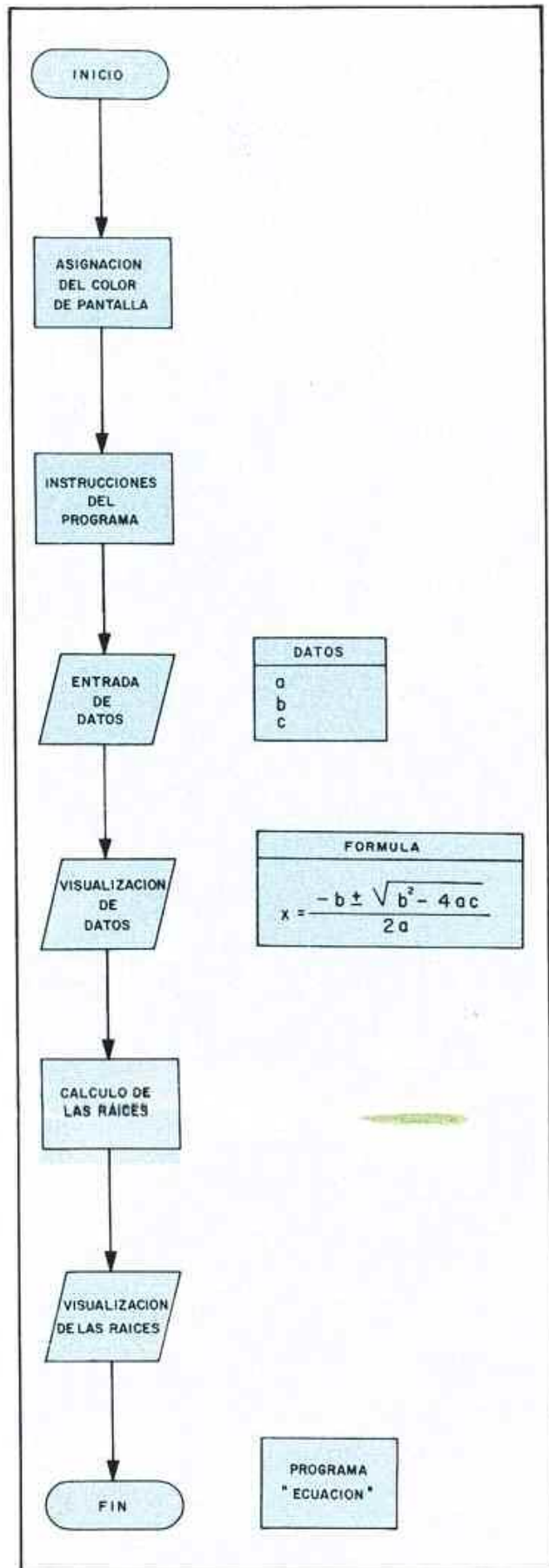
La estructura del programa es la siguiente:

- 1φ : Comentario con el nombre del programa.
- 11 : Asignación del color azul para el borde y el fondo y blanco para los caracteres.
- 13 — 18 : Visualización de una breve descripción del

programa. La segunda sentencia de la línea 18 es «PAUSE 3φφ», ésta proporciona una temporización de aproximadamente seis segundos, desde que aparece la información hasta que se borra con la siguiente sentencia (CLS). Si durante la temporización se pulsa una tecla, ésta termina y se ejecuta la instrucción siguiente.



Programa "Granja".



Programa "Ecuación".



- 2ϕ — 7ϕ : Entrada y visualización del contenido de las variables «capital», «réditos» y «tiempo».
- 9ϕ : Cálculo de los intereses. A la variable «interés» se le asigna el resultado de la fórmula.
- 11ϕ : Visualización de los resultados.

Programa «GRADOS»

Almacenarlo en cinta, por ejemplo, de la forma:

SAVE "grados"

El programa «GRADOS» consta de dos partes, en la primera transforma un valor de grados *centígrados* (°C), introducido por teclado, en grados *Fahrenheit* (°F) de acuerdo con la fórmula:

$$^{\circ}\text{F} = \frac{9}{5} ^{\circ}\text{C} + 32$$

La variable «c» contiene los grados centígrados a transformar y la variable «fahrenheit» el resultado.

En la segunda parte hace la transformación inversa, es decir, transforma un valor de grados Fahrenheit en centígrados, la fórmula implementada en este caso es:

$$^{\circ}\text{C} = (^{\circ}\text{F} - 32) \cdot \frac{5}{9}$$

La variable «f» contiene los grados fahrenheit a transformar y la variable «centígrados» el resultado.

El programa ha sido estructurado de la siguiente manera:

- 1ϕ : Comentario con el nombre del programa
- 2ϕ : Asignación del color azul para el borde y el fondo y blanco

para los caracteres.

- 4ϕ — 13ϕ : Breve descripción del programa. En la línea 11ϕ se utiliza el canal de comunicación ϕ.
- 15ϕ — 16ϕ: Entrada y visualización de la variable «c».
- 18ϕ — 19ϕ: Cálculo y visualización del resultado en grados Fahrenheit.
- 21ϕ — 22ϕ: Entrada y visualización de la variable «f».
- 24ϕ — 25ϕ: Cálculo y visualización del resultado en grados centígrados.

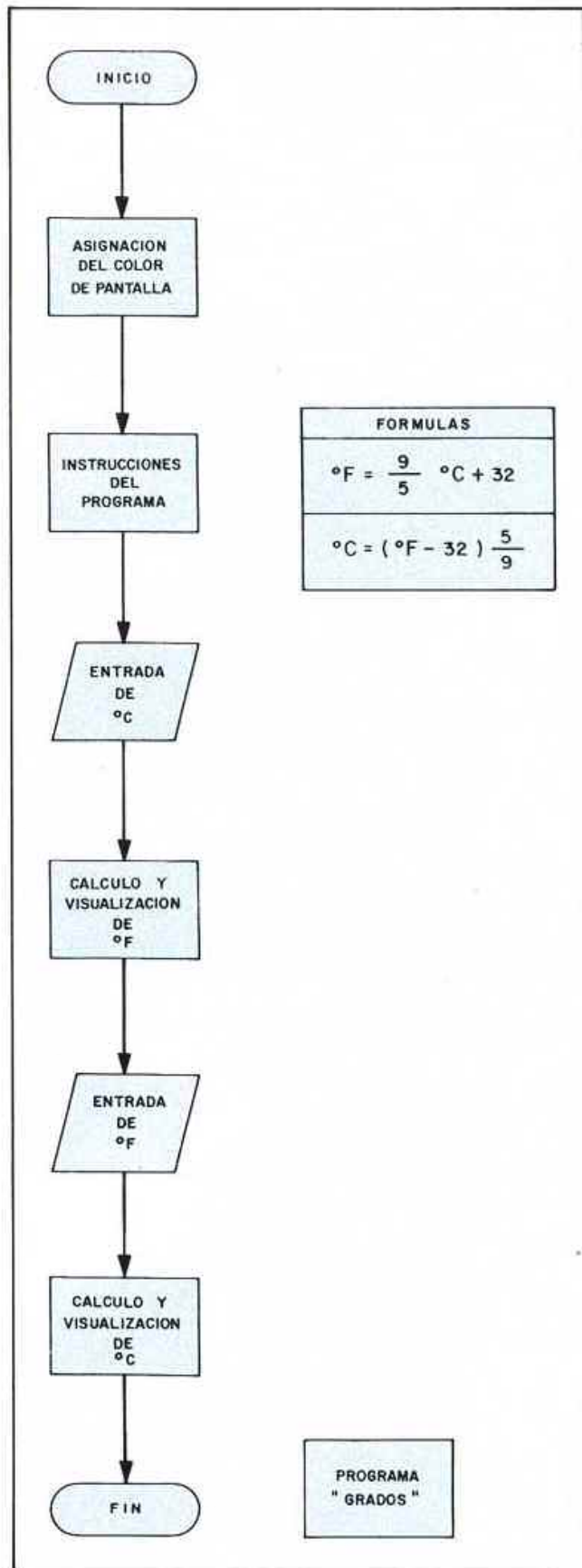
Programa «FICHA»

Salvarlo de la forma habitual:

SAVE "ficha"



Programa "Interés".



Programa "Grados".



Programa "Ficha".

Este programa simula una ficha con diversos *campos* (nombre, apellidos, etc...) que el usuario debe rellenar. Este programa puede formar parte de otro mayor que almacene, por ejemplo, los datos de los empleados de una empresa, los datos de los clientes o los suministradores.

Al editar el programa, debe poner atención en las líneas

5 ϕ a 11 ϕ , ya que van incluidos algunos de los gráficos predefinidos que incorpora el Spectrum. Debe pasar a modo **G** (gráficos).

Las líneas 8 ϕ a 10 ϕ forman lo que en programación se llama *bucle*, por tanto la instrucción 9 ϕ se repetirá 2 ϕ veces pero con diversos valores de la variable «n», ya que el índice del bucle está comprendido

entre los márgenes 1 y 2 ϕ . Los bucles «FOR»...«NEXT» serán estudiados con mayor detalle en otro capítulo.

La sentencia «FLASH 1» antepuesta al símbolo «<» (menor que) hace que este parpadee en la pantalla para llamar la atención sobre el dato a introducir.

Y por último la sentencia «PAUSE 2 $\phi\phi$ » temporiza, aproximadamente, la ejecución del programa durante cuatro segundos.

La estructura del programa es la siguiente:

- 1 ϕ : Comentario con el nombre del programa.
- 2 ϕ : Asignación de los colores de la pantalla, azul, para el borde y papel, y amarillo para los caracteres.
- 5 ϕ — 11 ϕ : Dibujo del contorno con los gráficos predefinidos.
- 13 ϕ — 21 ϕ : Visualización de los campos de la ficha.
- 22 ϕ : Temporización.
- 24 ϕ — 50 ϕ : Entrada de los datos y visualización de estos en los campos correspondientes.
- 51 ϕ : Utilización del canal de comunicación ϕ para visualizar el informe «Fin de edición».
- 52 ϕ : Temporización.

COMANDOS DE CONTROL

RUN

Acceso al teclado

INT



MODO **K** **VERIFY**

Definición

«RUN» se utiliza normalmente como comando directo y permite al usuario, mediante su ejemplo, ejecutar un programa editado en lenguaje BASIC.

La estructura general de este comando es:

| SENTENCIA | ARGUMENTO |
|-----------|--------------|
| RUN | N.º de línea |

Ejemplos:

- RUN 30
- RUN 10
- RUN

Si no se especifica número de línea, el intérprete BASIC tomará, por defecto, como primera línea a ejecutar, la de numeración más baja.

Cuando el número de línea especificado en el argumento no exista, la ejecución del programa comenzará en la línea siguiente; si esta tampoco existiera, por que se encuentra fuera de la zona de nuestro programa BASIC, no se ejecutará el programa y además aparecerá el mensaje:

0 OK, 0:1

Edite el siguiente programa:

```

1 REM *****
      PRUEBA
*****
10 PRINT "Linea 1"
20 PRINT "Linea 2"
30 PRINT "Linea 3"
40 PRINT "Linea 4"
50 PRINT "Linea 5"
61439 PRINT "Linea 6"
70 PRINT "Linea 7"

```

Ejécutele de las siguientes formas y compare los resultados:

- RUN
- RUN 10
- RUN 20
- RUN 30
-
-
-
- RUN 70

compare también los resultados proporcionados por los siguientes comandos directos:

- RUN 35
- RUN 40

Si en el argumento se especifica un número de línea comprendido entre "32768" y "61439" aparece el mensaje de error:

N Statement lost, 0:255

si es mayor a este último valor (61439) aparece:

B Integer out of range, 0:1

Una de las particularidades de la sentencia «RUN» es que borra la pantalla antes de ejecutar el programa almacenado en memoria.

Ejecute varias veces seguidas el siguiente programa y observe el efecto:

```

10 FOR n=1 TO 70
20 PRINT "randomize "
30 NEXT n

```

Otra de las particularidades, es que borra todas las varia-

bles que hasta ese momento estuvieran definidas.

Ejemplo:

- Edite estas dos líneas:

```

10 PRINT a
20 PRINT b

```

- Agisne unos valores a las variables «a» y «b» con comandos directos, por ejemplo:

```

LET a = 20
LET b = 127

```

- Compruebe los contenidos de dichas variables con:

```

PRINT a
PRINT b

```

- Ejecute el programa con «RUN», observará que en esta ocasión aparece el mensaje:

2 Variable not found, 10:1

ya que al ejecutarse «RUN» se han borrado las variables «a» y «b».

La sentencia «RUN» también puede ser incluida como línea dentro de un programa.

```

10 REM *****
      SIN FIN
*****
20 PRINT "Este programa solo
puede"
30 PRINT "pararse, pulsando"
40 PRINT "CAPS SHIFT + SPACE"
50 RUN

```

```

10 REM *****
      SIN FIN 2
*****
20 INPUT "Cuales tu nombre?"
30 PRINT "Te llamas " ; n$
40 PRINT #0, " Pulsa una tecla
para comenzar"
50 PAUSE 0
60 RUN

```

en estos dos ejemplos, el programa comienza a ejecutarse de nuevo al llegar a la última sentencia, de esta forma se crea un bucle sin fin.

El argumento también puede ser una variable numérica previamente definida.

Ejemplo:

```

1 REM *****
  RUN línea
*****
2 INPUT "En que línea quieres
empezar la ejecución del programa? " línea
4 RUN línea
10 PRINT "Ejecuto la 10"
20 PRINT "Ejecuto la 20"
30 PRINT "Ejecuto la 30"
40 PRINT "Ejecuto la 40"
50 PRINT "Ejecuto la 50"
60 PRINT "Ejecuto la 60"
  
```

Una vez ejecutado el programa, éste pide que le introduzcamos el n.º de línea de la nueva ejecución, valor asignado a la variable «línea», la sentencia «RUN línea» lo hace a partir de este valor.

BREAK

Acceso al teclado



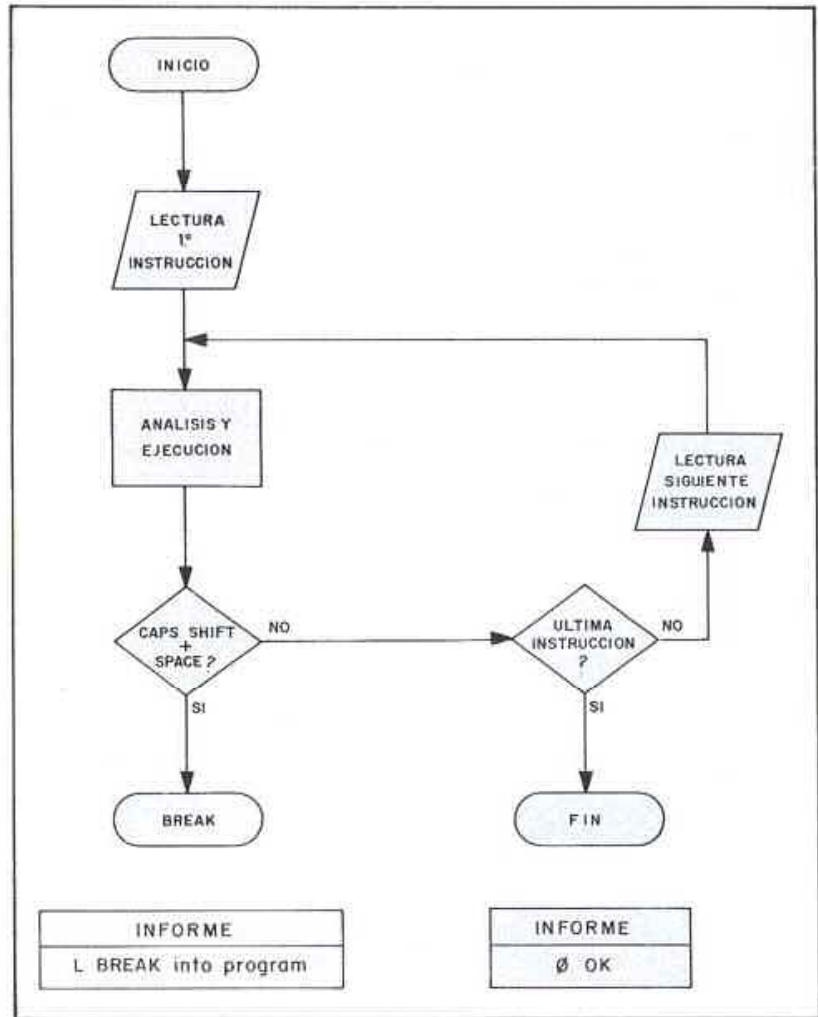
+



Definición

La función «BREAK» provoca una ruptura en la ejecución de un programa, en el acceso a los periféricos «impresora» y «cassette» y en los listados de más de 22 líneas, es decir, en

90 MICROBASIC



Análisis sentencia «Break».

aquellos en que aparece el mensaje:

scroll?

Esta ruptura sólo provoca una *interrupción* en la ejecución del programa, es decir, que *no borra* el contenido de la memoria. En la mayoría de los casos se podrá continuar con ella, utilizando el comando «CONTINUE» (CONT).

ADVERTENCIA

Si se está ejecutando una sentencia del tipo «INPUT», no se puede provocar la ruptura del programa, esto se consigue utilizando otra técnica que posteriormente será descrita.

El intérprete BASIC al *terminar* de ejecutar una instrucción verifica si están pulsadas las teclas «CAPS SHIFT» y «SPACE», si no lo están continúa con la ejecución de la siguiente instrucción, y si por el contrario, lo están, provoca su interrupción.

En aquellas instrucciones en que el tiempo de ejecución es prolongado, es necesario *mantener* estas teclas oprimidas hasta que aparezca el informe correspondiente.

Ejemplo:

```

1Ø BEEP 2, 1Ø
2Ø BEEP 3, 2Ø
3Ø BEEP 1, 15
4Ø BEEP 3, 5
  
```

Ejecute estas sentencias y utilice la función «BREAK».

Dependiendo de la situación en que se utilice «BREAK», existen dos tipos de informes. Cuando se utiliza para interrumpir un programa, el informe visualizado en pantalla es:

L BREAK into program

En los restantes casos, con sólo mantener pulsada la tecla «SPACE» (BREAK) se consigue la interrupción, y el mensaje presentado por el ordenador es:

D BREAK — CONT repeats

La diferencia entre estos dos mensajes será explicada con detalle al tratar la sentencia «CONTINUE» (CONT).

Al final del informe aparece la línea y el número de sentencia, dentro de la línea donde se produjo la interrupción.

STOP

Acceso al teclado

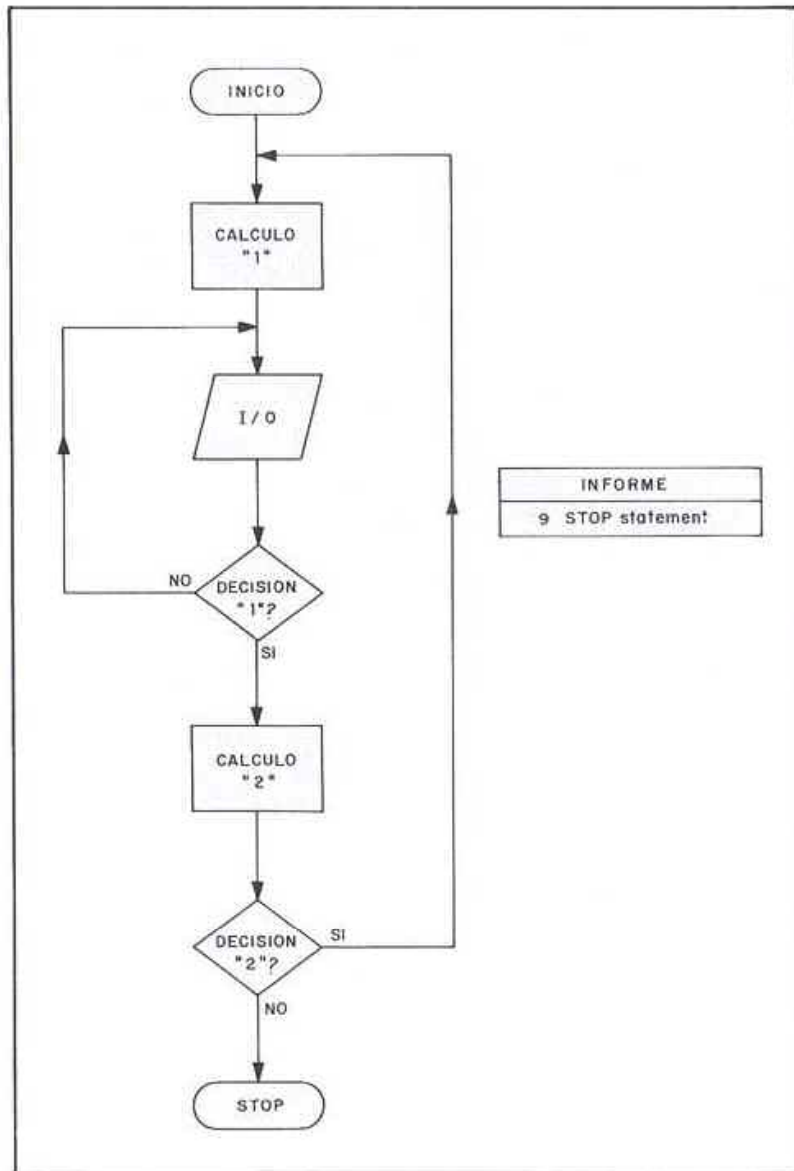


READ



Tipo de sentencia

Comando de programación.



Ejemplo sentencia «Stop».

Definición

A pesar de ser un comando de programación, «STOP» es tratado en este capítulo, ya que su función es de control, cuando se ejecuta se interrumpe el programa.

«STOP» no tiene sentido como comando directo, por lo que debe ser editado como línea de programa *sin argumento*.

Ejemplo:

400 STOP

cuando el intérprete BASIC analiza esta sentencia, se detiene en la línea 400 y presenta el mensaje:

9 STOP statement, 400:1

Esta sentencia puede ser utilizada en diversas ocasiones, pero principalmente:

a) Para separar diversas rutinas independientes dentro de un solo programa.

Ejemplo:

Edite el programa «1» que simula una calculadora básica.


```

280 STOP
290 REM
      *****
      *
      * MULTIPLICACION *
      *
      *****

300 PRINT AT 3,8;"MULTIPLICACION"
310 INPUT "Multiplicando? ";mul
320 INPUT "Multiplicador? ";mult
330 LET multiplicacion=mul*mult
340 CLS
350 PRINT mul;" * ";mult;" = ";
multiplicacion
360 STOP
370 REM
      *****
      *
      * DIVISION *
      *
      *****

380 PRINT AT 3,12;"DIVISION"
390 INPUT "Dividendo? ";div
400 INPUT "Divisor? ";divi
410 LET division=div/divi
420 CLS
430 PRINT div;" / ";divi;" = ";
division
440 STOP

```

PROGRAMA 2

```

10 REM *****
   *
   * CURSO BASIC *
   *
   * *****
   *
   * !!! NEW !!! *
   *
   * *****

20 BORDER 6: PAPER 6: INK 1: C
LS
30 REM *****
   *
   * DIBUJO CRONO *
   *
   * *****

40 PRINT AT 10,11: PAPER 2; IN
K 4; FLASH 1; "
50 FOR n=11 TO 13
60 PRINT AT n,11: PAPER 2; INK
4; FLASH 1; "
70 NEXT n
80 PRINT AT 14,11: PAPER 2; IN
K 4; FLASH 1; "
90 PRINT AT 12,13; "10:00"

```

Cuando son del tipo numérico basta simplemente con teclear la sentencia «STOP» (SYMBOL SHIFT + A) y «ENTER», inmediatamente se provoca la ruptura del programa y aparece el mensaje:

H STOP in INPUT

Ejemplo:

```

1Ø INPUT "> ";a
2Ø PRINT a
3Ø GO TO 1Ø

```

Si intenta utilizar la función «BREAK», observará que no sirve en estos casos.

Puede obtener la ruptura también, de una forma menos elegante, tecleando letras aleatoriamente, de esta manera el intérprete BASIC al analizar la entrada de datos y comprobar que no corresponde con un valor numérico o con una variable previamente definida, visualizará el mensaje:

2 Variable not found

Cuando el «INPUT» es del tipo alfanumérico, la técnica es ligeramente distinta.

Ejemplo:

```

1Ø INPUT "? ";a$
2Ø PRINT a$
3Ø GO TO 1Ø

```

Intente introducir «STOP», observará que el programa no se interrumpe, ya que la variable alfanumérica «a\$» asume el código correspondiente al token «STOP», y se ejecuta la siguiente instrucción, y así sucesivamente. La única manera de introducir «STOP», sin que lo tome como valor alfanumérico, es eliminando las comillas ("").

Para borrar las comillas, existen dos métodos. El primero es utilizando la función «DELETE», de esta forma desaparece la *comilla* situada a la izquierda del cursor. A partir de este instante ya se puede introducir la sentencia «STOP»; el mensaje que aparece es también:

H STOP in INPUT

En el segundo método se utiliza la función «EDIT», de esta manera desaparecen ambas comillas y al igual que en el caso anterior, se puede introducir «STOP».

Para cortar un «INPUT LINE», la filosofía es totalmente distinta, ya que ni se permite introducir «STOP», ni eliminar las comillas, simplemente por que estas no existen.

Ejemplo:

```
1Ø INPUT " "; LINE a$
dØ PRINT a$
3Ø GO TO 1Ø
```

La única manera de salir del programa anterior, una vez ejecutado, es utilizando el cursor de desplazamiento inferior («CAPS SHIFT» + «6»). El mensaje presentado, al igual que en las anteriores situaciones es:

H STOP in INPUT

CONTINUE

Acceso al teclado

L PRINT



MODOS **K**

PAPER

94 MICROBASIC

```
100 PRINT AT 3,0;"Este programa
se autodestruira" at llegar
la cuenta a 00:00"
110 REM
*****
*
* CUENTA ATRAS *
*
*****
120 FOR m=9 TO 0 STEP -1
130 PRINT AT 12,13;"0";m
140 FOR s=59 TO 0 STEP -1
142 IF s<10 THEN PRINT AT 12,16
;0;s: GO TO 155
150 PRINT AT 12,16;s
155 BEEP 0.02,20
160 NEXT s
170 NEXT m
180 REM
*****
*
* !!! BOOM !!! *
*
*****
190 FOR n=0 TO 7
200 BORDER n
210 BEEP 0.03,n
220 NEXT n
230 CLS
235 FOR n=1 TO 50
240 PRINT OVER 1;AT 10,8;"!!!!
BOOM !!!!"
245 NEXT n
250 REM
*****
*
* !!! NEW !!! *
*
*****
260 NEW
```

Definición

El comando directo «CONTINUE» se reconoce en el teclado por su forma abreviada «CONT». La utilidad de este comando es continuar con la ejecución de un programa que, debido a un informe de error o a un «BREAK» se ha interrumpido. Este comando no precisa argumento.

Cuando la interrupción se ha debido a un «BREAK» con informe:

L BREAK into program

o a una sentencia «STOP»:

9 STOP statement

el comando «CONTINUE» comienza a ejecutar el programa a partir de la siguiente instrucción, en que se produjo la ruptura.

Sin embargo, cuando el informe presentado por «BREAK» es:

D BREAK — CONT repeats

o se visualiza:

H STOP in INPUT

| TIPO DE "INPUT" | METODO | |
|-----------------|----------|-------------------|
| NUMERICO | STOP | |
| ALFANUMERICO | ELIMINAR | DELETE |
| | COMILLAS | EDIT |
| | + STOP | |
| LINE | ↓ | CAPS SHIFT + 6 |

Ruptura en «Input».

«CONTINUE» repite la ejecución en la misma línea donde se provocó la interrupción.

Ejemplo:

Introduzca las siguientes líneas

```
1Ø LOAD ""
2Ø PRINT "cargado"
```

cuando está en la fase de «carga» pulse la tecla «SPACE». Una vez interrumpido teclee «CONT», el programa volverá a ejecutar la sentencia «LOAD».

Si el programa se ha interrumpido debido a un error, podemos subsanar el problema momentáneamente y continuar con la ejecución.

Ejemplo:

```
10 REM *****
      FALLO *****
20 INPUT "Coordenada x? ",x
30 PRINT x+z
40 INPUT "Coordenada y? ",y
50 PRINT y+z
60 GO TO 10
```

Este programa una vez ejecutado e introducido el valor de la «coordenada X» presenta un fallo en la línea 3Ø, ya que no está definida la variable «Z». Defínala con un comando directo como por ejemplo:

```
LET Z = 5
```

y tecleando «CONTINUE» volverá a ejecutarse el programa a partir de la línea 3Ø

«CONTINUE» *no se puede* emplear con comandos directos. Se pueden distinguir tres casos:

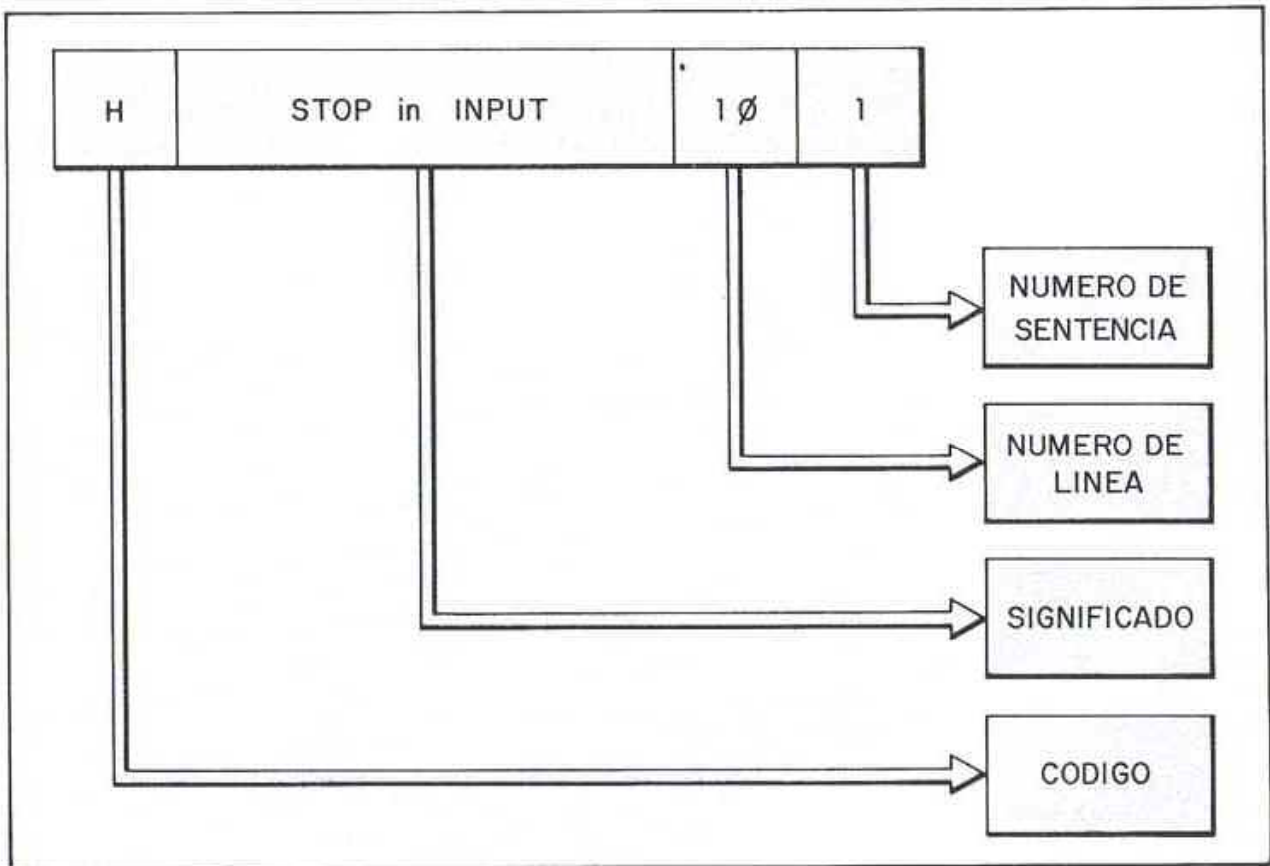
a) Cuando se interrumpe en la primera sentencia.

Ejemplo:

```
LOAD "" : LET a = 2 : PRINT a
```

si se pulsa «SPACE» (BREAK) y se pretende continuar con la ejecución de las restantes sentencias, el programa pierde el control y se queda un bucle sin fin, para salir de él pulse la tecla «CAPS SHIFT» + «SPACE».

b) Cuando se interrumpe en la segunda sentencia.



Informes.

Ejemplo:

```
PRINT "hola":LOAD"":PRINT "FIN"
```

En este caso al pretender continuar una vez realizada la interrupción se nos presenta el mensaje:

Ø OK

- c) Cuando se interrumpe en la tercera o siguiente sentencias.

Ejemplo:

```
INPUT "> "; a : PRINT a :  
LOAD"": LETb =2
```

aparece el mensaje:

N Statement lost

al intentar continuar con la ejecución una vez interrumpido en la sentencia «LOAD».

Informes de pantalla

Para comunicarnos el ordenador el resultado de la ejecución de los programas, tanto si han sido completados, interrumpidos, o con errores, utiliza estas dos líneas inferiores de la pantalla para enviarnos dichos informes.

Estos informes son visualizados de acuerdo a un formato.

CODIGO: Número comprendido entre «Ø» y «9» o letra de la «A» a la «R». Facilita la búsqueda en la tabla general de informes, apéndice B del manual (pág. 189).

SIGNIFICADO: Breve descripción del evento ocurrido. Para una mayor aclaración debe consultarse el manual.

LINEA: Número de línea donde se ha producido el evento.

SENTENCIA: Número de sentencia dentro de la línea.

Ejemplo:

Veamos el significado del informe:

H STOP in INPUT, 1 Ø : 1

«H» es el código y significa que ha ocurrido una interrupción del programa en la sentencia primera de la línea 1Ø; dicha interrupción ha sido debida a que se ha introducido «STOP» en una sentencia del tipo «INPUT».

Un informe con número de sentencia «2» se refiere a la sentencia situada a continuación del primer separador (;) o de la palabra clave «THEN», y así sucesivamente.

Los comandos directos al no poseer número de línea, se indican en los informes como línea Ø.

NEW

Acceso al teclado



MODO **K**

Definición

Generalmente este comando se utiliza de forma directa y no precisa de ningún argumento para poderse ejecutar.

«NEW» borra el programa o programas almacenados en memoria, también borra el valor de las variables definidas. Hay una serie de *variables de sistema* que no se ven afectadas por este comando, entre ellas los GDU o gráficos definidos por el usuario.

Debe utilizarse con mucho cuidado ya que de lo contrario, podríamos borrar un programa que aún no ha sido salvado, cosa no muy agradable por cierto.

Cuando se ejecuta da la im-

presión de haber conectado el aparato de nuevo, ya que nos presenta el famoso mensaje inicial:

© 1982 Sinclair Research Ltd.

Puede incluirse con precaución dentro de un programa para dar por finalizada su ejecución y borrado.

El programa n.º «2» incorpora esta sentencia, «sálvelo antes de ejecutar».

CLS

Acceso al teclado



MODO **K**

Definición

El comando «CLS» puede ser utilizado tanto en modo di-

recto, como formando parte de un programa; no precisa de ningún argumento.

La función de este comando es borrar la pantalla de caracteres y gráficos, asumiendo ésta el color especificado en la última sentencia «PAPER» ejecutada con anterioridad. El color del *borde* de la pantalla no se ve afectado por esta sentencia.

Ejemplos:

- Introduzca el siguiente programa:

```
10 REM *****
   REM   | | | eee | | |
   REM   | | | | | |
20 FOR I=1 TO 704
30 PRINT "e";
40 NEXT I
50 REM *****
   REM   | | | BORDE (ROJO) | | |
   REM   | | | | | |
60 BORDER 2
```

este programa llena la zona de visualización con el símbolo «e», teclee el comando directo «CLS» y observe la pantalla.

- En el siguiente programa la sentencia «CLS» se utiliza para borrar la pantalla y asumir los colores «magenta» para el fondo y «amarillo» el de los caracteres; el color del borde se asigna directamente con la sentencia «BORDER».

```
10 REM *****
   REM   | | | ASIGNAR COLORES | | |
   REM   | | | | | |
20 REM ** BORDE "verde" **
30 BORDER 4
40 REM ** FONDO "magenta" **
50 PAPER 3
60 REM ** TINTA "amarilla" **
70 INK 6
80 CLS
90 PRINT AT 10,10,"MICROHOBBY"
```

```
105 IF AT>7 THEN LET C=0: GO
25 INPUT "BEACON? ";X$: FOR
T TO 7: IF X$=M$(Q) THEN GO TO
5
26 NEXT Q: GO TO 25
28 LET B=Q-T: LET B=B+7*(B=Q)
GO SUB 9510: INPUT "RADIAL?
RD: LET RD=RD*RAD
30 INPUT "HEADING? ";HD: LE
HD=HD*RAD: INPUT "DME? ";DM:
T S=DM*COS RD: LET W=DM*SIN RD
40 LET S=S+SF*COS HD+W$*COS
: LET W=W+SF*SIN HD+W$*SIN WD:
ET DM=SQR (S*S+W*W): LET RD=AC
(S/DM): IF W<0 THEN LET RD=CR-
50 LET HD=HD+RL/(SF*7E3): IF
D>CR THEN LET HD=HD-CR
55 IF HD<0 THEN LET HD=HD+CR
60 IF C AND DM<6 THEN LET B=
GO TO EX
65 LET G=G-(PR+20)/M1: IF G<
0 THEN GO SUB 9644-M1
```

Ejemplo de listado.

SALTOS INCONDICIONALES Y CONDICIONALES

Hay ocasiones en que por razones de estructura de un programa, interesa que las instrucciones no se ejecuten de forma secuencial, es decir, una detrás de otra, sino que, por el contrario realicen *saltos*; éstos pueden clasificarse dependiendo de su función en:

- INCONDICIONALES.
- CONDICIONALES.

Como su propio nombre indica, un salto *incondicional* es aquel que salta directamente al número de línea especificado en el argumento, sin embargo, los condicionales necesitan que se cumpla previamente la condición prevista en la instrucción.

GO TO

Acceso al teclado



MODO **K**

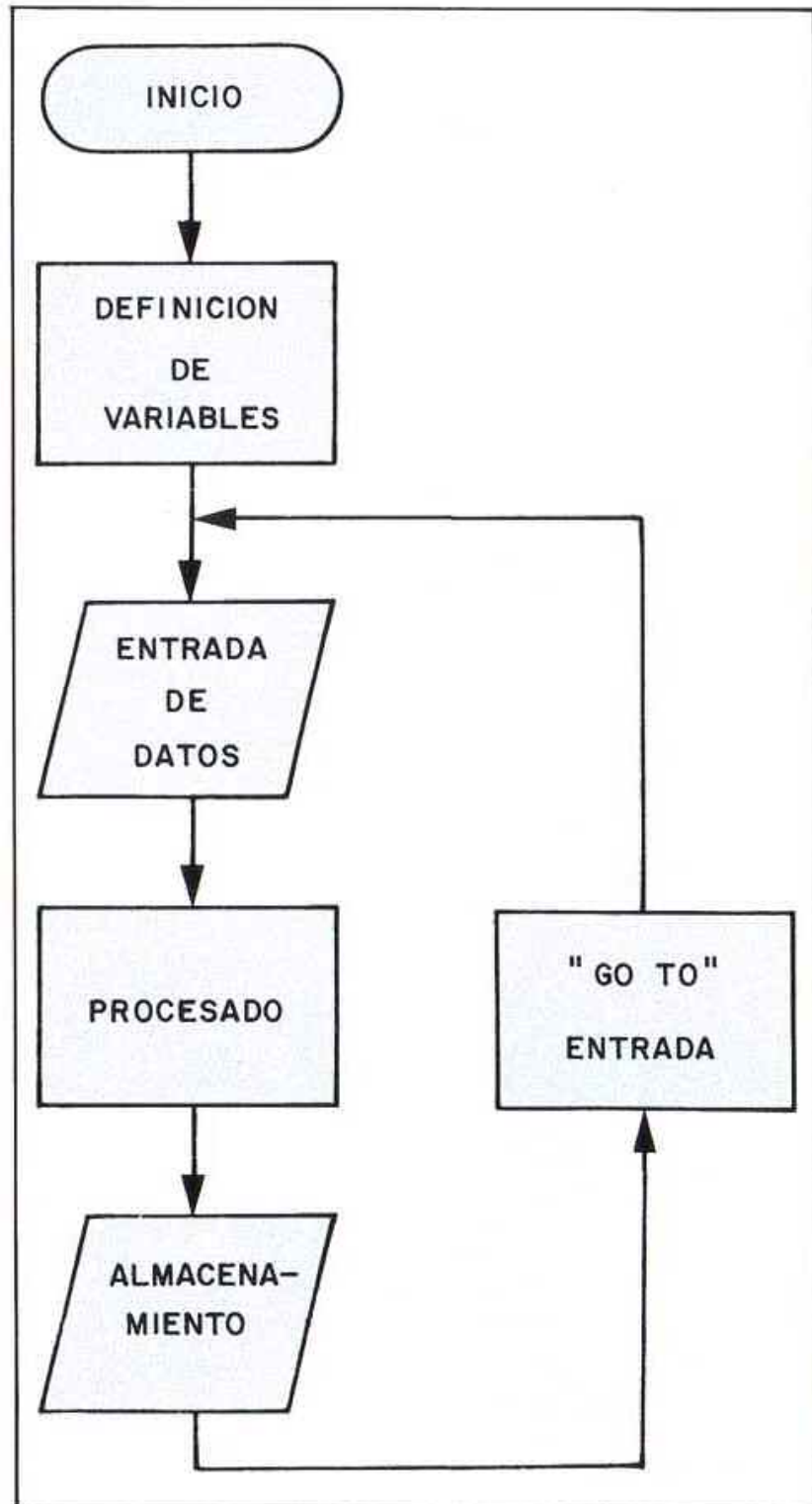
Tipo de sentencia

Comando de programación.

Definición

La sentencia «GO TO» realiza los saltos incondicionales dentro de un programa, su estructura general es:

| SENTENCIA | ARGUMENTO |
|-----------|--------------|
| GO TO | N.º de línea |



Ejemplo sentencia "GO TO".

Ejemplos:

- GO TO 30
- GO TO 70

Una sentencia de este tipo transfiere la ejecución del programa a la línea especificada en su argumento.

En el siguiente programa, al analizar el intérprete BASIC la instrucción «40», la siguiente que ejecutará será la «10».

```

10 REM *****
      GO TO
*****
20 INPUT "teclea lo que quieras
9 pulse "ENTER" >>>> ";a$
30 PRINT a$
40 GO TO 10
  
```

En este otro se ha incluido un *índice* asignado a la variable «potencia» que se incrementa en uno cada vez que se ejecuta la línea «50», este índice sirve tanto para indicar el número de veces menos uno, que se realiza la operación matemática (2ⁿ) como para utilizarse como potencia de la misma.

```

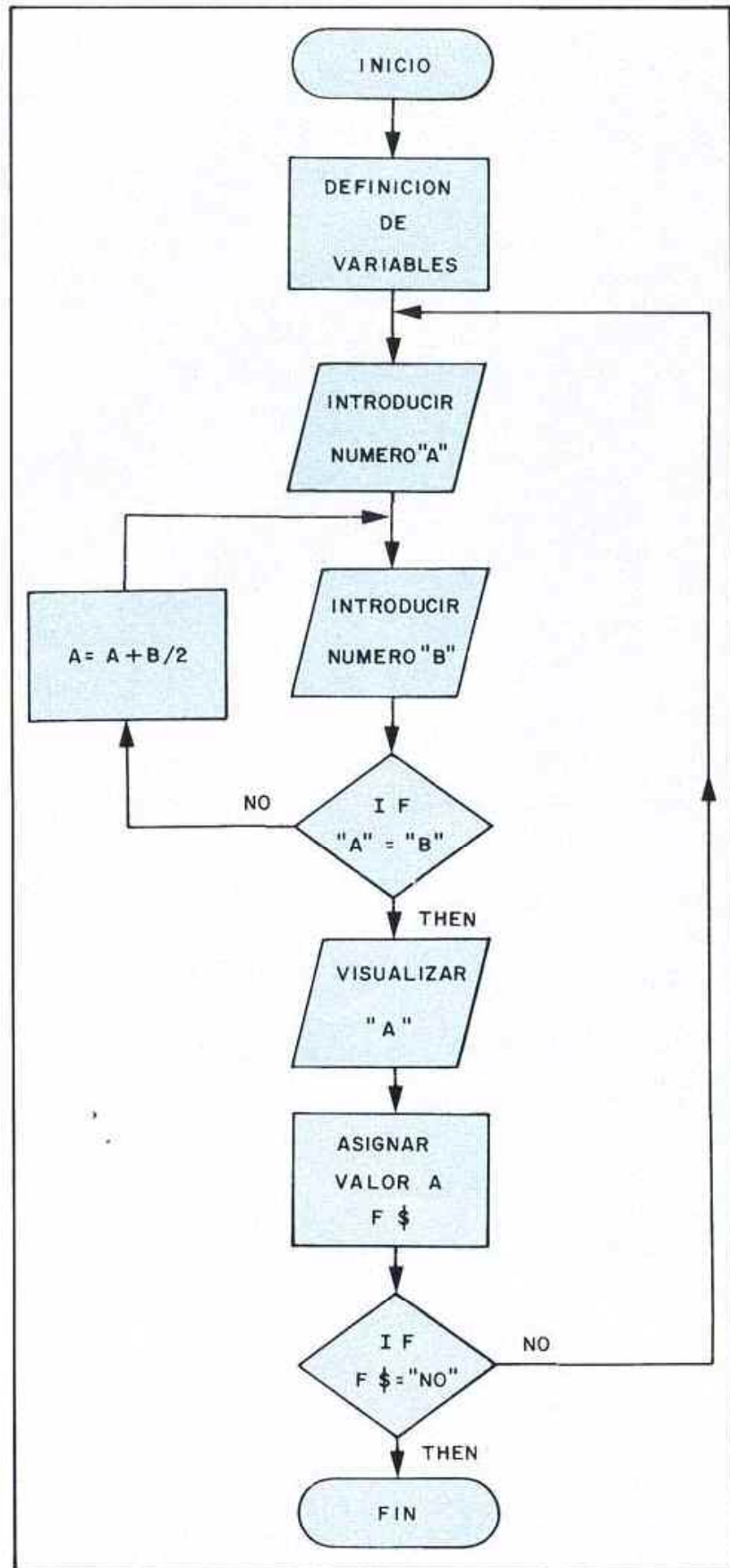
10 REM *****
      POTENCIAS
*****
20 LET potencia = 0
30 PRINT potencia;
40 PRINT " "; 2^potencia;
50 LET potencia=potencia+1
60 GO TO 30
  
```

Debe poner atención al calcular el número de línea donde desea que se realice el salto, ya que podrían no ejecutarse ciertas líneas intermedias.

Ejemplo:

```

10 REM *****
      INCORRECTO
*****
20 LET a$="MICRO"
30 LET b$="HOBBY"
40 LET c$="SEMANAL"
50 PRINT a$;
60 PRINT b$+ " ";
70 PRINT c$ " "
80 GO TO 60
  
```



Ejemplo salto y ejecución condicional (IF... THEN...).

en este ejemplo, solamente la primera vez se ejecuta el programa correctamente; en las siguientes, la variable «a\$» no se visualiza, por tanto la línea 8φ debería ser:

```
GO TO 5φ
```

«GO TO» también puede ser utilizado como comando directo, esta aplicación es bastante interesante, ya que permite ejecutar un programa sin alterar el contenido de las variables hasta ese momento definidas. En la depuración de programas se utiliza frecuentemente en sustitución del comando «RUN» que lo borra todo.

El argumento puede ser una variable de tipo numérico:

```

1 REM *****
  VARIABLE
*****
4 INPUT "A que línea hago el
"GO TO:" 10, 20, 30, 40, 50 o 60?
"línea
5 GO TO línea
10 PRINT "salto a la 10"
20 PRINT "salto a la 20"
30 PRINT "salto a la 30"
40 PRINT "salto a la 40"
50 PRINT "salto a la 50"
60 PRINT "salto a la 60"
70 PRINT
80 GO TO 4

```

una vez introducida la variable «línea» el control de ejecución salta hasta el valor especificado en ella.

```
IF ... THEN ...
```

Acceso al teclado



Tipo de sentencia

Comando de programación.

Definición

El grupo de sentencias «IF» y «THEN» permiten realizar los saltos o ejecutar una serie de instrucciones de una manera *condicional*, es decir, en función del resultado de una comparación.

Las estructuras básicas son:

a) Salto condicional.

| SENTENCIA | ARGUMENTO |
|--------------|---------------|
| IF condición | THEN GO TO... |

Ejemplo:

```

10 REM *****
  SALTO
*****
20 INPUT "Numero "A" : "a
30 INPUT "Numero "B" : "b
40 IF a=b THEN GO TO 70
50 LET a=a+b/2
60 GO TO 30
70 PRINT "Valor "A" : "a
80 PRINT "Fin"

```

cuando las variables «a» y «b» sean iguales, la ejecución del programa continuará en la línea 7φ, si no, se asignará a la variable «a» el valor de «a + b/2» y posteriormente, ejecuta un salto incondicional a la línea 3φ.

b) Ejecución condicional.

| SENTENCIA | ARGUMENTO |
|--------------|-----------------------|
| IF condición | THEN instrucciones... |

Ejemplo:

```

10 REM *****
  EJECUCION
*****
20 PRINT "MICROHOBBY"
30 INPUT "Quiere continuar? (S/N) : " ; $
40 IF $="N" THEN PRINT AT 10, 15, "FIN" : LET a=0 : LET b=0 : STO P
50 GO TO 20

```

si no se cumple que la variable de cadena «f\$» sea igual a «NO» se ejecuta la instrucción siguiente; si por el contrario lo son, se ejecutan las sentencias que acompañan al «THEN».

De lo explicado hasta este momento, se desprende que la sentencia «IF ... THEN ...» es como una encrucijada con dos caminos, donde el ordenador tiene que elegir uno de ellos.

OBSERVACION

Si se utiliza la sentencia «GO TO», dentro de la lista de instrucciones que deben ejecutarse si se cumple la condición impuesta en el «IF», ésta deberá ser colocada la última, ya que de lo contrario, quedarían sin ejecutar algunas instrucciones.

Ejemplo:

```

1φφ IF mes = 8 THEN PRINT
  "AGOSTO" : GO TO 12φ :
  LET mes = φ

```

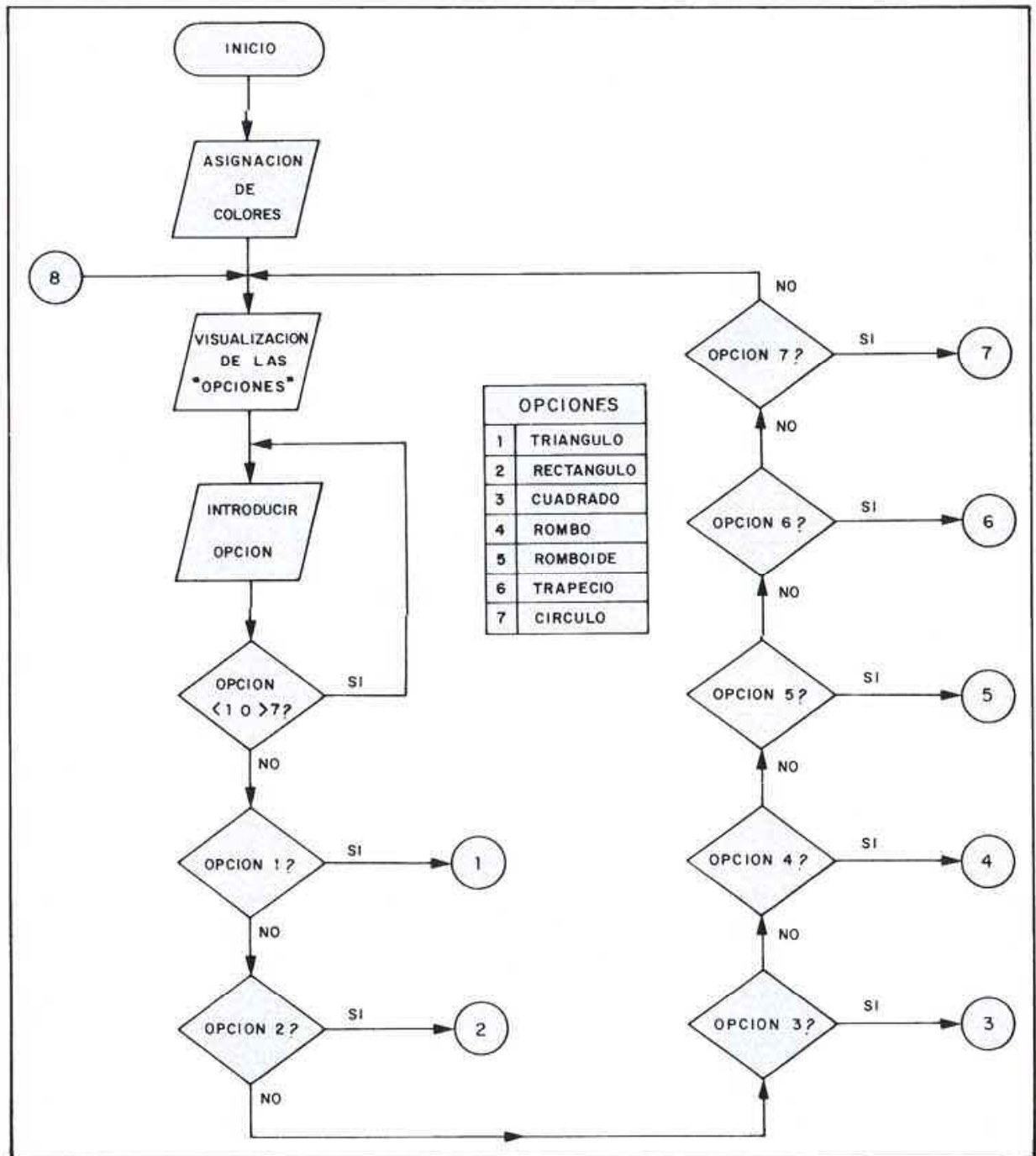
la asignación del valor «φ» a la variable «mes» no se realiza, ya que antes se ejecuta una instrucción de salto incondicional a la línea 12φ.

Para realizar las comparaciones, puede utilizarse cualquiera de los operadores *relacionales*:

| | |
|-----|---------------|
| > | MAYOR QUE ... |
| < | MENOR QUE ... |
| > = | MAYOR O IGUAL |
| < = | MENOR O IGUAL |
| < > | DISTINTO |

Ejemplos:

- IF a > b THEN ...
- IF J\$ < S\$ THEN ...
- IF n > = K + t THEN
- IF Kilo < = 7 THEN ...
- IF P\$ < > "SI" THEN...



Programa "Áreas" menú de opciones.

PROGRAMA 1

```

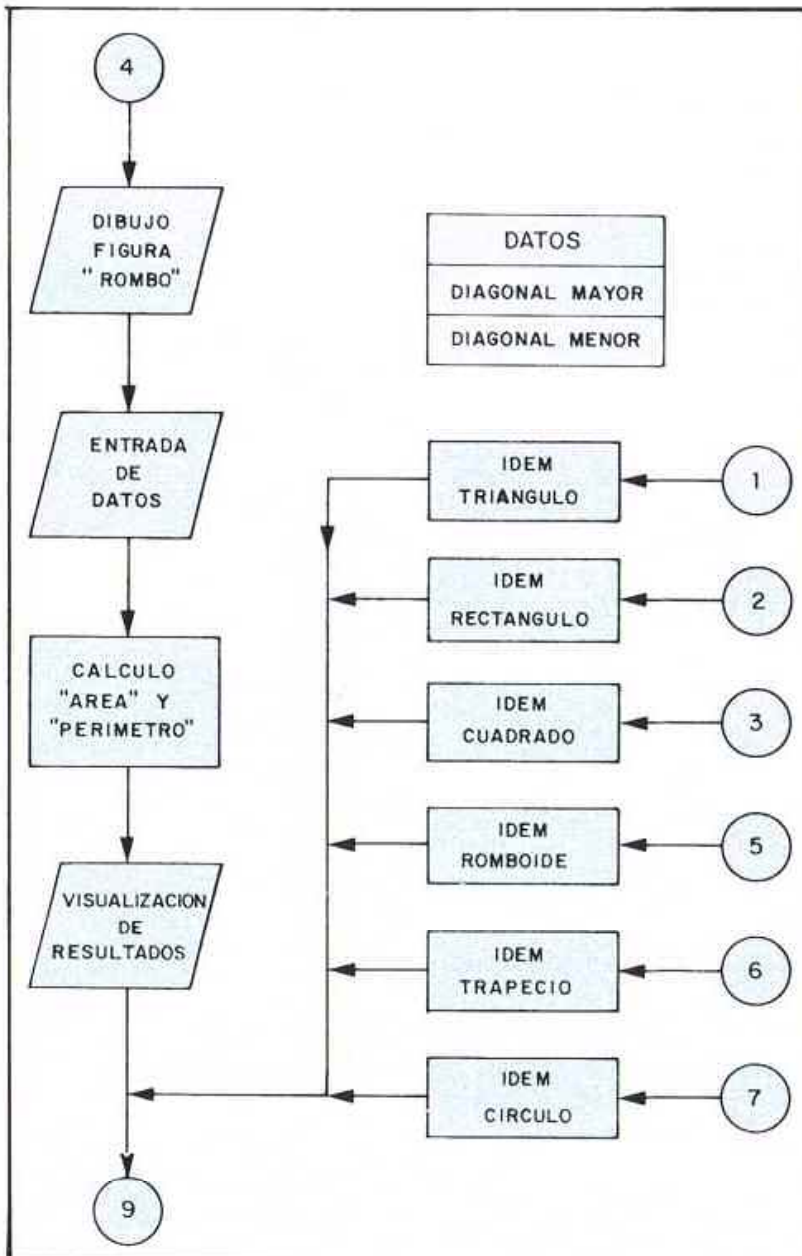
10 REM *****
   *          *
   * AGENDA  *
   *          *
   *          *
20 PRINT AT 0,4;"NOMBRE
TELEFONO"
25 PRINT "-----"
27 LET indice=1
30 INPUT "Nombre >>> "; LINE n
$

```

```

40 IF n$="FIN" OR n$="fin" THE
N GO TO 1000
50 PRINT n$,
60 INPUT "Telefono >>> "; telef
ono
70 PRINT telefono
80 IF indice=10 THEN GO TO 100
0
90 LET indice=indice+1
100 GO TO 30
1000 PRINT #0;AT 1,2;"<<<< Fin
de edicion >>>>"
1010 PAUSE 0

```



Programa "Areas" desarrollo opción "Rombo".

Los operadores *lógicos* «AND» y «OR» son utilizados cuando hay una combinación de condiciones dentro de una sentencia «IF ... THEN ...». El operador «AND» implica que deben cumplirse todas y cada una de las condiciones.

Ejemplo:

```
10 IF n >= 0 AND n <= 9 THEN ...
```

solamente se cumplen las dos condiciones, cuando la variable

«n» tiene un valor comprendido entre «0» y «9».

Utilizando el operador «OR» basta solamente con que se cumpla una de las condiciones previstas en la comparación:

Ejemplo:

```
10 IF A$ = "A" OR B$ <> "C" OR n > 30 THEN ...
```

en este ejemplo, se cumple la condición general en cualquiera de los siguientes casos:

- a) Cuando la variable A\$ tenga el valor "A".
- b) Cuando el contenido de la variable B\$ sea distinto de "C".
- c) Cuando "n" sea superior a "30".
- d) o con cualquier combinación de los anteriores casos.

También pueden combinarse los operadores «AND» y «OR». Ejemplo:

```
10 IF (J$ = "NO" AND n < 10) OR (J$ = "SI" AND (p = 10 OR t = 7)) THEN ...
```

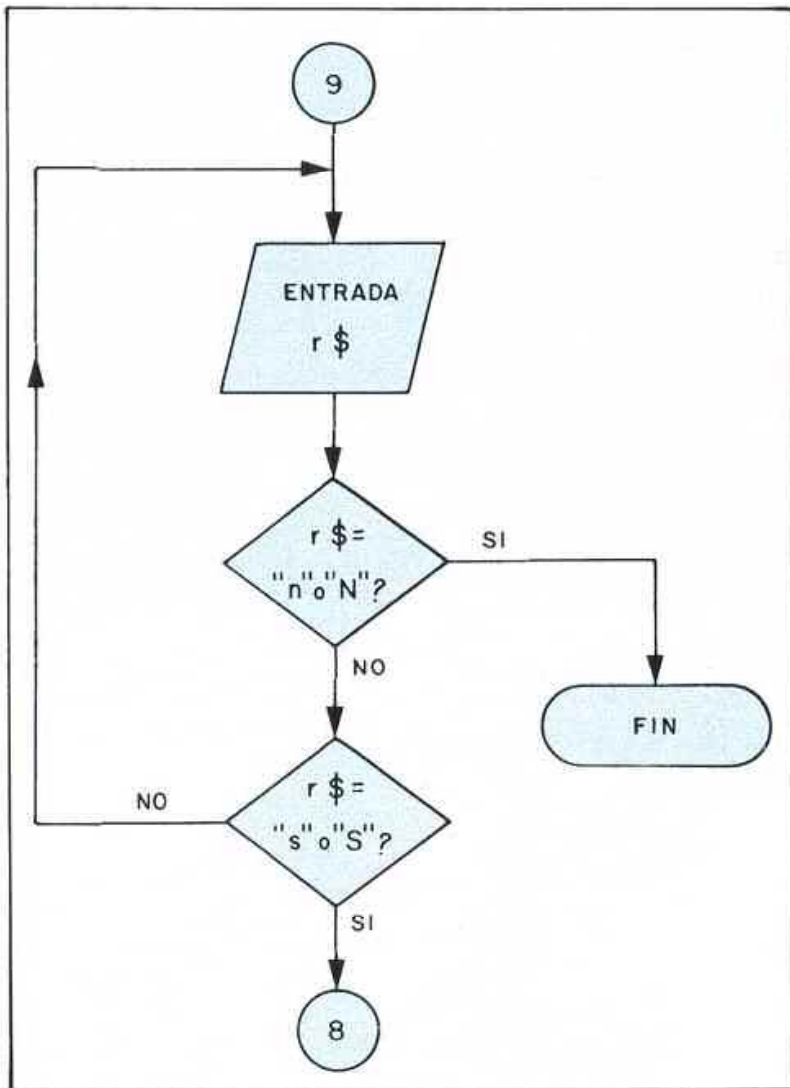
primero, se evalúan individualmente las condiciones encerradas entre paréntesis, y posteriormente, se evalúan entre sí los resultados parciales, por tanto, será necesario que se cumplan cualquiera de las siguientes condiciones:

- a) Cuando J\$ = "NO" y n < 10.
- b) Cuando J\$ = "SI" y p = 10.
- c) Cuando J\$ = "SI" y t = 7.
- d) Combinaciones de a, b y c.

El programa número «1» es un ejemplo de aplicación de las sentencias «IF ... THEN ...»; acaba su ejecución cuando se teclean dieciocho nombres con sus correspondientes teléfonos o cuando se introduce la palabra «FIN» o «fin» en el instante que el ordenador espera un nombre.

Evaluación de las condiciones

Cuando una condición se cumple, es decir, que es verdadera, se le asigna el valor "1" (distinto de 0) y cuando es falsa, el valor 0 (igual a 0). Para evaluar una condición compleja, primero se evalúan una a una, asignando los valores «0» o «1» según corresponda; pos-



Programa "Areas" Rutina "Fin".

teriormente, por parejas hasta que sólo quede una condición, si el resultado es « ϕ » el ordenador pasará a ejecutar la siguiente instrucción; si es «1», ejecutará antes las instrucciones contenidas en el «THEN...».

Ejemplo:

```
IF (a > b OR c = 7) AND (t < >
 $\phi$  AND p = a * 5) THEN ...
```

para los valores:

a = 4
b = 2
c = 6
t = 5 ϕ
p = 2 ϕ

104 MICROBASIC

Resolviendo por pasos:

a) (a > b) OR (c = 7)
(a > b) es verdadero (1), ya que «4» es mayor que «2».
(c = 7) es falso (ϕ), ya que «6» no es igual a «7».

Realizando la operación lógica «OR» de los valores anteriores

1 OR ϕ

observamos que el resultado es «1», ya que se cumple una de las condiciones. (Repasar el capítulo dedicado a «OPERADORES LOGICOS»).

b) (t < > ϕ) AND (p = a * 5)
(t < > ϕ) es verdadero (1), ya que «5 ϕ » es distinto de « ϕ ».
(p = a * 5) es verdadero (1), por que «2 ϕ » es igual a «4» por «5». Realizando la operación lógica «AND» de los valores anteriores «1» y «1»

1 AND 1

tenemos un resultado igual a «1» ya que se cumplen las dos condiciones.

c) (condición 1) AND (condición 2)

Como las dos condiciones se cumplen, el resultado global es también verdadero «1».

Asigne otros valores a las variables e intente resolver el resultado; si tiene algún problema o desea comparar los resultados ejecute el siguiente programa:

```

10 REM *****
   VERDADERO/FALSO
   *****
20 INPUT "A > " : a
30 INPUT "B > " : b
40 INPUT "C > " : c
50 INPUT "T > " : t
60 INPUT "P > " : p
70 IF (a > b OR c = 7) AND (t < >
AND p = a * 5) THEN PRINT "VERDA
DERO" : GO TO 20
80 PRINT "FALSO"
90 GO TO 20
  
```

El valor de una condición también puede ser asignado a una variable, de la forma:

```
LET resultado = f$ = "FIN"
```

la variable «resultado» tendrá el valor «1» cuando f\$ sea igual a la cadena «FIN», y « ϕ » cuando f\$ tenga otro valor. Podríamos, por tanto, editar una sentencia del tipo:

```
IF resultado THEN ...
```

PROGRAMA 2

```

10 REM *****
*          *
*  CURSO BASIC  *
*          *
*****
*          *
*   AREAS Y     *
*          *
* PERIMETROS    *
*          *
*****

20 BORDER 1: PAPER 1: INK 7: C
LS
30 REM *****
*          *
*  OPCIONES     *
*          *
*****

40 PRINT AT 1,0;"CALCULO DE AR
EAS
  @ MICROHOBBY"
50 PRINT AT 4,9;"MENU DE OPCIO
NES"
60 PRINT AT 5,7;"-----"

70 PRINT AT 8,8;"1 - TRIANGULO
"
80 PRINT AT 10,8;"2 - RECTANGU
LO"
90 PRINT AT 12,8;"3 - CUADRADO
"
100 PRINT AT 14,8;"4 - ROMBO"
110 PRINT AT 16,8;"5 - ROMBOIDE
"
120 PRINT AT 18,8;"6 - TRAPECIO
"
130 PRINT AT 20,8;"7 - CIRCUNFE
RENCIA"
140 INPUT "Introduzca la opcion
deseada >>> ";opcion
150 REM
*****
*          *
* VERIFICACION *
*          *
*****

160 IF opcion<1 OR opcion>7 THE
N GO TO 140
165 CLS
170 IF opcion=1 THEN GO TO 250
180 IF opcion=2 THEN GO TO 450
190 IF opcion=3 THEN GO TO 700
200 IF opcion=4 THEN GO TO 970
210 IF opcion=5 THEN GO TO 1170
220 IF opcion=6 THEN GO TO 1380
230 IF opcion=7 THEN GO TO 1630
240 GO TO 40
250 REM
*****
*          *
* TRIANGULO    *
*          *
*****

260 PRINT AT 2,11;"TRIANGULO"
270 PRINT AT 3,11;"-----"
280 PLOT INK 4,16,47
290 DRAW INK 4,63,0
300 DRAW INK 4,-63,73
310 DRAW INK 4,-31,-73
312 PRINT AT 17,5;"b"
314 PRINT AT 12,0;"h"
320 REM
*****
*          *
* ENTRADA DATOS *
*          *
*****

330 INPUT "Base >>> ";base
340 PRINT AT 7,11;"b : ";base
350 INPUT "Altura >>> ";altura
360 PRINT AT 9,11;"h : ";altura
370 REM
*****
*          *
* CALCULOS     *
*          *
*****

```

```

380 LET area=base*altura/2
390 PRINT AT 12,11;"AREA :
";area
400 LET lado=SOR (altura^2+(bas
e/2)^2)
410 LET perimetro=lado*2+base
430 PRINT AT 14,11;"PERIMETRO:
";perimetro
440 GO TO 9000
450 REM
*****
*          *
* RECTANGULO   *
*          *
*****

460 PRINT AT 2,11;"RECTANGULO"
470 PRINT AT 3,11;"-----"
480 PLOT INK 4,16,47
490 DRAW INK 4,63,0
500 DRAW INK 4,0,60
510 DRAW INK 4,-63,0
520 DRAW INK 4,0,-60
530 PRINT AT 17,5;"b"
540 PRINT AT 10,0;"h"
550 REM
*****
*          *
* ENTRADA DATOS *
*          *
*****

560 INPUT "Base >>> ";base
570 PRINT AT 7,13;"b : ";base
580 INPUT "Altura >>> ";altura
590 PRINT AT 9,13;"h : ";altura
600 REM
*****
*          *
* CALCULOS     *
*          *
*****

610 LET area=base*altura
620 PRINT AT 12,13;"AREA :
";area
630 LET perimetro=base*2+altura
640 PRINT AT 14,13;"PERIMETRO:
";perimetro
650 GO TO 9000
700 REM
*****
*          *
* CUADRADO     *
*          *
*****

710 PRINT AT 2,13;"CUADRADO"
720 PRINT AT 3,13;"-----"
730 PLOT INK 4,16,47
740 DRAW INK 4,63,0
750 DRAW INK 4,0,63
760 DRAW INK 4,-63,0
770 DRAW INK 4,0,-63
780 PRINT AT 17,5;"l"
790 REM
*****
*          *
* ENTRADA DATOS *
*          *
*****

800 INPUT "Lado >>> ";lado
900 PRINT AT 7,13;"l : ";lado
910 REM
*****
*          *
* CALCULOS     *
*          *
*****

920 LET area=lado^2
930 PRINT AT 12,13;"AREA :
";area
940 LET perimetro=lado*4
950 PRINT AT 14,13;"PERIMETRO:
";perimetro
960 GO TO 9000
970 REM

```

```

*****
* ROMBO *
*      *
*****
980 PRINT AT 2,14;"ROMBO"
990 PRINT AT 3,14;"-----"
1000 PLOT INK 4;40,40
1010 DRAW INK 4;24,48
1020 DRAW INK 4;-24,48
1030 DRAW INK 4;-24,-48
1040 DRAW INK 4;24,-48
1050 REM
*****
* ENTRADA DATOS *
*      *
*****
1060 INPUT "Diagonal mayor >>> "
;mayor
1070 PRINT AT 7,12;"D : ";mayor
1080 INPUT "Diagonal menor >>> "
;menor
1090 PRINT AT 9,12;"d : ";menor
1100 REM
*****
* CALCULOS *
*      *
*****
1110 LET area=mayor*menor/2
1120 PRINT AT 12,12;"AREA : "
;area
1130 LET lado=SOR ((mayor/2)2+
menor/2)2)
1140 LET perimetro=lado*4
1150 PRINT AT 14,12;"PERIMETRO: "
;perimetro
1160 GO TO 9000
1170 REM
*****
* ROMBOIDE *
*      *
*****
1180 PRINT AT 2,12;"ROMBOIDE"
1190 PRINT AT 3,12;"-----"
1200 PLOT INK 4;8,47
1210 DRAW INK 4;54,0
1220 DRAW INK 4;16,40
1230 DRAW INK 4;-54,0
1240 DRAW INK 4;-16,-40
1250 PRINT AT 17,5;"b"
1260 PRINT AT 13,4;"h"
1265 PRINT AT 13,0;"l"
1270 REM
*****
* ENTRADA DATOS *
*      *
*****
1280 INPUT "Base >>> ";base
1290 PRINT AT 7,13;"b : ";base
1300 INPUT "Altura >>> ";altura
1310 PRINT AT 9,13;"h : ";altura
1312 INPUT "Lado >>> ";lado
1314 PRINT AT 11,13;"l : ";lado
1320 REM
*****
* CALCULOS *
*      *
*****
1330 LET area=base*altura
1340 PRINT AT 14,13;"AREA : "
;area
1350 LET perimetro=lado*2+base*2
1360 PRINT AT 16,13;"PERIMETRO: "
;perimetro
1370 GO TO 9000
1380 REM
*****
* TRAPECIO *
*      *
*****

```

```

1390 PRINT AT 2,12;"TRAPECIO"
1400 PRINT AT 3,12;"-----"
1410 PLOT INK 4;8,47
1420 DRAW INK 4;80,0
1430 DRAW INK 4;-16,64
1440 DRAW INK 4;-48,0
1450 DRAW INK 4;-16,-64
1460 PRINT AT 17,5;"B"
1470 PRINT AT 12,3;"h"
1480 PRINT AT 6,5;"b"
1490 REM
*****
* ENTRADA DATOS *
*      *
*****
1500 INPUT "Base mayor >>> ";may
or
1510 PRINT AT 7,13;"B : ";mayor
1520 INPUT "Base menor >>> ";men
or
1530 PRINT AT 9,13;"b : ";menor
1540 INPUT "Altura >>> ";altura
1550 PRINT AT 11,13;"h : ";altur
a
1560 REM
*****
* CALCULOS *
*      *
*****
1570 LET area=(mayor+menor)*altu
ra/2
1580 PRINT AT 14,13;"AREA : "
;area
1590 LET lado=SOR (((mayor-menor
)/2)2+altura2)
1600 LET perimetro=mayor+menor+l
ado*2
1610 PRINT AT 16,13;"PERIMETRO: "
;perimetro
1620 GO TO 9000
1630 REM
*****
* CIRCUNFERENCIA *
*      *
*****
1640 PRINT AT 2,9;"CIRCUNFERENCI
A"
1650 PRINT AT 3,9;"-----"
1660 CIRCLE INK 4;40,88,32
1670 PRINT AT 9,3;"r"
1680 REM
*****
* ENTRADA DATOS *
*      *
*****
1690 INPUT "Radio >>> ";radio
1700 PRINT AT 7,13;"r : ";radio
1710 REM
*****
* CALCULOS *
*      *
*****
1720 LET area=PI*radio2
1730 PRINT AT 10,12;"AREA : "
;area
1740 LET perimetro=2*PI*radio
1750 PRINT AT 12,12;"PERIMETRO: "
;perimetro
1760 GO TO 9000
9000 REM
*****
* MAS CALCULOS? *
*      *
*****
9010 INPUT "Quiere continuar (S
C N) >>> ";LINE r$
9020 IF r$="n" OR r$="N" THEN ST
OP
9030 IF r$="s" OR r$="S" THEN CL
S : GO TO 40
9040 GO TO 9010

```

Programa

Como programa de repaso de las sentencias «GO TO» e «IF ... THEN ...», se propone el programa número «2». Sálvelo, por ejemplo, de la forma:

SAVE "AREAS" LINE 10

Este programa calcula el *área* y el *perímetro* de las siguientes figuras geométricas.

- TRIANGULO.
- RECTANGULO.
- CUADRADO.
- ROMBO.
- ROMBOIDE.
- TRAPECIO.
- CIRCULO.

El programa, al autoejecutarse, presenta en pantalla un menú con las diversas opciones; seleccionando una de ellas, pasaremos a una pantalla en la que aparecerá dibujada la figura geométrica correspondiente, ésta es realizada con ayuda de

las sentencias «PLOT», «DRAW» o «CIRCLE». Una vez introducidos los datos (lado, altura,...) que nos pide el ordenador, los resultados correspondientes al «área» y al «perímetro» serán visualizados.

Para retornar al menú principal debe pulsar «S» o «s», pulsando «n» o «N» el programa se detendrá presentando el mensaje:

9 STOP statement, 90 20 :2

La estructura del programa es:

| | | | |
|---------|---|-----------|--|
| 10: | Comentario con el nombre del programa. | 440 | : Salto de la rutina «FIN». |
| 20: | Asignación de los colores de la pantalla. | 450-540 | : Dibujo del <i>rectángulo</i> . |
| 30-130 | : Menú de opciones. | 550-590 | : Entrada de datos, (base y altura). |
| 140 | : Entrada de «opción». | 600-640 | : Cálculo y visualización. |
| 150-240 | : Verificación y selección. | 650 | : Salto a la rutina «FIN». |
| 250-314 | : Dibujo de <i>triángulo</i> . | 700-780 | : Dibujo del <i>cuadrado</i> . |
| 320-360 | : Entrada de datos, (base y altura). | 790-900 | : Entrada de datos, (lado). |
| 370-430 | : Cálculo y visualización. | 910-950 | : Cálculo y visualización. |
| | | 960 | : Salto a la rutina «FIN». |
| | | 970-1040 | : Dibujo del <i>rombo</i> . |
| | | 1050-1090 | : Entrada de datos, (diagonal mayor y diagonal menor). |
| | | 1100-1150 | : Cálculo y visualización. |
| | | 1160 | : Salto a la rutina «FIN». |
| | | 1170-1265 | : Dibujo del <i>romboide</i> . |
| | | 1270-1314 | : Entrada de datos, (base, altura y lado). |
| | | 1320-1360 | : Cálculo y visualización. |
| | | 1370 | : Salto a la rutina «FIN». |
| | | 1380-1480 | : Dibujo del <i>trapecio</i> . |
| | | 1490-1550 | : Entrada de datos, (base mayor, base menor y altura). |
| | | 1560-1610 | : Cálculo y visualización. |
| | | 1620 | : Salto a la rutina «FIN». |
| | | 1630-1670 | : Dibujo del <i>círculo</i> . |
| | | 1680-1700 | : Entrada de datos, (radio). |
| | | 1710-1750 | : Cálculo y visualización. |
| | | 1760 | : Salto a la rutina «FIN». |
| | | 9000-9040 | : Rutina «FIN». ■ |

BUCLES

Al analizar un programa, sucede con bastante frecuencia, que deba repetirse un cálculo o realizar una misma tarea con distintos datos. Sería una manera ilógica, en principio, editar las instrucciones de cálculo tantas veces como datos tengamos, ya que se utilizaría más memoria del ordenador; una forma algo más lógica sería utilizar un *bucle* (loop en inglés), que repitiera las mismas instrucciones tantas veces como quisieramos.

En programación, la terminología inglesa de los tipos de bucle más utilizada es:

- DO WHILE
- REPEAT UNTIL
- FOR ... NEXT

DO WHILE

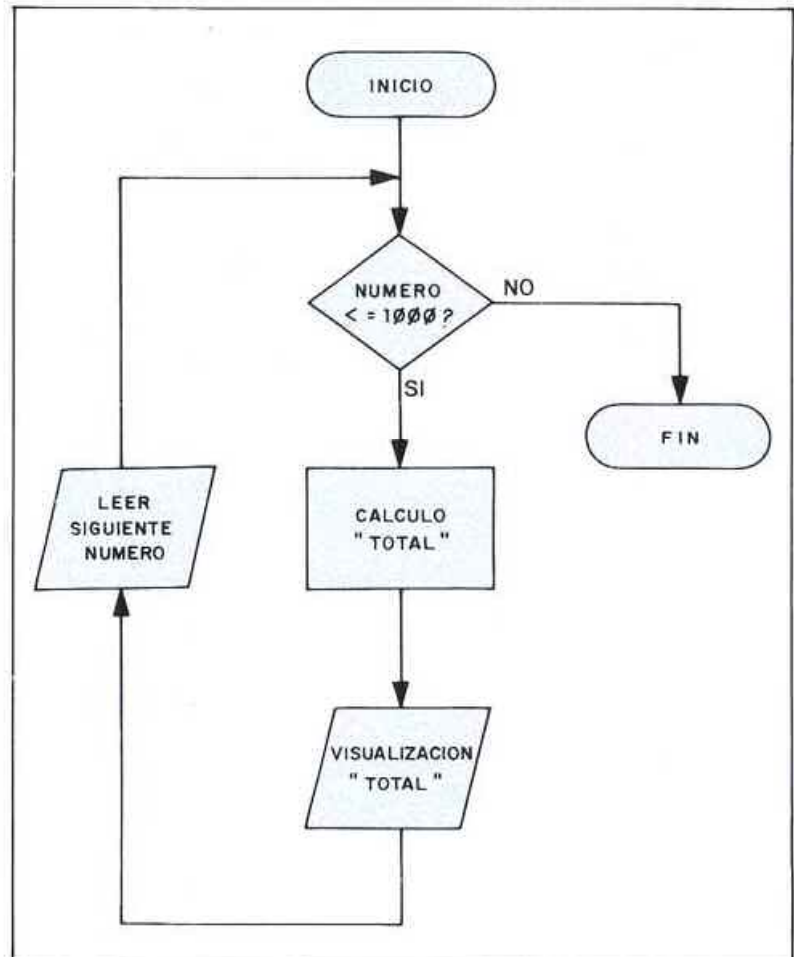
La estructura de bucle DO WHILE (Hacer mientras...) permite realizar una tarea varias veces, siempre y cuando se cumpla la condición impuesta.

Ejemplo:

```

10 REM *****
   DO WHILE *****
15 LET total=0
20 INPUT "Numero >>> "; numero
30 IF numero>1000 THEN PRINT "
   >>>> Final <<<<< STOP
40 LET total=total+numero
50 PRINT numero,total
60 GO TO 20
    
```

mientras se cumpla que la variable «número» sea igual o inferior a «1000», se realizarán indefinidamente las tareas de cálculo y visualización de la variable «total».



Estructura «DO WHILE».

REPEAT UNTIL

Con este tipo de bucle, (Repetir hasta..., es la traducción) el ordenador realiza una y otra vez la tarea hasta que se da una condición.

Ejemplo:

```

10 REM *****
   DO UNTIL *****
15 LET total=0
20 INPUT "Numero >>> "; numero
30 LET total=total+numero
40 PRINT numero,total
50 IF total>1000 THEN PRINT "
   >>>> FIN <<<<< STOP
60 GO TO 20
    
```

en este otro ejemplo, se repite el bucle hasta que se cumple la condición de que la variable «total» es superior a «1000».

Diferencias

La diferencia entre estos dos tipos de estructura estriba, en que en el primer caso (DO WHILE) la salida del bucle se encuentra antes de realizar la tarea, y en el segundo (REPEAT UNTIL) se encuentra al final, ¿Qué significado prác-

tico tiene esto? Que con una estructura DO WHILE, si al entrar en el bucle no se cumple la condición prevista, se sale de él sin haber ejecutado ninguna tarea; sin embargo, con la estructura REPEAT UNTIL, al menos *una vez* se ejecutan las instrucciones contenidas en él.

FOR/NEXT

Acceso al teclado

SGN



MODO **K**

IN KEY \$



MODO **K**

OVER

Tipo de sentencia

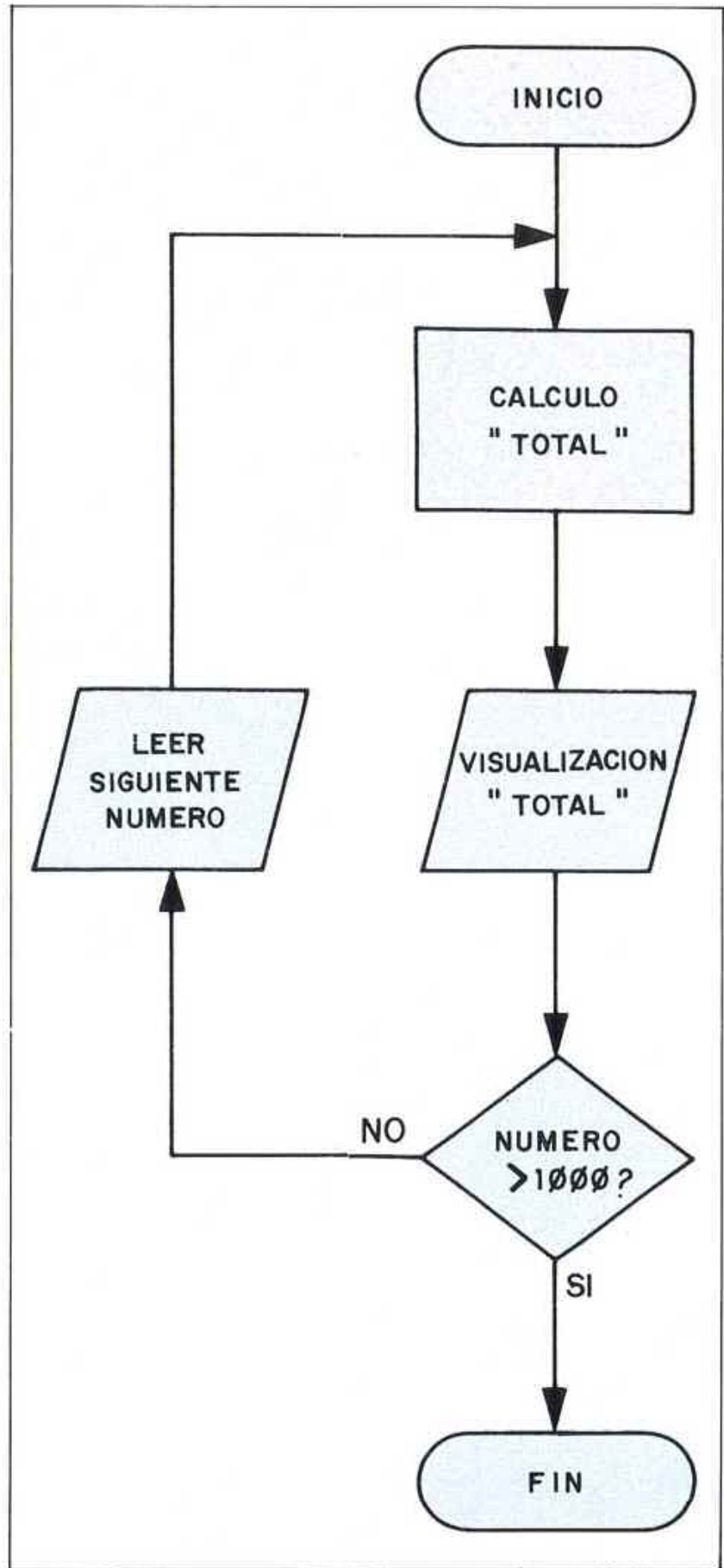
Comando de programación.

Definición

Los bucles «FOR/NEXT» permiten realizar las instrucciones contenidas en él tantas veces como se indica en los límites del argumento. El comienzo del bucle está definido por la sentencia «FOR», y el final por «NEXT». La estructura de la sentencia «FOR» es:

| SENTENCIA | ARGUMENTO |
|-----------|-------------------|
| FOR | var.= exp.TO exp. |

var. = variable.
exp. = expresión.
dentro del argumento se utili-



Estructura «REPEAT UNTIL».

za la palabra clave «TO», cuyo acceso al teclado es:



SGN



La estructura de «NEXT» tiene el siguiente formato:

| SENTENCIA | ARGUMENTO |
|-----------|-----------|
| NEXT | variable |

Ejemplos:

```

10 FOR j = 2 TO 10
_____
_____
_____
_____
50 NEXT j

```

En palabras sencillas, la explicación del ejemplo anterior es:

Ejecutar las instrucciones siguientes al «FOR ... TO ...» hasta que la variable «j», partiendo de «2», tome el valor «10»; la sentencia «NEXT» se encarga de incrementar el valor de esta variable; el bucle se repite por tanto, nueve veces.

La variable de control del bucle, sólo puede estar formada por una letra, si existe otra variable con el mismo nombre, ésta es borrada y asume el nuevo valor. La expresión anterior al «TO» es el valor inicial que debe tomar la variable de control, y la expresión posterior el valor final. A diferencia de otros lenguajes BASIC, la

sentencia «NEXT» debe incluir como argumento el nombre de la variable de control; por tanto *no puede omitirse*.

Ejemplo:

```

10 REM *****
      FOR...NEXT *****
20 FOR n=1 TO 61
30 PRINT "MICROHOBBY ";
40 NEXT n

```

En el programa anterior la cadena alfanumérica «MICROHOBBY» es visualizada 61 veces; se utiliza para formatear, el signo ortográfico «;».

La variable de control puede ser incluida en el grupo de sentencias que forman el bucle.

Ejemplo:

```

10 REM *****
      CUADRADOS *****
20 FOR c=100 TO 199
30 PRINT "Numero: ";c,
40 LET a=c^2
50 PRINT "Cuadrado: ";a
60 NEXT c

```

la variable de control «C» que varía entre 100 y 199 es elevada al cuadrado, asignando este valor a la variable «a»; ambas variables son visualizadas.

Los límites, al ser expresiones de tipo numérico, pueden estar constituidos por variables previamente asignadas.

Ejemplo:

```

10 REM *****
      PARAMETROS *****
20 INPUT "Limite inferior >>>";inferior

```

```

30 INPUT "Limite superior >>>";superior
40 IF superior<inferior THEN GOTO 20
50 FOR n=inferior TO superior
60 PRINT "inferior = ";inferior
70 PRINT "superior = ";superior
80 NEXT n

```

En este programa, los parámetros o límites son introducidos con sentencias «INPUT»; la línea 40 verifica que el valor asignado a la variable «superior» es mayor que el de la variable «inferior», en caso contrario, han de introducirse, de nuevo, los límites.

Dentro de un bucle se puede modificar el valor de la variable de control.

Ejemplo:

```

10 REM *****
      MODIFICAR 1 *****
20 FOR x=10 TO 40
30 LET x=x-2
40 PRINT "Valor de x: ";x
50 NEXT x

```

Como se puede observar, al ejecutar el programa, la variable «X» no alcanza los valores asignados en los límites, ya que en la línea 30 se decrece en dos su valor, ¿por qué al visualizar «X» se decrece sólo en una unidad?, la respuesta es sencilla, ya que la instrucción «LET X = X - 2» decrece dos unidades, pero la sentencia «NEXT X» incrementa en uno su valor, luego $-2 + 1 = 1$.

Cuando se modifican los límites en el interior de un bucle, el intérprete BASIC hace caso omiso de los nuevos valores y lo ejecuta con los iniciales.

Ejemplo:

```

10 REM *****
   : MODIF 2 :
   : *****
20 LET LIM INF=1
30 LET LIM SUP=20
40 FOR f=LIM INF TO LIM SUP
50 LET LIM INF=30
60 LET LIM SUP=90
70 PRINT "Control: ",f,"Limite
s: ",LIM INF," y ",LIM SUP
80 NEXT f
    
```

A pesar de las líneas 5φ y 6φ que modifican los límites, la variable «f» asume los valores iniciales 1 a 2φ.

Dentro de un bucle «FOR/NEXT» se puede incluir una instrucción del tipo «IF ... THEN ...» que si se cumple la condición, se produzca una ruptura en su ejecución, aunque la variable de control no haya alcanzado su valor final.

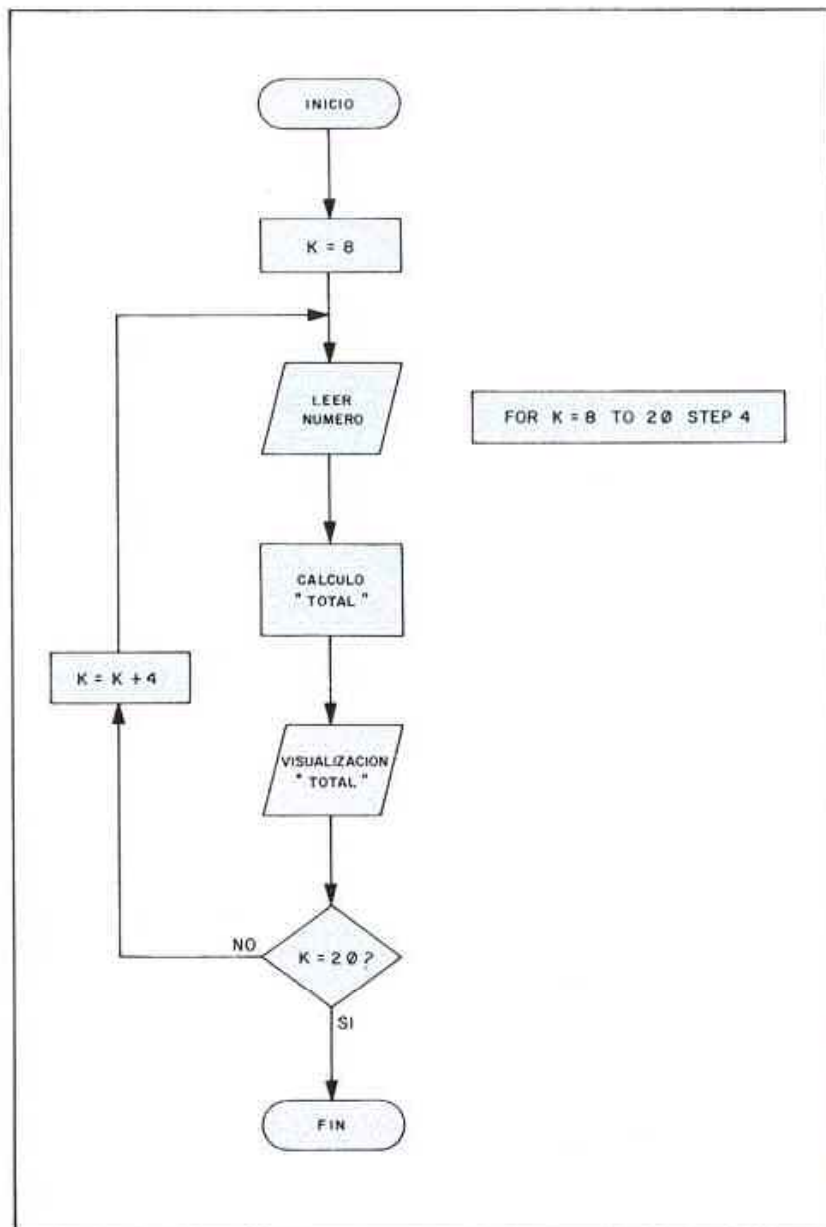
Este tipo de estructura de bucle es similar a la que, en otros lenguajes de más alto nivel, se denomina «DO WHILE/BREAK».

Ejemplo:

```

10 REM *****
   : BREAK :
   : *****
20 FOR w=1 TO 15
30 INPUT "Radio:"); r:radio
40 IF radio=0 THEN GO TO 80
50 LET area=PI*radiot2
60 PRINT "Radio: ",radio,"Area
: ",area
70 NEXT w
80 PRINT "***** F I
N *****"
    
```

si se introducen valores de radio distintos de «φ», el bucle se ejecuta un número de veces que coincide con los parámetros especificados (15 veces); si por el contrario, se introduce el código de ruptura, que en este caso es igual a «φ»; la ejecución pasa a la línea 1φ sin haber concluido el bucle.



Desarrollo de un bucle FOR-NEXT.

STEP

Acceso al teclado



+

DATA

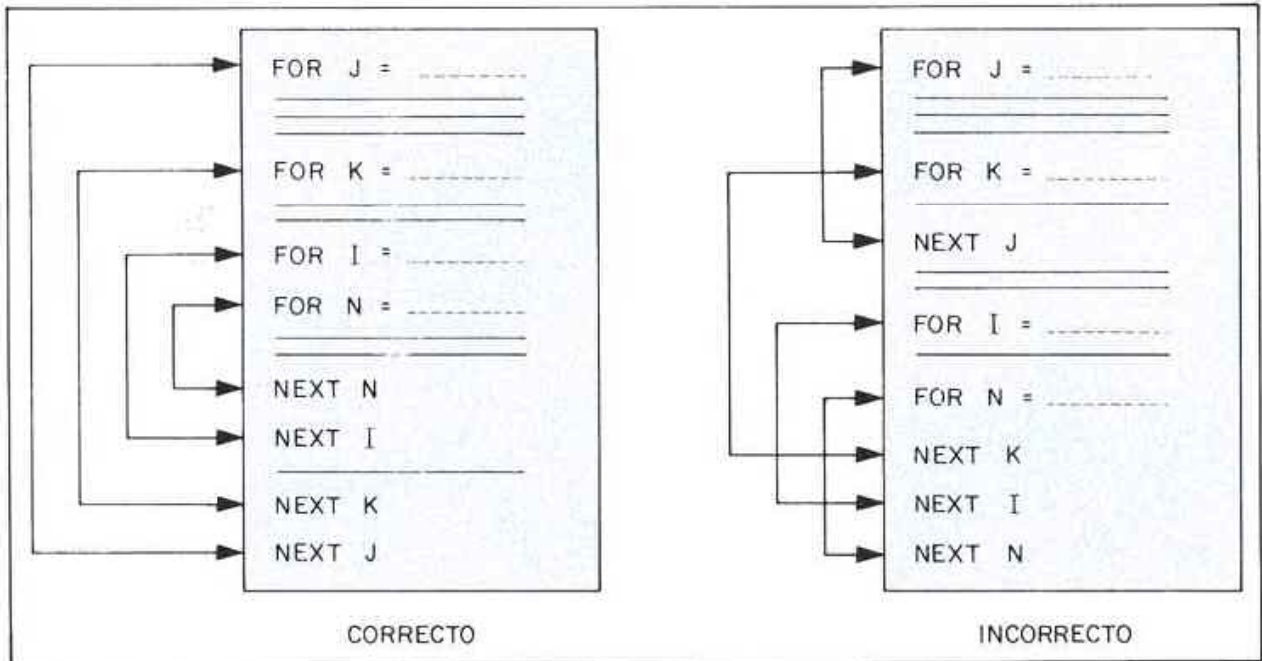


Tipo de sentencia

Comando de programación.

Definición

La palabra clave «STEP» se maneja conjuntamente con la sentencia «FOR ... TO...», formando parte de su argumento. Se utiliza para modificar el paso de un bucle, es decir, el incremento que se realiza a la variable de control cuando se ejecuta la sentencia «NEXT» correspondiente.



Anidamiento de bucles.

Cuando «STEP» se omite, el intérprete BASIC toma por defecto el valor «1», como se ha visto en los ejemplos anteriores, que la variable de control asumía inicialmente el valor del límite inferior y se incrementaba en uno hasta alcanzar el del límite superior.

Su estructura es la siguiente:

| SENTENCIA | ARGUMENTO |
|-----------|-----------------------|
| FOR | ... TO STEP expresión |

Ejemplo:

```
FOR j = 8 TO 32 STEP 4
```

la variable de control «j» tomará el valor «8» y se incrementará de cuatro en cuatro hasta alcanzar el valor «32», es decir, 8-12-16-20 ... 32. El bucle se realiza siete veces.

Ejemplo:

```
10 REM *****
   :          : STEP
   :          : *****
20 INPUT "Paso >>> " ; paso
30 FOR n=0 TO 100 STEP paso
40 PRINT "Variable de control: "; n
50 NEXT n
```

el número de veces que se ejecuta el programa anterior, así como los valores que toma la variable de control «n», depende del que se asigne a la variable «paso», introducida por el teclado.

El *paso* puede ser una expresión negativa, en este caso, el valor inicial debe ser superior al final.

Ejemplo:

```
10 REM *****
   :          : NEGATIVO
   :          : *****
12 LET bucle=110
14 LET error=80
20 INPUT "Valor inicial >>> " ; i
30 INPUT "Valor final >>> " ; f
   :          :
40 IF i<f THEN LET line
   :          : a=20 GO TO error
50 INPUT "Paso >>> " ; paso
60 IF paso>0 THEN LET linea=5
   :          : GO TO error
70 GO TO bucle
80 PRINT "0-"; i; "1-"; f; "2-"; error
   :          : *****
90 PAUSE 50
100 GO TO linea

110 FOR c=i TO f STEP pa
120 PRINT "Control "; c
130 NEXT c
```

este programa sólo se ejecuta con bucles decrecientes, ya que presenta un mensaje de

ERROR si el valor inicial es inferior al final o si el paso es mayor o igual a cero.

Budes anidados

Se denomina bucle *anidado*, aquel que contiene otro en su interior; no existe límite en cuanto a la cantidad de bucles que se pueden anidar.

Los bucles deben estar completamente *encajados*, por lo que no deben solaparse.

Ejemplo:

```
10 REM *****
   :          : ANIDADO
   :          : *****
20 FOR a=1 TO 5
30 FOR b=1 TO 10
40 PRINT "Bucle 1 "; a
50 PRINT "Bucle 2 "; b
60 NEXT b
70 PRINT "*****"
80 NEXT a
```

El bucle interior «b» se ejecuta 50 veces aunque sus límites van del 1 al 10, ya se multiplica por los del exterior «a». El programa visualiza los valores de las dos variables de

control, la variable «a» se incrementa en 1 cada vez que «b» completa su ciclo.

El siguiente programa calcula las potencias segunda, tercera, cuarta y quinta correspondientes a los veinte primeros números:

```

10 REM *****
      POTENCIACION
*****
20 FOR q=1 TO 20
25 LET s110=0
30 FOR m=2 TO 5
40 LET potencia=q**m
42 IF s110=0 THEN PRINT TAB 0
  "TAB 3:potencia; GO TO 60
50 PRINT TAB s110:potencia;
60 LET s110=s110+B
70 NEXT m
80 NEXT q
  
```

en la primera columna, aparecen los números del 1 al 20 y en las siguientes, por orden, la correspondiente potenciación.

Errores

Hay una serie de mensajes de error relacionados con los bucles «FOR ... NEXT ...»:

- Sentencia «NEXT» sin «FOR».

El mensaje:

1 NEXT without FOR

aparece cuando el ordenador encuentra una sentencia «NEXT» sin haber ejecutado con anterioridad su sentencia «FOR» correspondiente y, además, existe definida una variable con el mismo nombre que el argumento de «NEXT».

Ejemplo:

```

10 REM *****
      ERROR 1
*****
20 LET J=30
30 LET S=30
40 PRINT S*J
50 NEXT J
  
```

- Variable no encontrada.

2 Variable not found

similar al error anterior, pero sin estar definida ninguna variable con el mismo nombre.

Ejemplo:

```

10 REM *****
      ERROR 2
*****
30 LET S=30
40 PRINT S*10
50 NEXT J
  
```

- Sentencia «FOR» sin «NEXT».

1 FOR with out NEXT

Este mensaje lo presenta el ordenador cuando se encuentra con una sentencia «FOR» en la que los límites o el paso están incorrectos y además no encuentra la sentencia «NEXT» correspondiente.

Ejemplo:

```

10 REM *****
      ERROR 3
*****
20 FOR J=4 TO 2
30 LET S=30
40 PRINT S*J
  
```

- STEP 0.

Cuando por error se edita una sentencia «FOR» con paso 0, la variable de control no se incrementa al ejecutarse la sentencia «NEXT» correspondiente.

Ejemplo:

```

10 REM *****
      ERROR 4
*****
  
```

```

20 FOR J=10 TO 200 STEP 0
30 LET S=J+S
40 PRINT "Variable: ",J,"Resultado: ",S
50 NEXT J
  
```

a pesar de estar comprendidos los límites entre 10 y 200, la variable «j» asume el valor «10» una y otra vez y, por tanto, no alcanza el valor final.

- Cuando por error, se omite la palabra clave «STEP» en un bucle decreciente, éste no se ejecuta y, por tanto, continúa en la instrucción siguiente al «NEXT». El ordenador no presenta en este caso mensaje de error.

Ejemplo:

```

10 REM *****
      ERROR 5
*****
20 FOR I=200 TO 10
30 LET S="Bucle"
40 PRINT "Variable: ",I,"Resultado: ",S
50 NEXT I
60 PRINT "***** FIN *****"
  
```

Programas

Los dos programas que se muestran a continuación, son aplicaciones de bucles «FOR ... NEXT ...».

El programa número «1» es de utilidad en matemáticas. Según se introducen los datos, se calcula su suma aritmética, su media aritmética y la suma de sus cuadrados.

La estructura es la siguiente:

- 10 : Comentario con el nombre el programa.
- 20 : Asignación de los colores de pantalla.
- 30-50 : Entrada del número de datos a calcular,


```

*****
*          *
*  CALCULO  *
*          *
*****

210 LET suma=suma+dato
220 LET cuadrados=cuadrados+dat
0↑2
230 LET media=suma/n
240 REM
*****
*          *
*  BORRADO  *
*          *
*****

250 FOR a=7 TO 13 STEP 2
260 PRINT AT a,19;"
270 NEXT a
280 GO TO 120

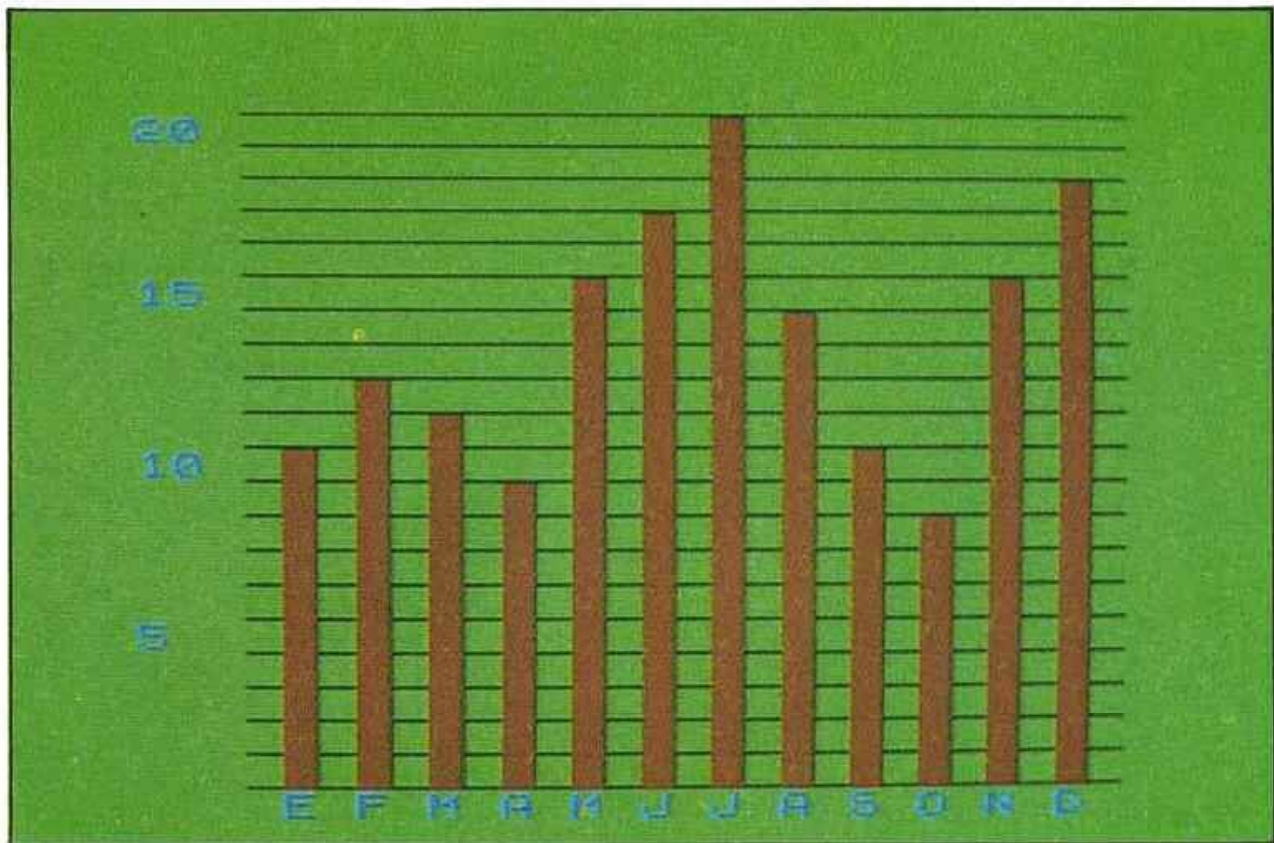
```

verificación de los mismos, tienen que estar comprendidos entre «0» y «20» (millones), visualización y salto a la rutina de almacenamiento.

- 252-256 : Pausa y borrado de la pantalla.
- 260 : Salto a la rutina de dibujo.
- 262-500 : Almacenamiento de datos y retorno al bucle principal.
- 510-560 : Bucle anidado

para dibujar las rayas horizontales del «histograma» con el símbolo del subrayado «_».

- 570-620 : Visualización de los valores en millones en el eje vertical (tinta azul).
- 630-680 : Visualización de las iniciales de los meses en el eje horizontal (tinta azul).
- 690-790 : Bucle para el salto a la rutina de extracción de datos de memoria y su representación en gráfico de barras (tinta roja).
- 800 : Salto de la rutina «fin».
- 810-1050 : Rutina de extracción de datos y

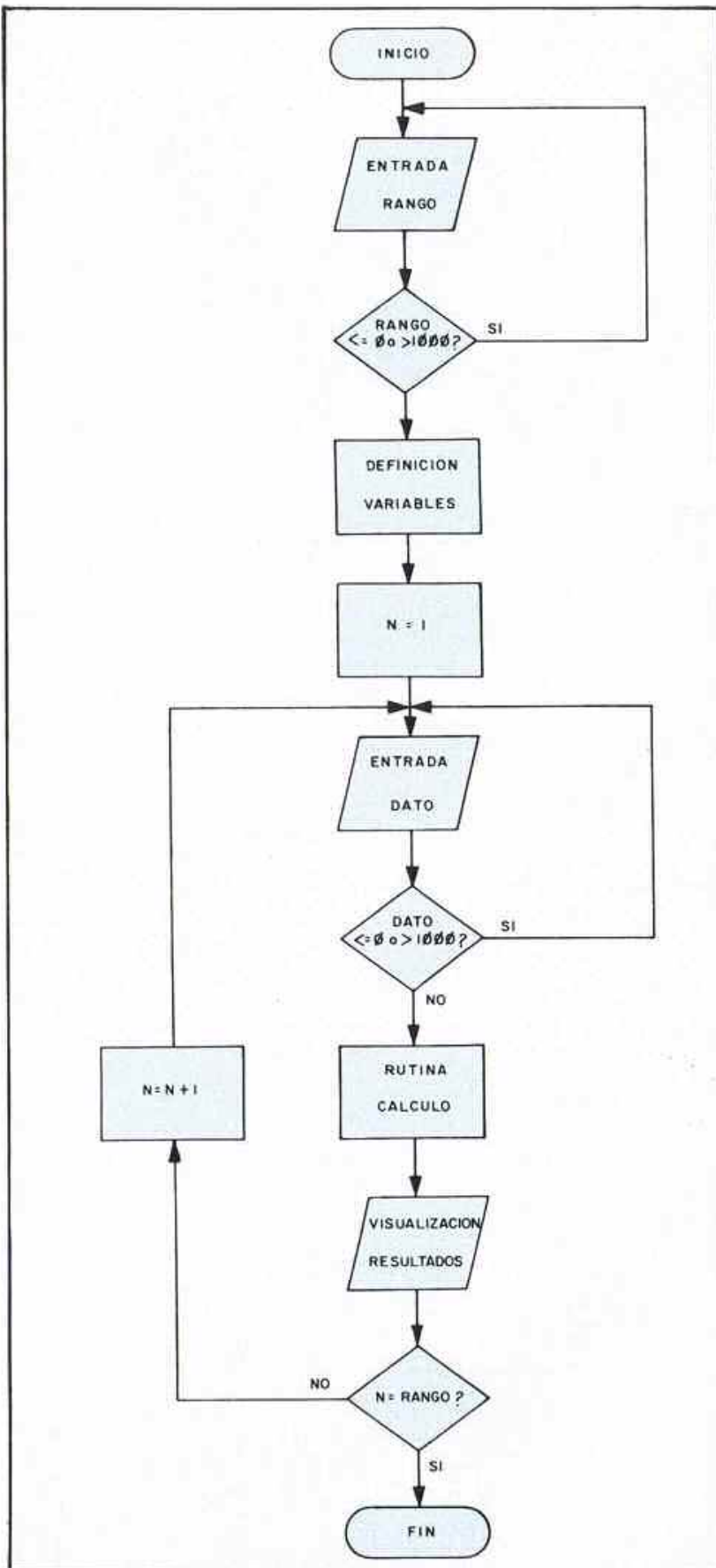


PROGRAMA 2

```

10 REM *****
* HISTOGRAMAS *
*****
20 BORDER 4: PAPER 4: INK 0: C
LS
22 LET memoria=270
30 REM *****
* CARATULA *
*****
40 PRINT AT 0,7;"Entrada de da
tos"
50 PRINT AT 1,7;"_____
60 PRINT AT 6,0;"Enero .....
70 PRINT "Febrero ....."
80 PRINT "Marzo ....."
90 PRINT "Abril ....."
100 PRINT "Mayo ....."
110 PRINT "Junio ....."
120 PRINT "Julio ....."
130 PRINT "Agosto ....."
140 PRINT "Septiembre .."
150 PRINT "Octubre ....."
160 PRINT "Noviembre ....."
170 PRINT "Diciembre ....."
180 REM *****
* ENTRADA DATOS *
*****
190 FOR n=1 TO 12
200 INPUT "Datos mes ";(n);" >>
" ,dato
210 IF dato<0 OR dato>20 THEN G
O TO 200
220 PRINT AT 5+n,15;dato;AT 5+n
,18;"millones"
230 GO TO memoria
240 LET memoria=memoria+20
250 NEXT n
252 PRINT #0;"Pulse una tecla p
ara continuar"
254 PAUSE 0
256 CLS
260 GO TO 510.
262 REM *****
* MEMORIA *
*****
270 LET enero=dato
280 GO TO 240
290 LET febrero=dato
300 GO TO 240
310 LET marzo=dato
320 GO TO 240
330 LET abril=dato
340 GO TO 240
350 LET mayo=dato
360 GO TO 240
370 LET junio=dato
380 GO TO 240
390 LET julio=dato
400 GO TO 240
410 LET agosto=dato
420 GO TO 240
430 LET septiembre=dato
440 GO TO 240
450 LET octubre=dato
460 GO TO 240
470 LET noviembre=dato
480 GO TO 240
490 LET diciembre=dato
500 GO TO 240
510 REM *****
* HORIZONTALES *
*****
515 PRINT #0;AT 1,3;"Espere un
momento, por favor."
520 FOR n=0 TO 20
530 FOR x=4 TO 28
540 PRINT AT n,x;"_"
550 NEXT x
560 NEXT n
570 REM *****
* VALORES *
*****
580 LET paso=20
590 FOR j=1 TO 16 STEP 5
600 PRINT INK 1;AT j,1;paso
610 LET paso=paso-5
620 NEXT j
630 LET a$="EFMAMJJASOND"
640 LET mes=1
650 FOR j=5 TO 27 STEP 2
660 PRINT INK 1;AT 21,j;a$(mes)
670 LET mes=mes+1
680 NEXT j
690 REM *****
* DIBUJO *
*****
700 LET memoria=820
710 FOR j=5 TO 27 STEP 2
720 GO TO memoria
730 IF valor=0 THEN GO TO 780
740 LET valor=21-valor
750 FOR t=20 TO valor STEP -1
760 PRINT INK 2;AT t,j;"█"
770 NEXT t
780 LET memoria=memoria+20
790 NEXT j
800 GO TO 1060
810 REM *****
* MEMORIA *
*****
820 LET valor=enero
830 GO TO 730
840 LET valor=febrero
850 GO TO 730
860 LET valor=marzo
870 GO TO 730
880 LET valor=abril
890 GO TO 730
900 LET valor=mayo
910 GO TO 730
920 LET valor=junio
930 GO TO 730
940 LET valor=julio
950 GO TO 730
960 LET valor=agosto
970 GO TO 730
980 LET valor=septiembre
990 GO TO 730
1000 LET valor=octubre
1010 GO TO 730
1020 LET valor=noviembre
1030 GO TO 730
1040 LET valor=diciembre
1050 GO TO 730
1060 REM *****
* FINAL *
*****
1070 INPUT "Otros datos? (S/N) >
" ; LINE a$
1080 IF a$="s" OR a$="S" THEN GO
TO 10
1090 IF a$="n" OR a$="N" THEN ST
OP
1100 GO TO 1070 =

```

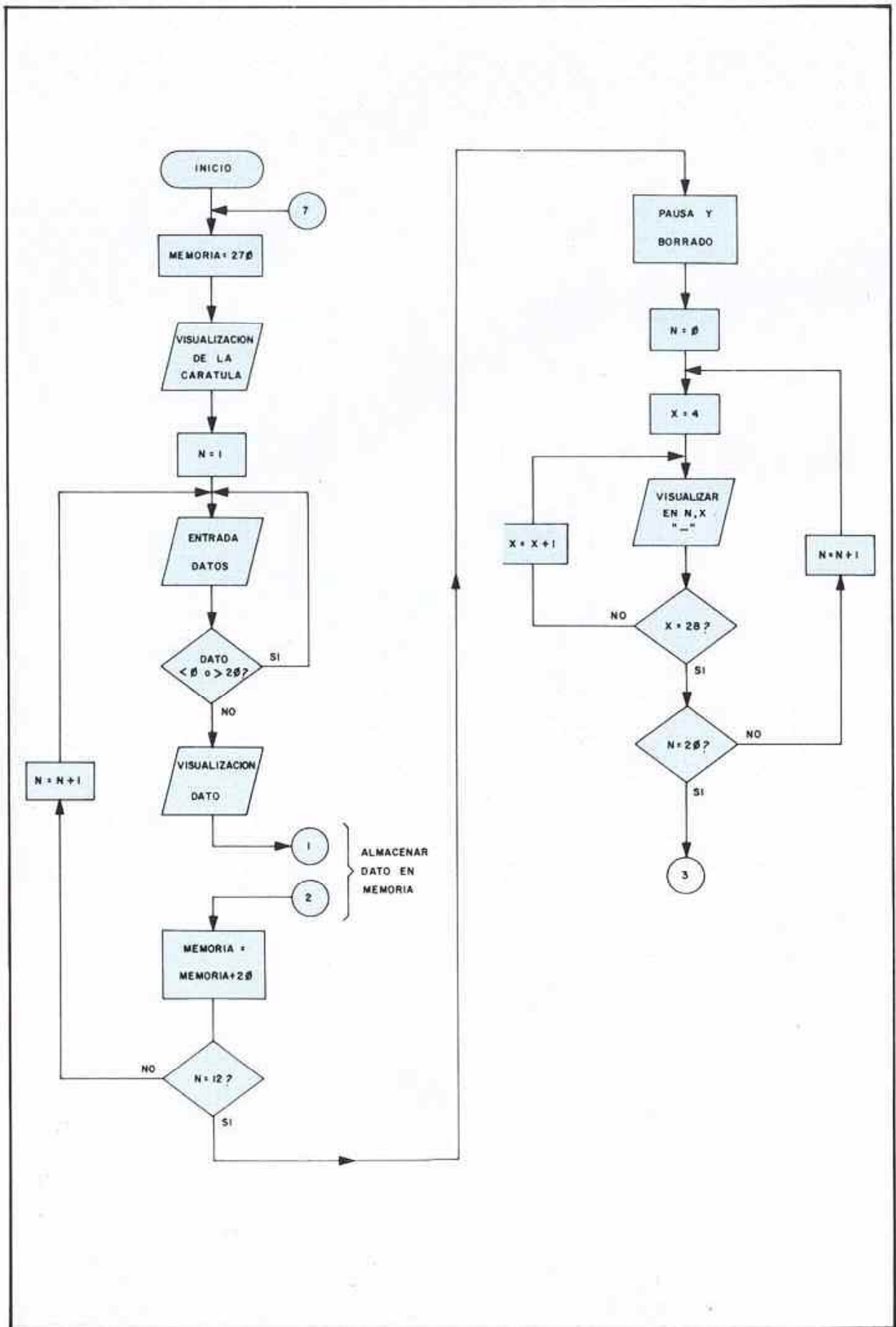


Programa «ESTADISTICA».

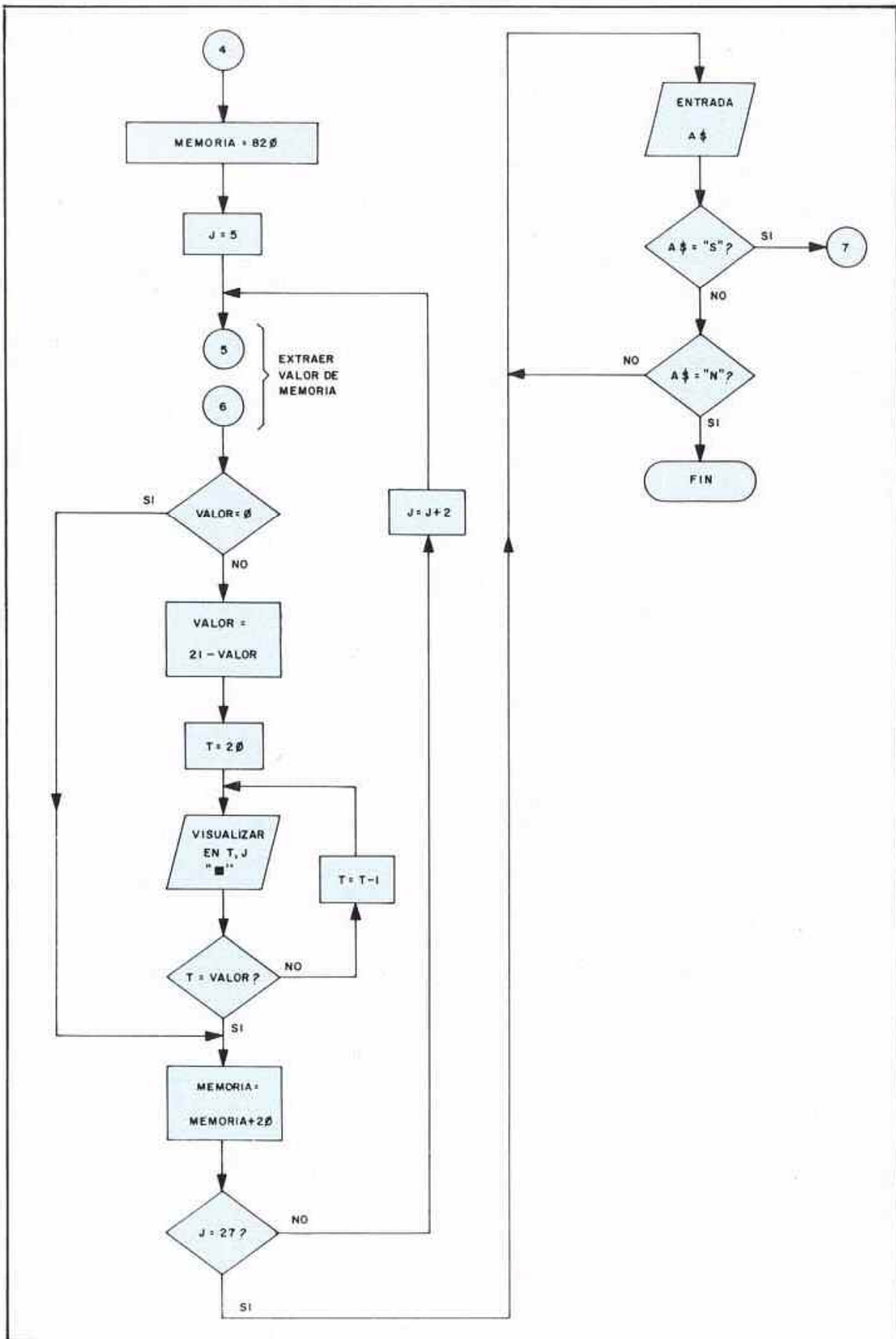
retorno al bucle principal.

1060-1100 : Decisión, de retorno al principio o terminar el programa.

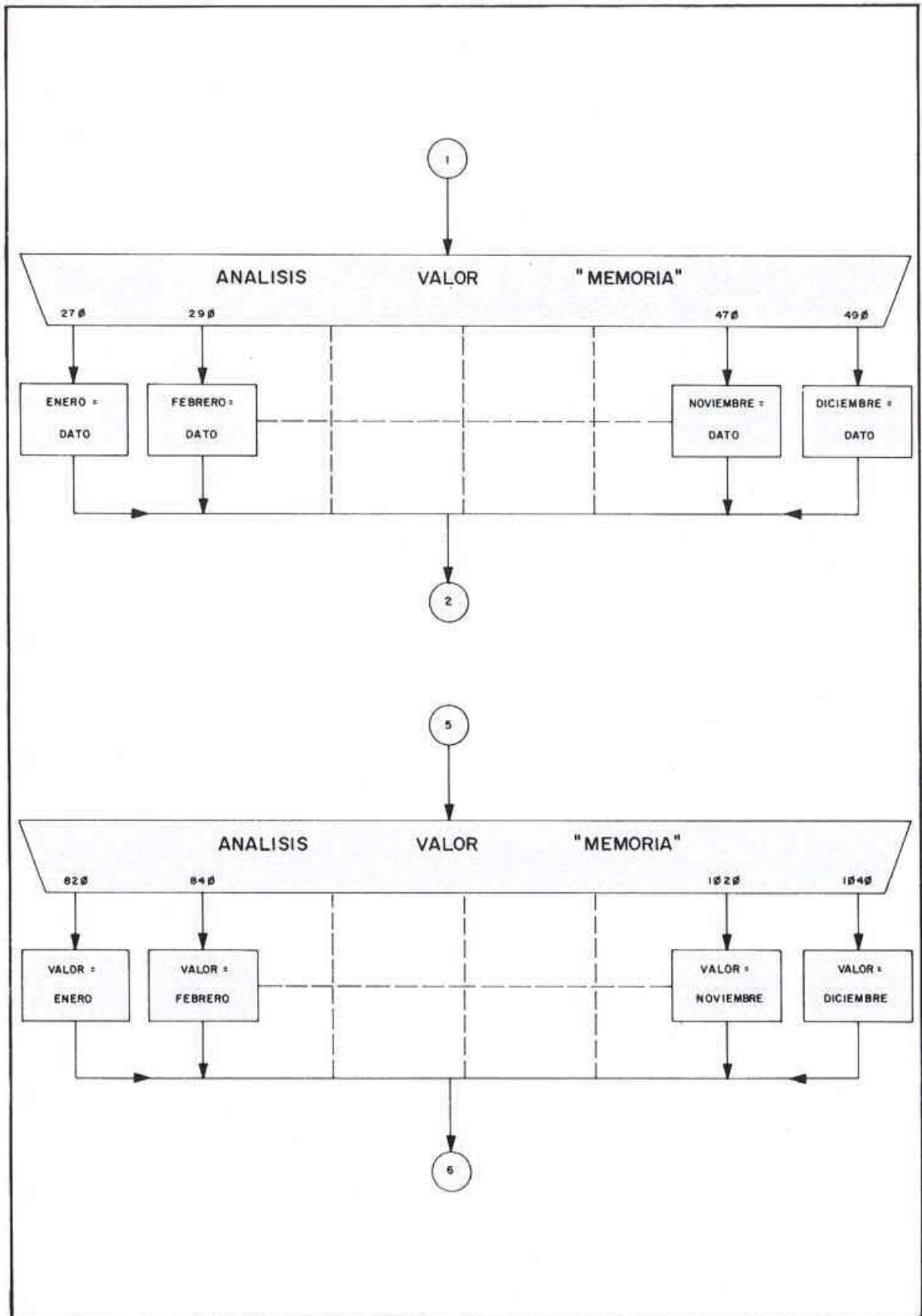
La rutina de almacenamiento y extracción de datos, se ha realizado de una manera un tanto especial, ya que todavía no se han visto las *matrices*, que serán estudiadas en un capítulo posterior. ■



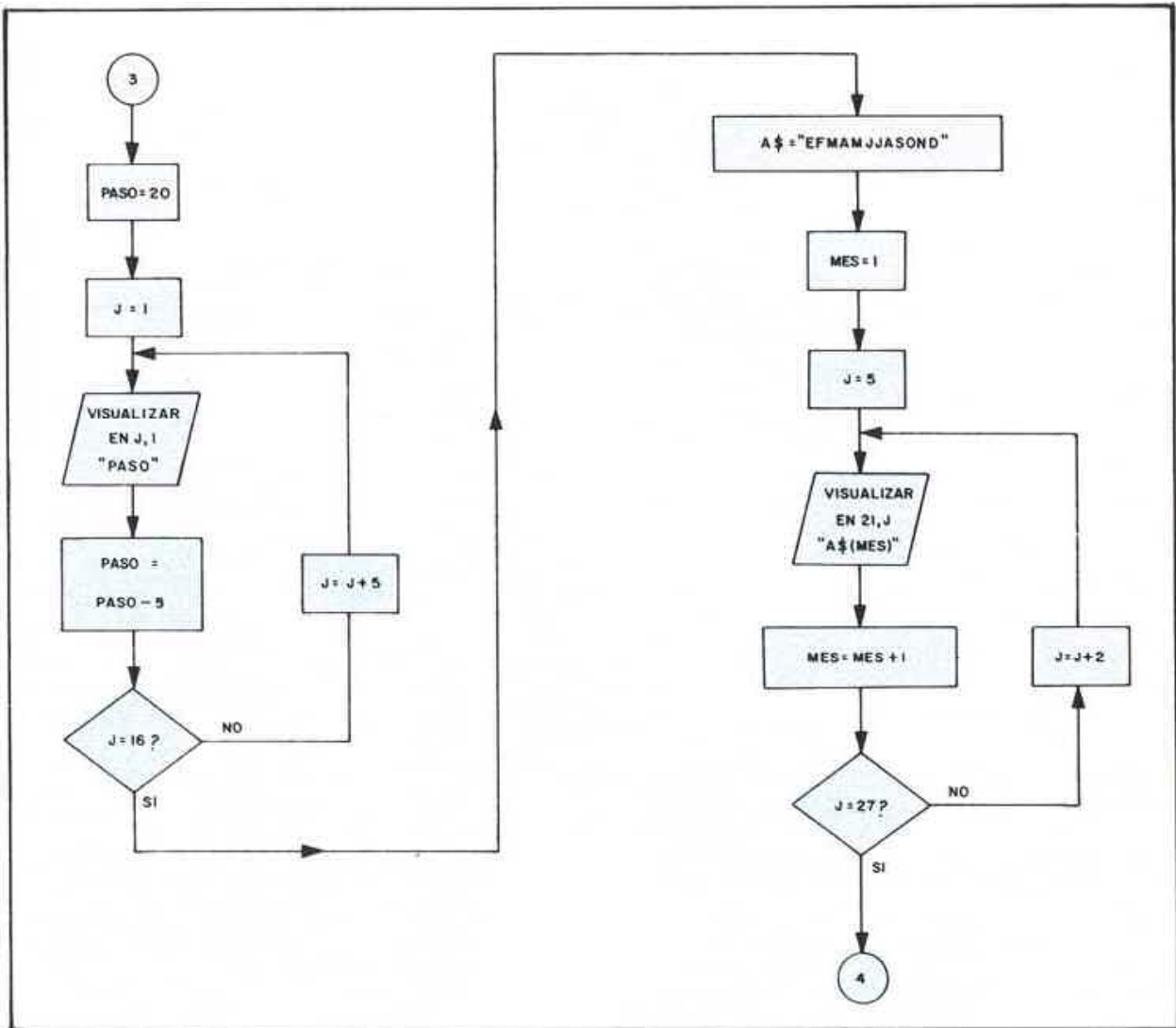
Programa «HISTOGRAMA» entrada de datos y dibujo líneas horizontales.



Programa «HISTOGRAMA» dibujo, barras y rutina «FIN».



Programa «HISTOGRAMA» almacenamiento y extracción de datos.



Programa «Histograma» datos eje vertical y horizontal.

SUBROUTINAS

Normalmente, en un programa hay ciertos cálculos o funciones que se necesitan en *distintas* partes del mismo; en lugar de editarlas varias veces, conviene hacerlo una sola vez en formato *subrutina* o *sub-programa*, de esta manera, este grupo de instrucciones sólo se ejecutan cuando el programa principal lo indica, mediante una instrucción de llamada a subrutina; cuando ésta termina de ejecutarse devuelve el control al programa principal.

Un bucle también repite varias veces una serie de instrucciones, pero siempre en una misma zona de programa; a diferencia, la subrutina puede ser llamada desde cualquier parte.

Dentro de la programación *estructurada*, la utilización de subrutinas es un hábito muy recomendable. Conviene para una mayor estructuración que todas las subrutinas estén localizadas en la zona final del programa; una a continuación de la otra.

GO SUB

Acceso al teclado

SQR



MODULO K

CIRCLE
122 MICROBASIC

Tipo de sentencia

Comando de programación.

Definición

Es la instrucción utilizada, dentro de un programa, para «llamar» a una subrutina, aunque también admite el formato de comando directo.

Su estructura general es:

| SENTENCIA | ARGUMENTO |
|-----------|--------------|
| GO SUB | N.º de línea |

Ejemplos:

- GO SUB 500.
- GO SUB 30.

RETURN

Acceso al teclado

STR \$



MODULO K

Tipo de sentencia

Comando de programación.

Definición

Esta palabra clave se utiliza para retornar, de una subrutina, al programa principal.

Su estructura es la siguiente:

| SENTENCIA | ARGUMENTO |
|-----------|-----------|
| RETURN | _____ |

Utilización de «GO SUB» y «RETURN»

La llamada a subrutina «GO SUB» puede hacerse en cualquier parte del programa que se necesita. La palabra clave «RETURN», sin embargo, debe utilizarse *siempre* al *final* de cada subrutina.

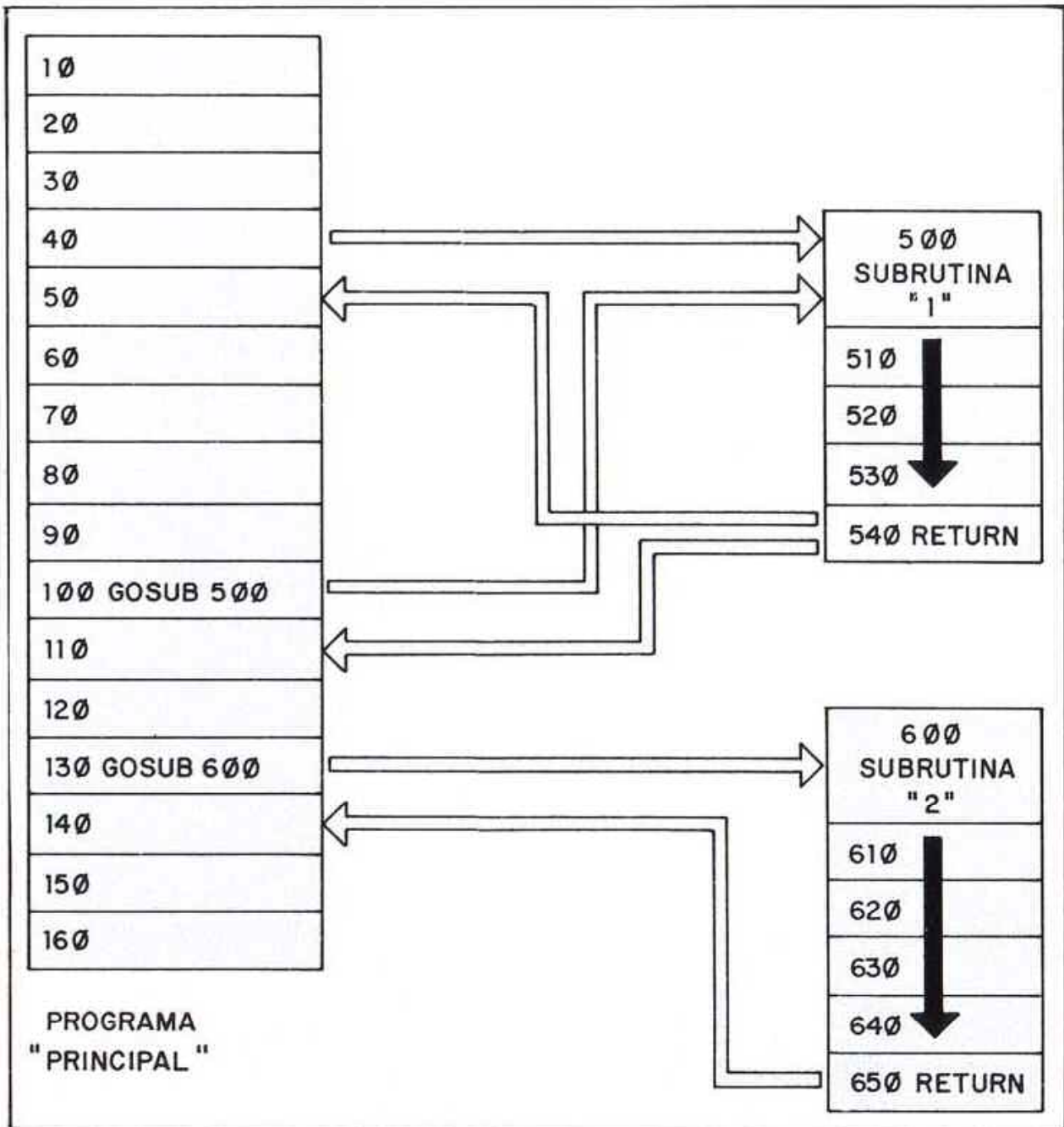
Ejemplo:

```

10 REM .....
      GO SUB .....
20 INPUT "A?" : a
30 INPUT "B?" : b
40 GO SUB 100
50 PRINT "r"
60 STOP
100 REM .....
      SUBROUTINA COMPARA .....
110 IF a=b THEN LET r$="igual"
GO TO 130
120 LET r$="distinto"
130 RETURN
    
```

Cuando se ejecuta la línea «40» el programa principal cede el control a la subrutina localizada en la línea «100», ésta se va ejecutando hasta encontrar la sentencia «RETURN» que causa el retorno al programa principal (línea 50).

A primera vista, parecen similares las sentencias «GO TO» y «GO SUB», ya que ambas provocan un salto al número de línea especificado en



Llamadas a «subrutinas».

su argumento, la diferencia radica en que, el ordenador, cuando se ejecuta la sentencia «GO SUB», apunta en una zona de la memoria denominada STACK o «Pila de GO SUB», memoria tipo «LIFO» (Last Input First Output) en la que el último dato almacenado es el primero en ser recuperado, la *dirección de retorno*, formada por el número de línea y sentencia dentro de la línea de la

instrucción «GO SUB». Cuando se ejecuta la última línea de la subrutina (RETURN), se recupera la dirección de retorno almacenada en la memoria, y el programa continúa en la instrucción siguiente. Como se puede observar, el usuario no tiene que preocuparse para nada de la dirección de retorno de una subrutina, de esta manera su manejo se hace sencillo.

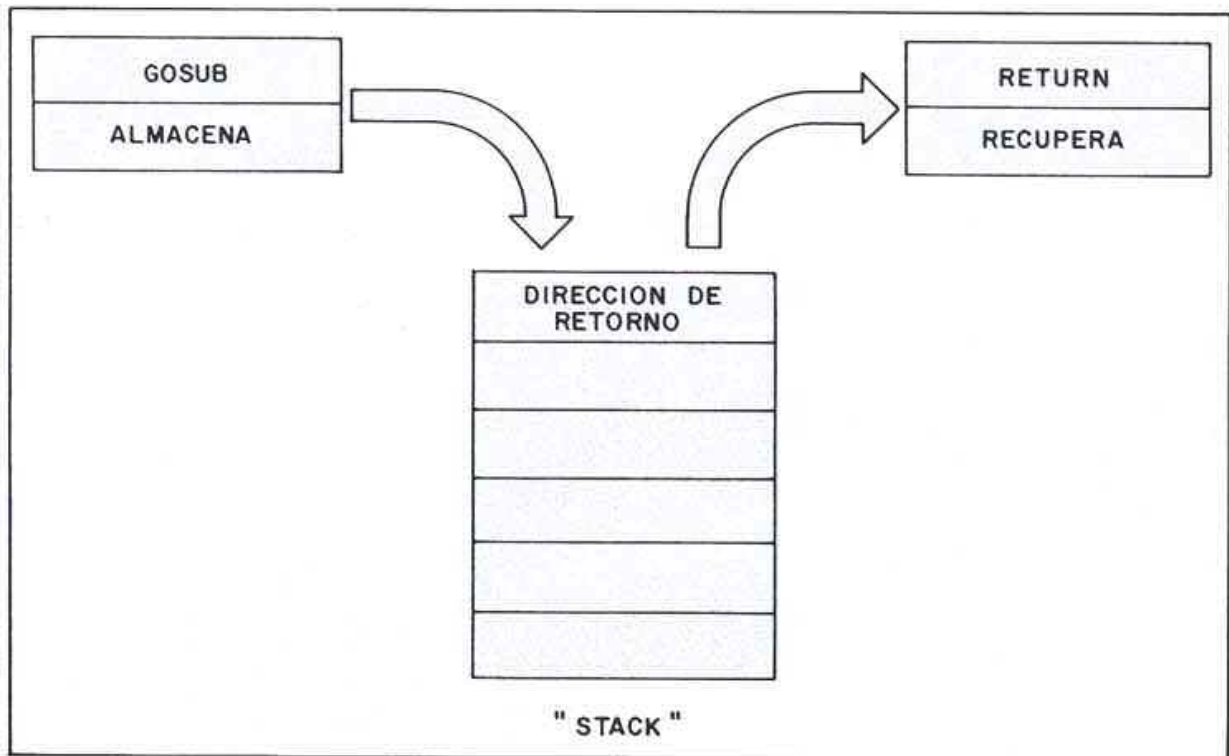
Las llamadas a subrutina pueden hacerse con variables numéricas, cuyo valor sea el número de línea donde están localizadas.

Ejemplo:

```

10 REM *****
   REM   VARIABLES
   REM *****
20 LET entrada=90
30 LET visualizacion=120

```



Pila de «Gosub».

```

40 PRINT "Subrutina 1"
50 GO SUB entrada
60 PRINT "Subrutina 2"
70 GO SUB visualizacion
75 STOP
80 REM
*****
SUBROUTINAS
*****
90 INPUT "Nombre >>> ", LINE n
$
100 INPUT "Apellidos >>> ", LIN
E $
110 RETURN
120 PRINT "Tu nombre es " $ n $ .
130 RETURN

```

Tipos de subrutinas

Las subrutinas pueden clasificarse según la filosofía de su ejecución en:

- Subrutinas sin parámetros.
- Subrutinas con parámetros.

Una subrutina sin parámetros es aquella que realiza un cálculo o función siempre de la misma manera, es decir, con los mismos datos o valores.

Ejemplo:

```

10 REM *****
   SIN PARAMETROS
*****

20 GO SUB 100
30 PRINT AT 10,10,"MICROHOBBY"
40 STOP
100 REM
*****
   SUBROUTINA
*****

105 CLS
110 FOR a=0 TO 21
120 PRINT PAPER a,AT a,0;"
130 NEXT a
140 RETURN

```

La subrutina localizada en la línea «100», borra y posteriormente colorea de verde la pantalla, siempre que es llamada.

Las subrutinas con parámetros se ejecutan de forma más generalizada, ya que permiten realizar una misma tarea para diversos valores.

El siguiente programa es una variable del anterior, permite borrar y colorear la pantalla del color que se desee.

Antes de llamar a la subrutina debe asignarse, a la variable utilizada, el valor correspondiente al color.

```

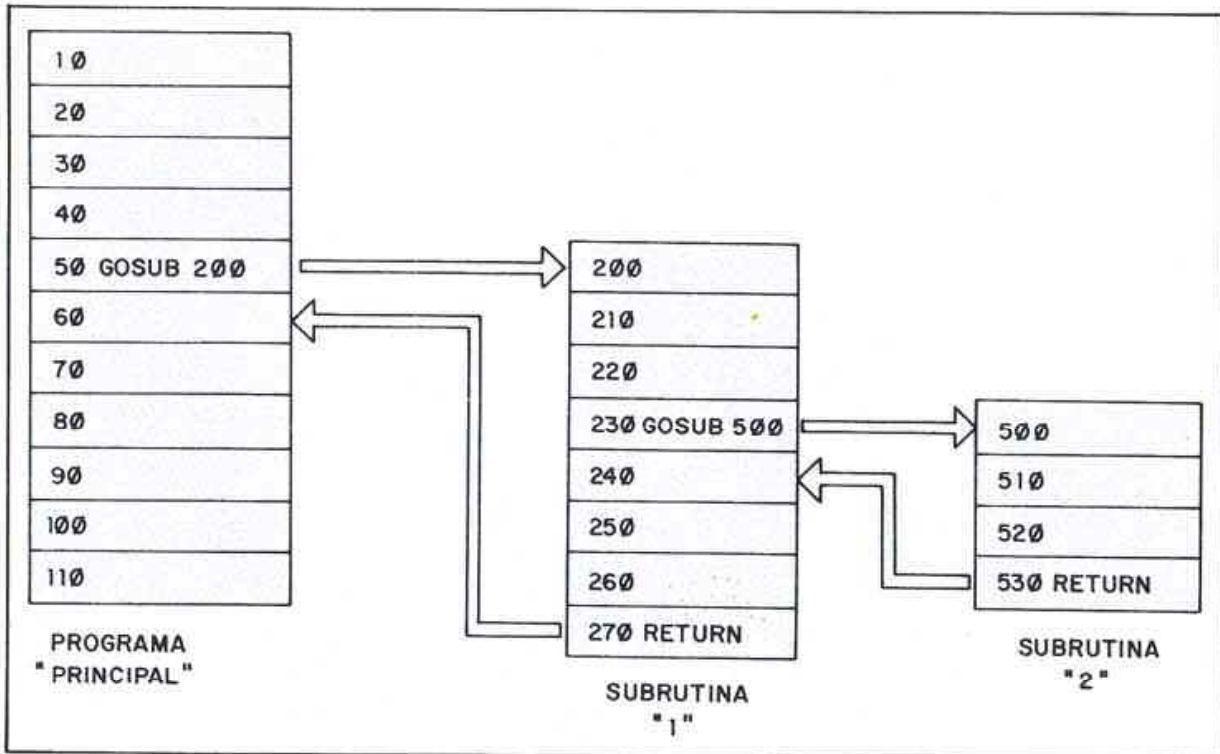
10 REM *****
   CON PARAMETROS
*****

15 LET color=2
20 GO SUB 100
30 PRINT AT 10,10,"MICROHOBBY"
35 GO SUB 200
40 LET color=4
50 GO SUB 100
70 PRINT AT 10,11,"SEMANAL"
75 GO SUB 200
80 LET color=5
85 GO SUB 100
90 PRINT AT 8,10,"MICROHOBBY"
95 PRINT AT 10,11,"SEMANAL"
99 STOP
100 REM
*****
   SUBROUTINA
*****

105 CLS
110 FOR a=0 TO 21
120 PRINT PAPER color,AT a,0;"
130 NEXT a
140 RETURN
200 FOR n=1 TO 150
210 NEXT n
220 RETURN

```

Se combina con otra subrutina localizada en la línea «200», que permite ver los mensajes sucesivos, ya que su



«Subrutinas» anidadas.

única misión es la de *temporizar*.

Los parámetros pueden ser tanto de *entrada* como de *salida*, veamos un ejemplo en el que se combinan ambos.

```

10 REM *****
   ENTRADA/SALIDA
*****
20 INPUT "Precio >>> " entrada
30 INPUT "Impuesto de Lujo >>> " entrada2
40 PRINT "Precio ....."
50 PRINT "Impuesto de Lujo ....."
60 PRINT "Total ....."
70 GO SUB 100
80 PRINT salida
90 STOP
100 REM *****
   SUBROUTINA
*****
110 LET impuesto=entrada1*entra
da2/100
120 LET salida=entrada1+impuest
o
130 RETURN

```

Atendiendo al tipo de llamada, las subrutinas pueden clasificarse en:

- INCONDICIONALES.
- CONDICIONALES.

Incondicionales son aquellas que no necesitan de ninguna condición para que se ejecuten.

Ejemplo:

```
GO SUB 750
```

Una llamada a subrutina condicional, es aquella que necesita que se cumplan una o varias condiciones previas.

Ejemplo:

```
IF a$ = "PEREZ" THEN GO SUB 100
```

Subrutinas anidadas

Al igual que los bucles, las subrutinas pueden anidarse, es decir, que desde una subrutina se puede llamar a otra y así sucesivamente. Las direcciones de retorno se van almacenando, como ya se comentó anteriormente, en el STACK;

la primera dirección en recuperarse corresponde con la de la última subrutina llamada.

Aunque no es muy frecuente, una subrutina puede llamarse a sí misma, esto es lo que en programación se denomina *subrutina Recursiva*.

Un ejemplo de subrutinas anidadas es el siguiente:

```

10 REM *****
   ANIDADOS
*****
20 LET fondo=4
30 LET tinta=6
40 GO SUB 100
50 PRINT PAPER fondo, INK tinta
60 STOP
100 REM *****
   SUBROUTINAS
*****
110 FOR a=8 TO 12
120 FOR b=9 TO 22
130 PRINT PAPER fondo, AT a, b, "
140 IF a=8 OR a=11 THEN GO SUB
200 GO TO 150
150 NEXT b
160 NEXT a
170 RETURN
200 PRINT PAPER fondo, INK 7, AT
a, b, "
210 RETURN

```



```

15 LET subrutina=100
20 INPUT "Sumando 1 >>> " :a
30 INPUT "Sumando 2 >>> " :b
40 GO SUB subrutina
100 REM
*****
SUBROUTINA
*****
105 PRINT a + " + " + b, " = "
110 LET resultado=a+b
115 PRINT resultado
120 RETURN

```

Programas

Como aplicación a las «subrutinas», se presentan dos programas:

- ADIVINO.
- LONGITUD.

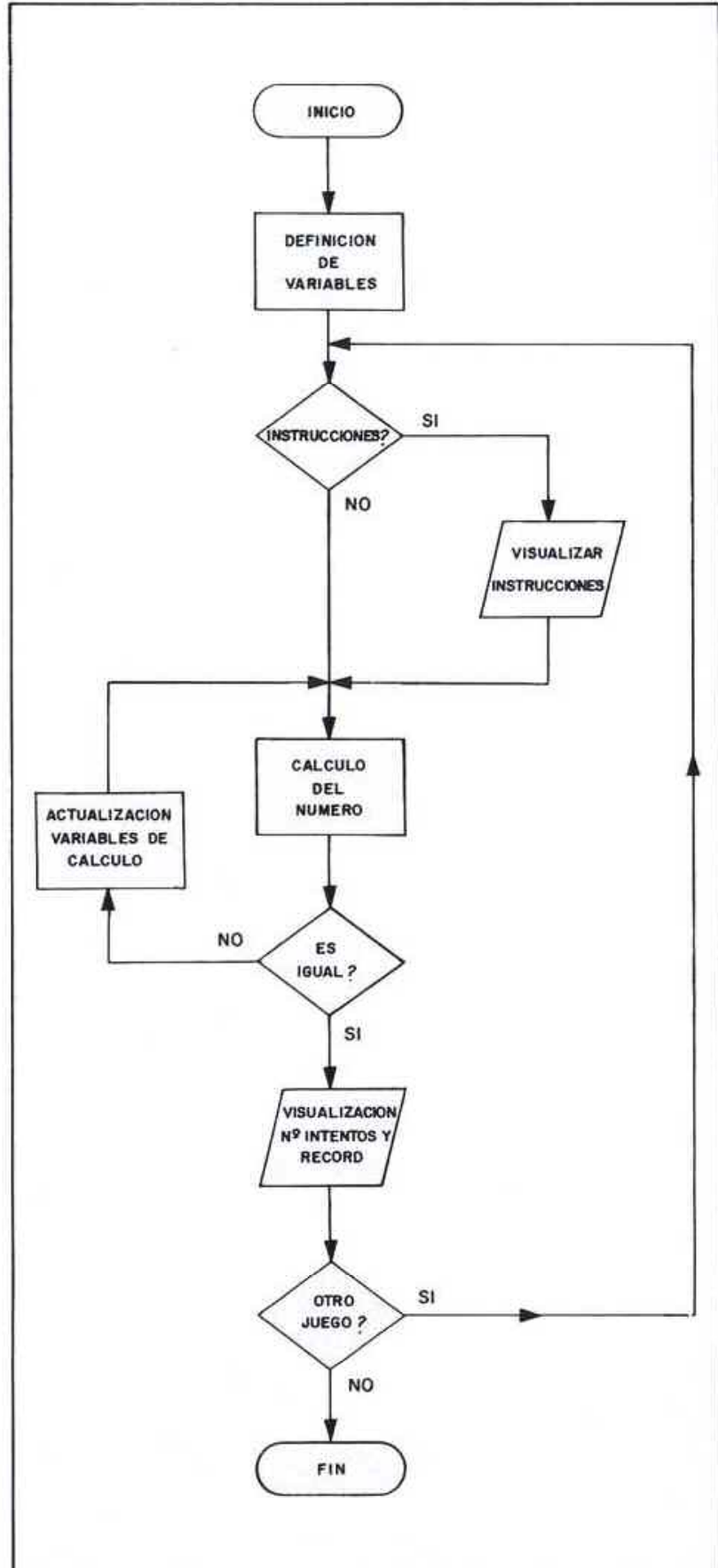
El primero se trata de un juego de *adivinanza*, en el que el ordenador debe descubrir el número pensado por usted, que deberá estar comprendido entre «0» y «100».

Al principio del programa, se da la opción de poder visualizar las instrucciones del juego. El ordenador, posteriormente, presenta un número de pantalla; según la relación que exista entre éste y el número pensado por nosotros, deberemos introducir una de las siguientes claves:

- «M» - Mayor.
- «N» - Menor.
- «I» - Igual.

La estructura general del programa es:

- 10 : Comentario con el nombre del programa.
- 12 : Asignación del color verde para el borde y el fondo, y azul para los caracteres.
- 14-80 : Definición de la variable «record», utilizada para almacenar el mínimo número de intentos; inicialmente tiene asignado



Estructura programa «Adivino».

el valor «99», para poder ser actualizado en la primera jugada.

También se definen, para una mejor interpretación, las variables utilizadas como dirección de comienzo de las subrutinas.

102-104 : Inicialización de las variables utilizadas en cada partida.

106-1505 : Presentación del programa y visualización de instrucciones, si se desea.

160-190 : Mensaje de invitación a comenzar el juego.

200-240 : Visualización del número calculado por el ordenador.

250 : Si el ordenador está seguro del número, se ejecuta la subrutina «acierto».

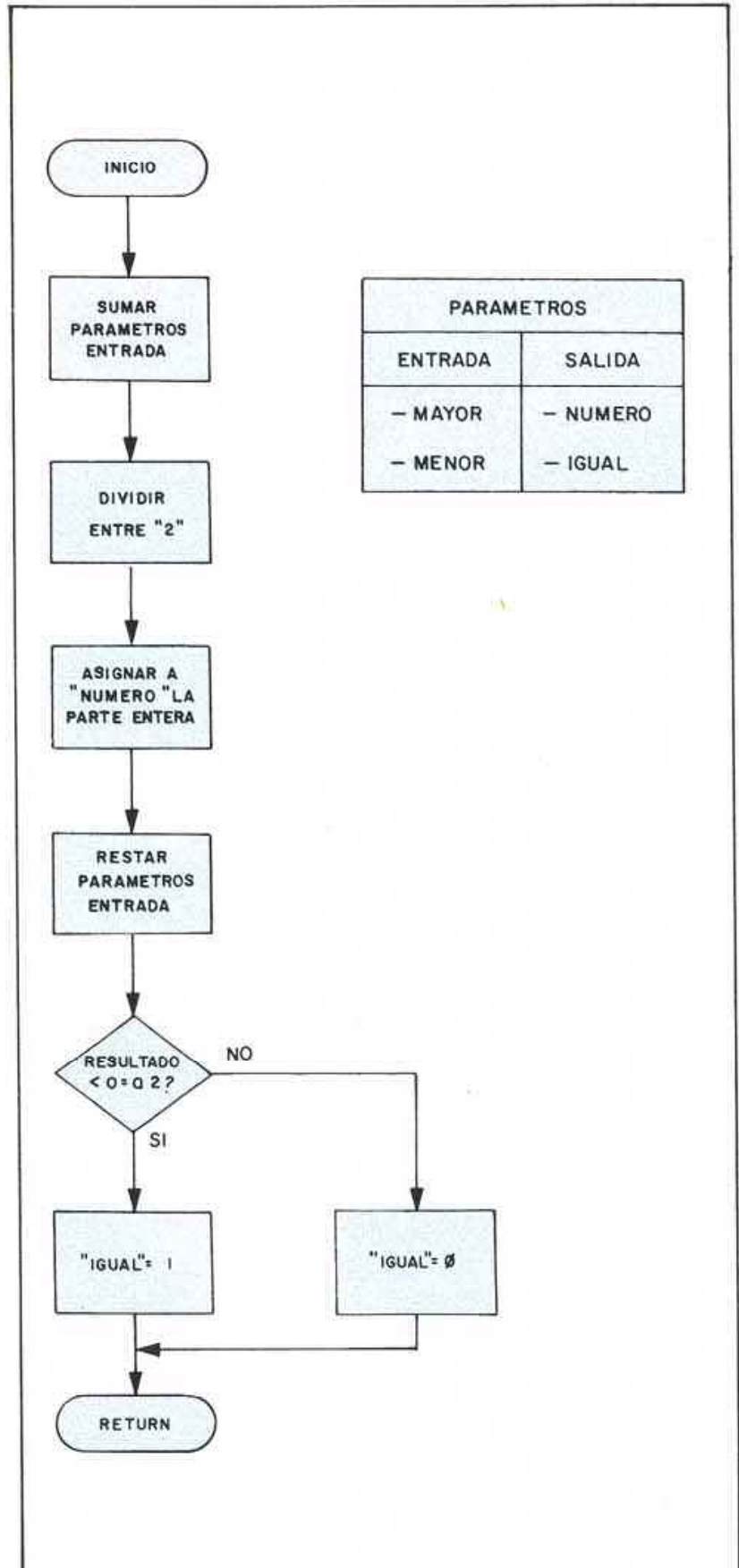
255-290 : Introducción de la clave correspondiente a la pista (M, N o I), si no es «igual», se salta a la línea 200 para calcular un nuevo número.

300 : El número ha sido acertado.

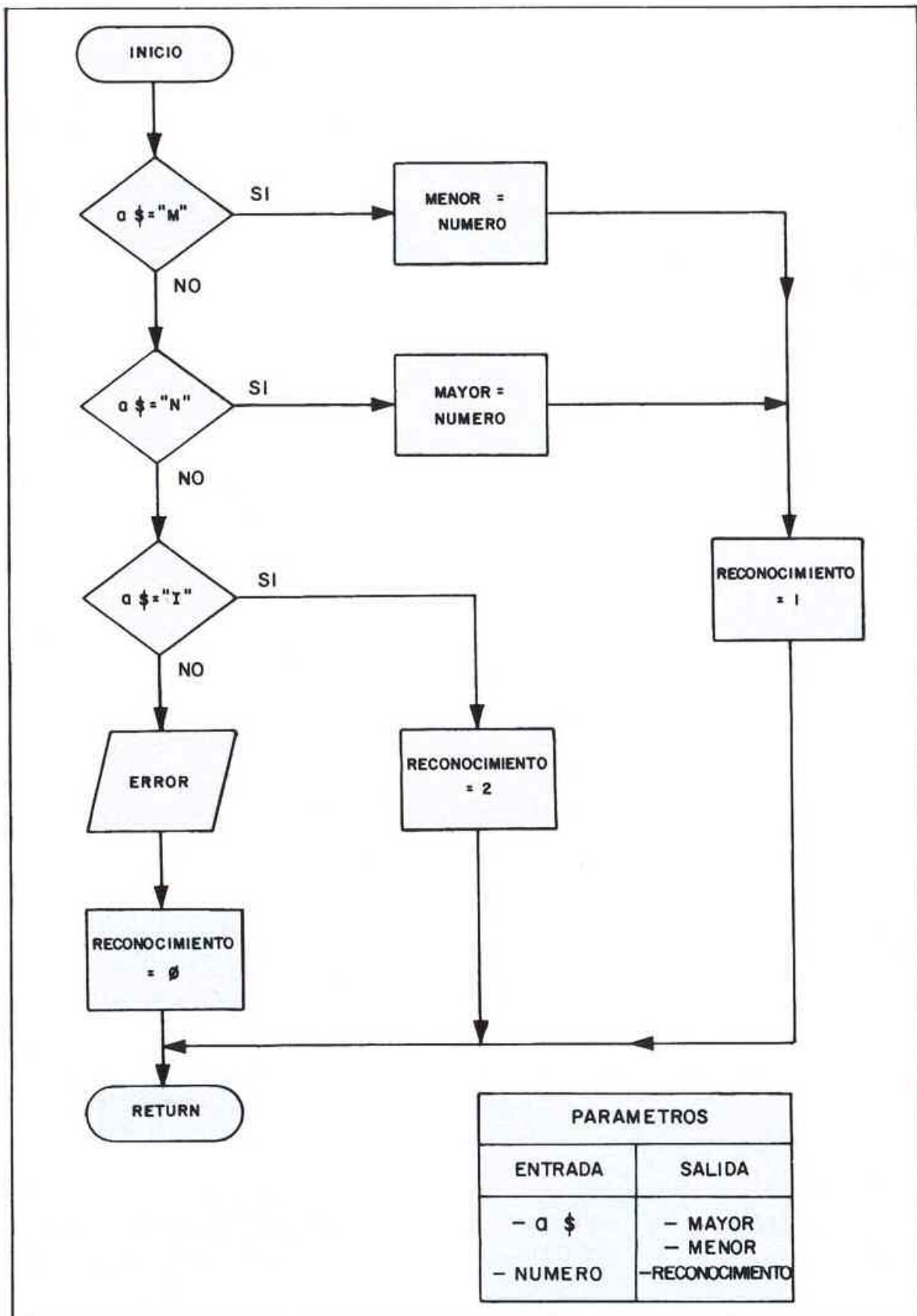
310 : ¿Se desea continuar?

320-330 : En caso afirmativo, se borra la pantalla y se comienza de nuevo en la línea «100».

1000-1070 : Subrutina «INSTRUCCION». Se visualizan las instrucciones durante un tiempo y luego se borra la pantalla. Se llama a la



Programa «Adivino» subrutina «Cálculo».



Programa «Adivino» subrutina «Verificación».

PROGRAMA 2

```

10 REM *****
*          *
* CURSO/BASIC *
*          *
*****
*          *
* LONGITUD *
*          *
*****

20 BORDER 4: PAPER 4: INK 1: C
LS
30 REM

*****
*          *
* VARIABLES *
*          *
*****

40 LET menu=1000
50 LET visualizacion=2000
100 REM

*****
*          *
* BUCLE CENTRAL *
*          *
*****

110 GO SUB menu
120 INPUT "Unidad a transformar
>>> ";unidad
125 IF unidad<1 OR unidad>7 THE
N GO TO 120
130 CLS
140 INPUT "Cantidad >>> ";valor
150 IF valor<0 OR valor>9999999
9 THEN GO TO 140
160 LET operacion=1000+(unidad*
100)
170 GO SUB operacion
180 GO SUB visualizacion
190 INPUT "Quiere continuar (S/
N) >>> "; LINE a$
200 IF a$="s" OR a$="S" THEN CL
S : GO TO 100
210 IF a$="n" OR a$="N" THEN ST
OP
220 GO TO 190
999 REM

*****
*          *
* SUBROUTINAS *
*          *
*****

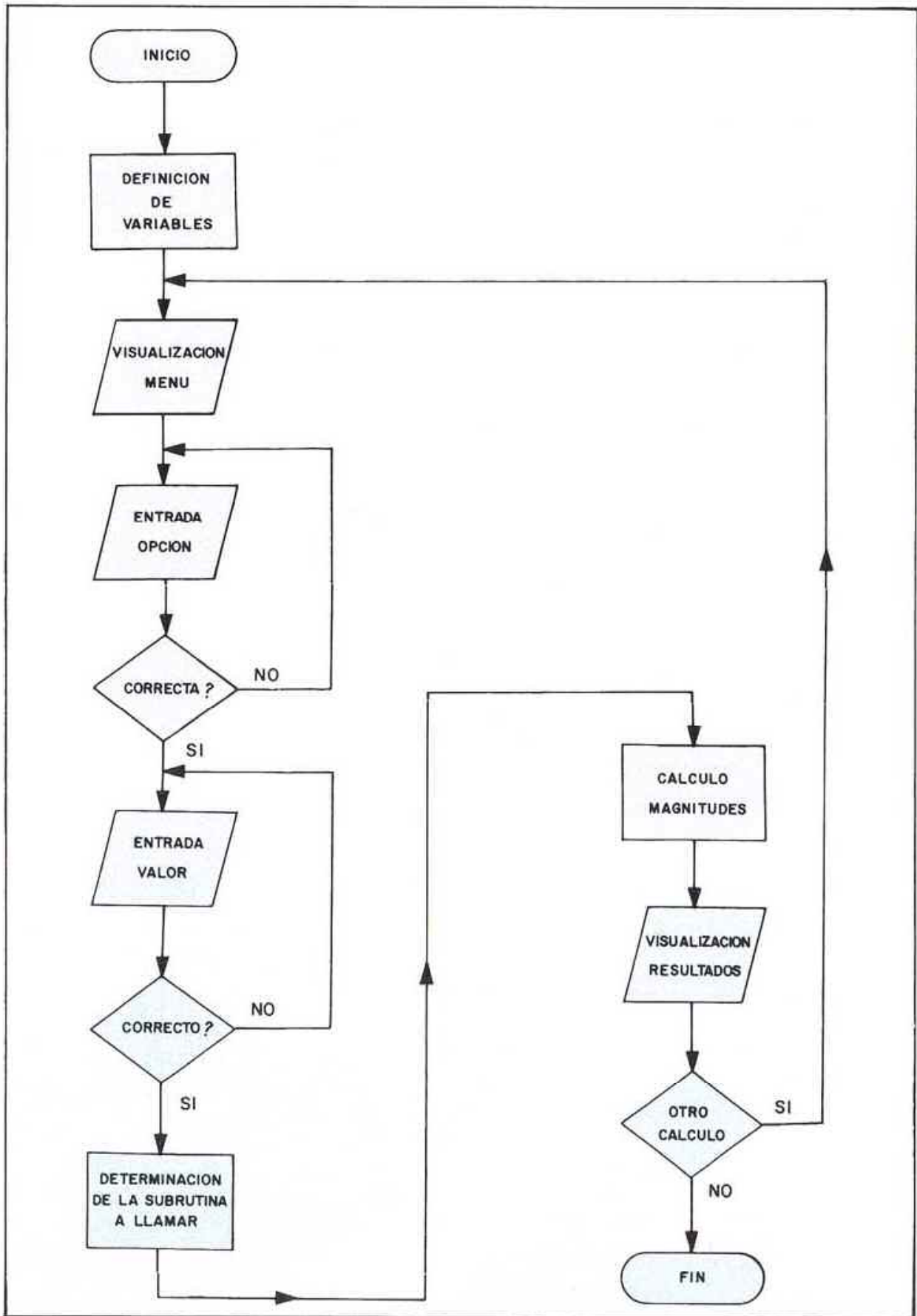
1000 REM MENU
1010 PRINT AT 1,5;"UNIDADES DE L
ONGITUD"
1020 PRINT AT 2,5;"_____
"
1030 PRINT AT 6,8;"1 - Milimetro
."
1040 PRINT AT 8,8;"2 - Centimetr
o."
1050 PRINT AT 10,8;"3 - Decimetr
o."
1060 PRINT AT 12,8;"4 - Metro."
1070 PRINT AT 14,8;"5 - Decametr
o."
1080 PRINT AT 16,8;"6 - Hectomet
ro."
1090 PRINT AT 18,8;"7 - Kilometr
o."
1095 RETURN
1100 REM MILIMETRO
1110 LET milimetro=valor
1120 LET centimetro=milimetro/10
1130 LET decimetro=centimetro/10
1140 LET metro=decimetro/10

```

```

1150 LET decametro=metro/10
1160 LET hectometro=decametro/10
1170 LET kilometro=hectometro/10
1180 RETURN
1200 REM CENTIMETRO
1210 LET centimetro=valor
1220 LET milimetro=centimetro*10
1230 LET decimetro=centimetro/10
1240 LET metro=decimetro/10
1250 LET decametro=metro/10
1260 LET hectometro=decametro/10
1270 LET kilometro=hectometro/10
1280 RETURN
1300 REM DECIMETRO
1310 LET decimetro=valor
1320 LET milimetro=decimetro*100
1330 LET centimetro=decimetro*10
1340 LET metro=decimetro/10
1350 LET decametro=metro/10
1360 LET hectometro=decametro/10
1370 LET kilometro=hectometro/10
1380 RETURN
1400 REM METRO
1410 LET metro=valor
1420 LET milimetro=metro*1000
1430 LET centimetro=metro*100
1440 LET decimetro=metro*10
1450 LET decametro=metro/10
1460 LET hectometro=decametro/10
1470 LET kilometro=hectometro/10
1480 RETURN
1500 REM DECAMETRO
1510 LET decametro=valor
1520 LET milimetro=decametro*100
00
1530 LET centimetro=decametro*10
00
1540 LET decimetro=decametro*100
1550 LET metro=decametro*10
1560 LET hectometro=decametro/10
1570 LET kilometro=hectometro/10
1580 RETURN
1600 REM HECTOMETRO
1610 LET hectometro=valor
1620 LET milimetro=hectometro*10
0000
1630 LET centimetro=hectometro*1
0000
1640 LET decimetro=hectometro*10
00
1650 LET metro=hectometro*100
1660 LET decametro=hectometro*10
1670 LET kilometro=hectometro/10
1680 RETURN
1700 REM KILOMETRO
1710 LET kilometro=valor
1720 LET milimetro=kilometro*100
0000
1730 LET centimetro=kilometro*10
0000
1740 LET decimetro=kilometro*100
00
1750 LET metro=kilometro*1000
1760 LET decametro=kilometro*100
1770 LET hectometro=kilometro*10
1780 RETURN
2000 REM VISUALIZACION
2010 PRINT AT 2,10;"RESULTADOS"
2020 PRINT AT 3,10;"_____
"
2030 PRINT AT 6,1;"Milímetros ..
";milimetro
2040 PRINT AT 8,1;"Centímetros .
";centimetro
2050 PRINT AT 10,1;"Decímetros .
";decimetro
2060 PRINT AT 12,1;"Metros .....
";metro
2070 PRINT AT 14,1;"Decametros .
";decametro
2080 PRINT AT 16,1;"Hectometros
";hectometro
2090 PRINT AT 18,1;"Kilometros .
";kilometro
2100 RETURN

```



Estructura programa «longitud».

| | | |
|---|--|---|
| <p>subrutina «TEMPORIZACION».</p> <p>1100-1130 : Subrutina «ERROR». Visualiza un mensaje de error durante un tiempo. La subrutina «TEMPORIZACION» es utilizada.</p> | <p>la variable «reconocimiento»; ésta puede tener tres valores, «0» si el valor de «a\$» no corresponde con ninguna de las pistas (M, N o I). «1» si la opción elegida es «M» o «N», y «2» si la opción es «I».</p> | <p>za el equivalente en las restantes unidades.</p> <p>Su estructura general es:</p> |
| <p>1200-1230 : Subrutina «TEMPORIZACION». Tiene como parámetro de entrada la variable «retardo», dependiendo de este valor el tiempo de temporización. Aproximadamente se consigue un segundo de retardo, asignando el valor «100» a dicha variable.</p> | <p>1500-1590 : Subrutina «ACIERTO». Se encarga de visualizar, cuando el ordenador acierta, el número de intentos realizados; al cabo de, aproximadamente, cuatro segundos visualiza el record de todas las jugadas.</p> | <p>10 : Comentario con el nombre del programa.</p> <p>20 : Asignación de los colores verde para fondo y borde, y azul para los caracteres.</p> <p>40-50 : Definición de variables.</p> <p>110 : Llamada a la subrutina «MENU».</p> <p>120 : Entrada «unidad».</p> <p>125 : Verificación de la selección; tiene que estar comprendida entre «1» y «7».</p> |
| <p>1300-1330 : Subrutina «CALCULO». Calcula un número en función de los parámetros de entrada «mayor» y «menor», el resultado es devuelto al programa principal en el parámetro de salida «numero». La variable «igual» es también utilizada como parámetro de salida, si su valor es igual a «1» es que el ordenador está seguro del número.</p> | <p>1600-1650 : Subrutina «SEGUIR». Comienza de nuevo el juego si se introduce la letra «S» (SI) o se termina si es la «N» (NO). Si se pulsa otra letra, se ejecuta una llamada a la subrutina «ERROR».</p> <p>El programa número «2», realiza transformaciones entre magnitudes expresadas en unidades de longitud.</p> <p>Presenta un menú con las diversas unidades; para elegir una de ellas, es necesario introducir el código correspondiente y, a continuación, la magnitud. Una vez realizados los cálculos, el programa visuali-</p> | <p>130 : Borrado pantalla.</p> <p>140 : Entrada «valor».</p> <p>150 : Verificación de la magnitud, debe estar comprendida entre «0» y «9999999».</p> <p>160 : Cálculo del número de línea, correspondiente a la subrutina encargada de realizar las transformaciones elegidas.</p> <p>170 : Llamada a la subrutina calculada en la línea 160.</p> <p>180 : Llamada a la subrutina de visualización de resultados.</p> |
| <p>1400-1470 : Subrutina «VERIFICACION». Tiene como parámetro de entrada la variable «a\$», en función de su valor, modifica el de las variables «mayor» y «menor». El parámetro de salida está asignado en</p> | <p>la variable «reconocimiento»; ésta puede tener tres valores, «0» si el valor de «a\$» no corresponde con ninguna de las pistas (M, N o I). «1» si la opción elegida es «M» o «N», y «2» si la opción es «I».</p> | <p>1900-220 : Rutina para saltar al principio y realizar otro cálculo o, por el contrario, terminar.</p> <p>1000-1095 : Subrutina «MENU». Visualiza el menú con las diversas opciones.</p> <p>1100-1180 : Subrutina «MILI-</p> |

- METRO». Asigna a la variable «milímetro» el contenido de la variable «valor». A partir de este dato realiza las equivalencias con las restantes unidades.
- 1200-1280 : Subrutina «CENTI-METRO». Idem con la variable «centímetro».
- 1300-1380 : Subrutina «DECI-METRO». Idem con la variable «decímetro».
- 1400-1480 : Subrutina «METRO». Idem con la variable «metro».
- 1500-1580 : Subrutina «DECA-METRO». Idem con la variable «decámetro».
- 1600-1680 : Subrutina «HECTOMETRO». Idem con la variable «hectómetro».
- 1700-1780 : Subrutina «KILO-METRO». Idem con la variable «kilómetro».
- 2000-2100 : Subrutina «VISUALIZACION». Visualiza en pantalla los resultados obtenidos con cualquiera de las anteriores subrutinas. ■

DATOS DE UN PROGRAMA

Cuando estudiamos las sentencias «LET» e «INPUT», vimos que se podían asignar o introducir datos en nuestros programas; pero cuando estos datos son de valor constante y numerosos, podemos utilizar las sentencias «DATA» para almacenarlos y «READ» para leerlos.

READ

Acceso al teclado



MODO E



MODO E

Tipo de sentencia

Comando de entrada.

Definición

Con esta sentencia, se pueden leer los datos que haya dentro de un programa; su estructura es:

| SENTENCIA | ARGUMENTO |
|-----------|-----------------------|
| READ | variable, variable... |

Las variables pueden ser tanto numéricas como de cadena, pero teniendo en cuenta que deben ir en concordancia con el dato leído, y separadas por comas.

Ejemplos:

— READ día, mes.

- READ j\$.
 - READ numero, a\$, nota.
 - READ valor.
- El dato es asignado a la variable del argumento.

DATA

Acceso al teclado

Tipo de sentencia

Datos.

Definición

Los valores que forman el argumento de esta sentencia, sirven para ser asignados a las variables de las instrucciones «READ», y forman, por tanto, tablas de datos.

La estructura de esta sentencia es:

| SENTENCIA | ARGUMENTO |
|-----------|-------------------------|
| DATA | constante, constante... |

Las constantes pueden ser numéricas o alfanuméricas.

Ejemplos:

- DATA 30, 47, 3225, 17.14
- DATA «MICROHOBBY», «CURSO», «BASIC»
- DATA «SPECTRUM», 23, 10, «MICRO»
- DATA 8

Los valores del argumento deben ir separados por comas y pueden estar incluidos en cualquier parte del programa, incluso después de una sentencia «STOP», ya que el ordenador los ignora mientras no se ejecute una instrucción «READ».

Ejemplo:

```
50 DATA 10, 20, 67
100 DATA 5, 15, 27, 33
170 DATA «MICRO», «CHIP»
300 DATA «ENTRADA», 30
```

aunque por razones de claridad y estructuración del programa, conviene que estén todos juntos al principio o a final de éste.

Ejemplo:

```
1000 DATA 10, 20, 67, 5
1010 DATA 15, 27, 33
1020 DATA «MICRO», «CHIP», «ENTRADA»
1030 DATA 30
```

La cantidad de valores a incluir dentro de una instrucción «DATA», viene determinada únicamente por la capacidad de una línea (22 filas).

Utilización de «READ» y «DATA»

Cuando el intérprete BASIC encuentra una sentencia del tipo:

READ variable

analiza una por una todas las

líneas del programa, hasta que encuentra la primera que contiene «DATA»; posteriormente, asigna a la variable el primer dato de ésta. La siguiente vez que encuentra «READ» se le asigna el segundo dato, y así sucesivamente.

Ejecute el siguiente programa:

```

10 REM *****
   : READ/DATA :
   : ***** :
20 READ a$,b$
30 READ numero
40 READ f$
50 READ q$,valor
60 PRINT a$," "+b$
70 PRINT numero
80 PRINT " "f$,valor
90 PRINT " "q$
100 REM *****
   : DATOS :
   : ***** :
110 DATA "MICROHOBBY","SEMANAL"
120 DATA 1985,"BASIC/SINCLAIR"
130 DATA "SPECTRUM",16

```

según se van ejecutando las sentencias «READ», se van asignando los sucesivos datos, así...

a\$ tendrá el valor «MICROHOBBY»

b\$ = «SEMANAL»

numero = 1985

f\$ = «BASIC/SINCLAIR»

q\$ = «SPECTRUM»

valor = 16

Para conocer el ordenador cuál es el siguiente dato que tiene que leer, la tabla de datos tiene un *puntero* o *índice* que lo señala y que se incrementa en uno cada vez que se hace una lectura.

Una cosa importante es que no pueden realizarse más lecturas que datos haya en el programa, ya que tendríamos un mensaje de error; sin embargo, puede haber más datos que lecturas.

La instrucción «READ» puede formar parte de un bucle,

de esta manera puede agilizarse la lectura de datos.

Ejemplo:

```

10 REM *****
   : BUCLE READ :
   : ***** :
15 LET total=0
20 FOR n=1 TO 22
30 READ numero
40 LET total=total+numero
50 PRINT numero,total
60 NEXT n
70 REM *****
   : DATOS :
   : ***** :
80 DATA 23,45,67,45,21,67,89,3
90 DATA 12,45,83,27,54,12,67,7
0
100 DATA 34,56,71,89,21,83

```

Dentro de una tabla de datos puede leerse uno determinado; en la siguiente aplicación, introduciendo un número entre 1 y 12, el ordenador visualiza el mes correspondiente:

```

10 REM *****
   : MESES :
   : ***** :
20 INPUT "Mes (1-12) >>>";mes
30 IF mes<1 OR mes>12 THEN GO TO 20
40 LET puntero=1
50 READ a$
60 IF puntero>mes THEN LET puntero=puntero+1: GO TO 50
70 PRINT puntero,a$
80 REM *****
   : DATOS :
   : ***** :
90 DATA "ENERO","FEBRERO","MARZO"
100 DATA "ABRIL","MAYO","JUNIO"
110 DATA "JULIO","AGOSTO","SEPTIEMBRE"
120 DATA "OCTUBRE","NOVIEMBRE","DICIEMBRE"

```

Veamos otro ejemplo de aplicación de las sentencias «READ/DATA».

```

10 REM *****
   : PRIMOS :
   : ***** :
15 PRINT "Numero Primo","Cuadrado"

```

```

17 PRINT " "
20 READ primo
30 IF NOT primo THEN STOP
40 LET Cuadrado=primo^2
50 PRINT primo,Cuadrado
60 GO TO 20
100 REM *****
   : DATOS :
   : ***** :
110 DATA 1,2,3,5,7,11,13,17
120 DATA 19,23,29,31,37,41,43
130 DATA 47,59

```

En este caso, el ordenador calcula el cuadrado de los números primos menores de cincuenta, que se han introducido como constantes en una serie de «DATAS». Al final de los datos se introduce el valor «Ø», como código de ruptura; cuando se realiza la lectura de este código, el programa se para, de esta manera no es necesario llevar la cuenta de la cantidad de datos. Para preguntar por esta condición se ha utilizado:

IF NOT primo THEN STOP

la condición se hace verdadera y por tanto se ejecuta «STOP», cuando la variable «primo» es igual a «Ø», ver función «NOT» (pág. 35).

RESTORE

Acceso al teclado



Tipo de sentencia

Comando de programación.

Definición

Una de las ventajas de las tablas de datos es poder leer éstos cuantas veces deseemos. Para volver al principio de la tabla o a cualquier parte de ella, se necesita restaurar el puntero o índice, esto se consigue con la sentencia «RESTORE».

Su estructura es la siguiente:

| SENTENCIA | ARGUMENTO |
|-----------|--------------|
| RESTORE | N.º de línea |

Ejemplos:

- RESTORE 4Ø.
- RESTORE 123Ø.
- RESTORE.

Cuando el argumento se omite, el intérprete BASIC toma por defecto el valor Ø. Al ejecutar un programa con el comando directo «RUN» inicia el puntero en la primera sentencia «DATA».

Si no se restaura el puntero al finalizar la lectura de la tabla, el siguiente dato que se quiera leer provocará un mensaje de error.

Ejemplo:

```

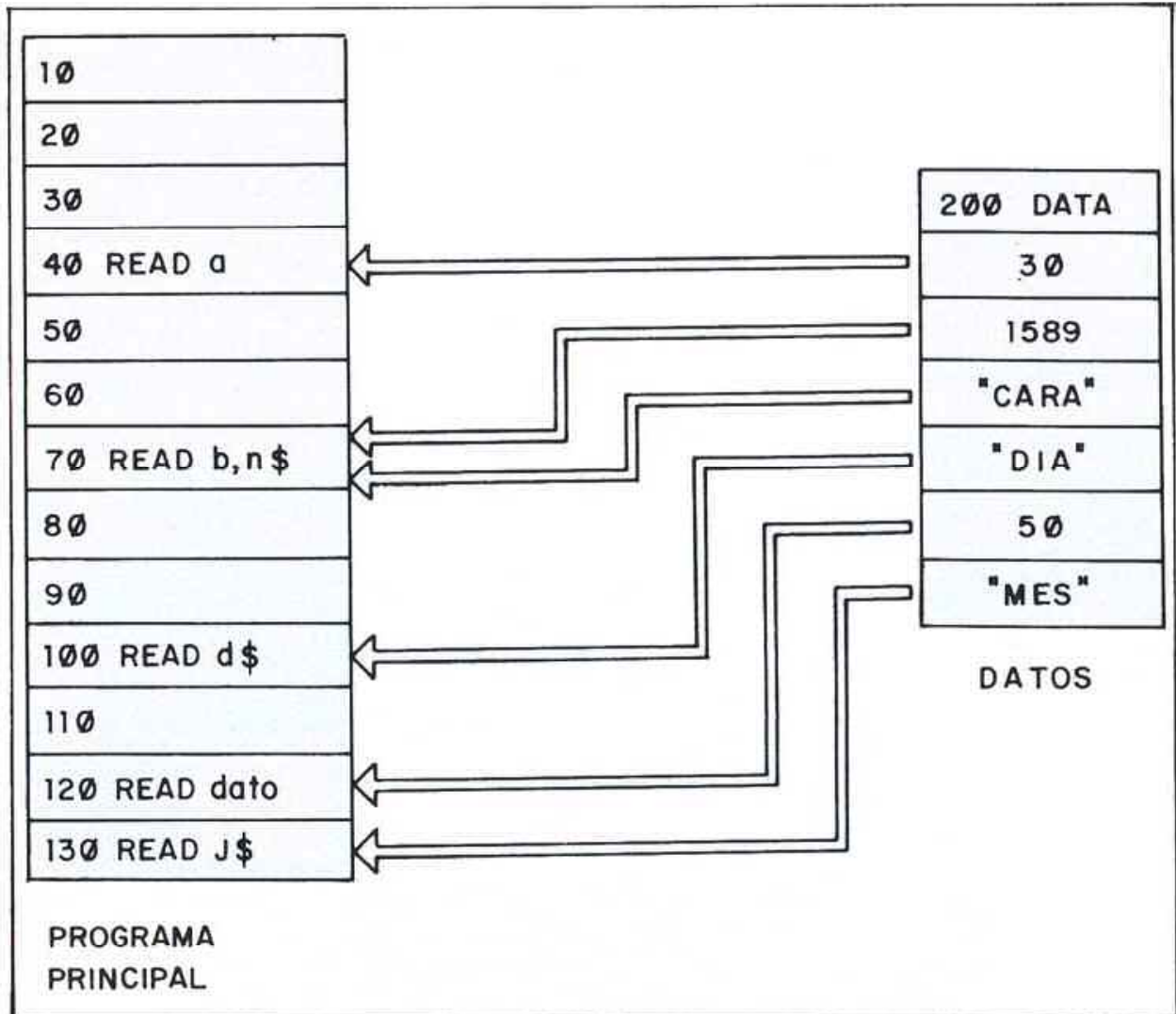
1Ø READ código
2Ø READ precio
3Ø PRINT código
4Ø PRINT precio
5Ø RESTORE
6Ø READ número
    
```

```

7Ø READ valor
8Ø PRINT número
9Ø PRINT valor
1ØØ DATA 27, 35ØØ
    
```

La sentencia «RESTORE» de la línea 5Ø, permite leer la tabla dos veces. En la primera lectura, el valor «27» es asignado a la variable «código»; sin embargo, en la segunda se asigna a la variable «número»; otro tanto ocurre con el valor «35ØØ» que se asigna primeramente a la variable «precio», y posteriormente a «valor».

En el siguiente programa tiene como argumento la sentencia «RESTORE», una variable de tipo numérico, cuyo va-



Lectura de datos.

lor está en función de un «INPUT»; dependiendo de éste la inicialización del puntero y, por tanto, del acceso a los datos.

```

10 REM *****
   : RESTORE
   : *****
20 INPUT "Clave (1 a 5) >>> ";
clave
30 IF clave<1 OR clave>5 THEN
GO TO 20
40 LET indice=1000+(clave*10)
50 RESTORE indice
60 FOR n=1 TO 4
70 READ codigo
80 PRINT codigo
90 NEXT n
1000 REM *****
   : DATOS
   : *****
1010 DATA 1,2,3,4
1020 DATA 5,6,7,8
1030 DATA 9,10,11,12
1040 DATA 13,14,15,16
1050 DATA 17,18,19,20

```

Errores

Cuando se manejan sentencias del tipo READ/DATA, hay tres tipos de error que suelen producirse frecuentemente.

- a) Cuando se ejecuta una sentencia «READ» y el puntero se encuentra al final de la tabla, indicando que no hay más datos. El mensaje que se visualiza es:

E Out of DATA

Ejemplo:

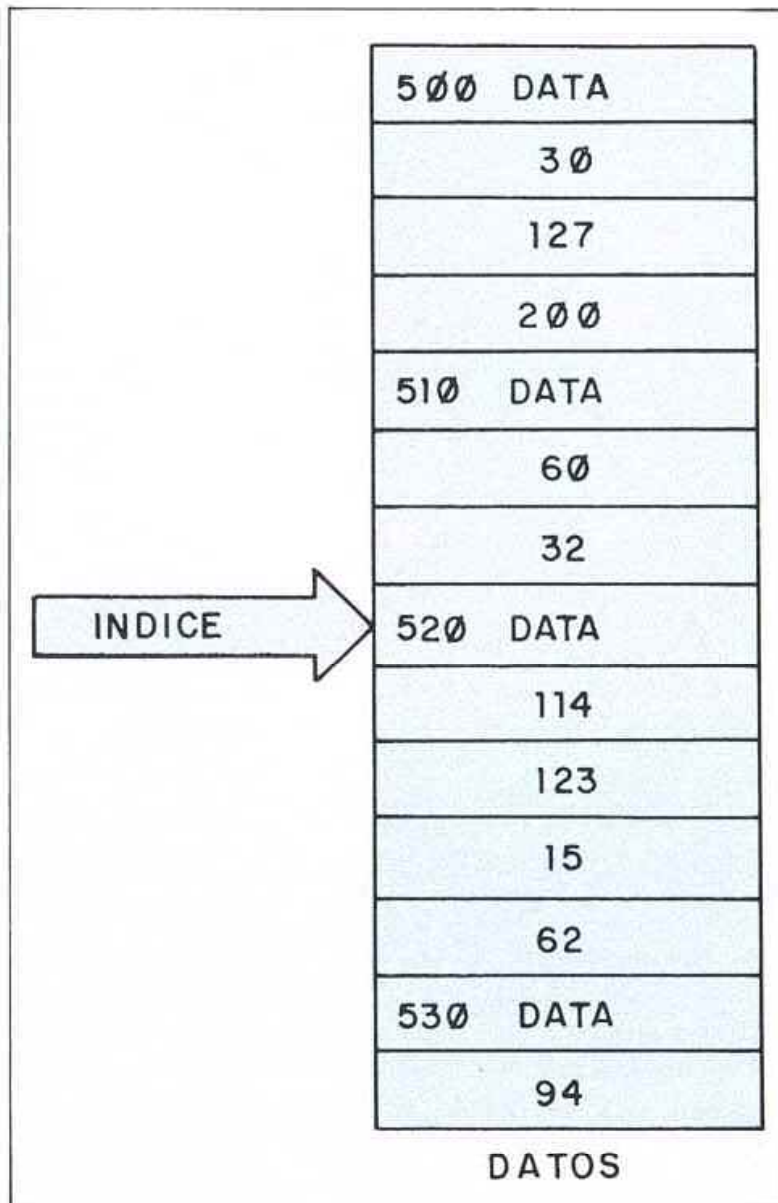
```

10 REM *****
   : ERROR 1
   : *****
20 FOR n=1 TO 5
30 READ nota
40 PRINT nota
50 NEXT n
100 REM *****
   : DATOS
   : *****
110 DATA 10,23,56,78

```

se pretende leer cinco datos, cuando en realidad sólo hay cuatro.

- b) Cuando al confeccionar la tabla de datos se incluye un valor numérico en lugar de uno alfanumérico, o viceversa. Al ejecutarse di-



Mecanismo "Restore 520".

cho programa y tratar de leer un dato que no está en concordancia con el de la variable que acompaña a «READ», se produce el error:

C Nonsense in BASIC

Ejemplo:

```

10 REM *****
   : ERROR 2
   : *****

```

```

15 PRINT "NOTA"; " ALUMNO"
16 PRINT " "; " "
20 FOR n=1 TO 8
30 READ nota,js
40 PRINT nota,js
50 NEXT n
100 REM *****
   : DATOS
   : *****
110 DATA 5,"ANTONIO",8,"GERARDO"
120 DATA 2,"SEBASTIAN",10,"LUIS"
130 DATA 7,"CARLOS",9,"PABLO"
140 DATA 3,4,6,"RAFAEL"

```

En la línea 140, el segundo dato debería ser alfanumérico.

PROGRAMA 1

```

10 REM *****
*          *
*  CURSO/BASIC *
*          *
*****
*  GEOGRAFIA 1 *
*          *
*****

20 BORDER 4: PAPER 4: INK 1: C
LS
30 GO SUB 1000
40 REM *****
*          *
*  BUCLE PRINCIPAL *
*          *
*****

50 PRINT AT 1,3;"AUTONOMIAS DE
LA PENINSULA"
60 PRINT AT 2,3;"_____
"

70 FOR c=5 TO 19
80 READ a$
90 LET opcion=c-4
100 PRINT AT c,3;opcion
110 IF opcion<10 THEN PRINT " "
;
120 PRINT " - ";a$
130 NEXT c
140 INPUT AT 0,1;"Que Autonomia
desea visualizar >>> ";opcion
150 IF opcion<1 OR opcion>15 TH
EN GO TO 140
160 CLS
170 LET direccion=2000+(opcion*
100)
180 RESTORE direccion
190 READ a$
200 PRINT AT 1,3;a$
210 PRINT "_____"

220 READ numero
230 FOR n=1 TO numero
240 PRINT " - ";
250 READ a$,b$
260 PRINT a$,"(",b$;")"
270 NEXT n
280 INPUT "Pulse ""ENTER"" para
retornar "; LINE a$
290 RESTORE 1100
300 CLS
310 GO TO 40
1000 REM *****
*          *
*  DEFINICION "N" *
*          *
*****

1010 RESTORE 1060
1020 FOR d=USR "n" TO USR "n"+7
1030 READ dato
1040 POKE d,dato
1050 NEXT d
1060 DATA 24,66,98,82,74,70,66,0
1070 RETURN
1100 REM *****
*          *
*  MENU *
*          *
*****

1110 DATA "ANDALUCIA","ARAGON","
ASTURIAS","CANTABRIA","CASTILLA-
LA MANCHA"
1120 DATA "CASTILLA-LEON","CATAL
UNA","COMUNIDAD VALENCIANA","EXT
REMA DURA","GALICIA"
1130 DATA "MADRID","MURCIA","NAV

```

```

ARRA","PAIS VASCO","LA RIOJA"
2000 REM *****
*          *
*  PROVINCIAS *
*          *
*****

2100 DATA "ANDALUCIA",8,"ALMERIA
","ALMERIA","CADIZ","CADIZ","COR
DOBA","CORDOBA","GRANADA","GRANA
DA","HUELVA","HUELVA","JAEN","JA
EN","MALAGA","MALAGA","SEVILLA",
"SEVILLA"
2200 DATA "ARAGON",3,"HUESCA","H
UESCA","TERUEL","TERUEL","ZARAGO
ZA","ZARAGOZA"
2300 DATA "ASTURIAS",1,"ASTURIAS
","OVIEDO"
2400 DATA "CANTABRIA",1,"CANTABR
IA","SANTANDER"
2500 DATA "CASTILLA-LA MANCHA",5
,"ALBACETE","ALBACETE","CIUDAD R
EAL","CIUDAD REAL","CUENCA","CUE
NCA","GUADALAJARA","GUADALAJARA",
"TOLEDO","TOLEDO"
2600 DATA "CASTILLA-LEON",9,"AVI
LA","AVILA","BURGOS","BURGOS","L
EON","LEON","PALENCIA","PALENCIA
","SALAMANCA","SALAMANCA","SEGOV
IA","SEGOVIA","SORIA","SORIA","V
ALLADOLID","VALLADOLID","ZAMORA",
"ZAMORA"
2700 DATA "CATALUNA",4,"BARCELON
A","BARCELONA","GERONA","GERONA",
"LERIDA","LERIDA","TARRAGONA","
TARRAGONA"
2800 DATA "COMUNIDAD VALENCIANA",
3,"ALICANTE","ALICANTE","CASTEL
LON","C. DE LA PLANA","VALENCIA",
"VALENCIA"
2900 DATA "EXTREMADURA",2,"BADAJ
OZ","BADAJOZ","CACERES","CACERES"

3000 DATA "GALICIA",4,"LA CORUNA",
"LA CORUNA","LUGO","LUGO","ORE
NSE","ORENSE","PONTEVEDRA","PONT
EVEDRA"
3100 DATA "MADRID",1,"MADRID","M
ADRID"
3200 DATA "MURCIA",1,"MURCIA","M
URCIA"
3300 DATA "NAVARRA",1,"NAVARRA",
"PAMPLONA"
3400 DATA "PAIS VASCO",3,"ALAVA",
"VITORIA","GUIPUZCOA","SAN SEBA
STIAN","VIZCAYA","BILBAO"
3500 DATA "LA RIOJA",1,"LA RIOJA",
"LOGROÑO"

2270 DATA "SELLA","VALLE SAJAMBR
E","LEON",1,"PONGA"
2300 DATA 7,"MEDITERRANEA"
2310 DATA "EBRO","FONTIBRE","SAN
TANDER",7,"ARAGON","EGA","GALLEG
O","GUADALOPE","HUERVA","JALON",
"SEGRE"
2320 DATA "GUADALHORCE","SIERRA
DE GIBALTO","MALAGA-GRANADA",0
2330 DATA "JUCAR","CERRO SAN FEL
IPE","CUENCA",1,"CABRIEL"
2340 DATA "LLOBREGAT","FONT DEL
LLOBREGAT","BARCELONA",2,"CARDON
ER","NOYA"
2350 DATA "SEGURA","SIERRA DEL S
EGURA","JAEN",3,"ARGOS","BENAMOR",
"MONDO"
2360 DATA "TER","MONT COSTA BOND",
"GERONA",2,"FRESSER","ONAR"
2370 DATA "TURIA","MUELA SAN JUA
N","TERUEL",0

```

c) En las tablas de datos con valores de cadena, puede suceder que se nos olvide

colocar las correspondientes comillas. El intérprete BASIC, al ejecutarse el pro-

grama, tomará dicho dato como variable numérica; si no existe ninguna variable


```

CIN", "TERUEL", 7, "ALAGON", "ALBERC
HE", "ALMONTE", "GUADARRAMA", "GUAD
IELA", "JARAMA", "TIETAR"
2170 DATA "TAMBRE", "MONTES DE BO
CELO", "LA CORUNA", 2, "SAMO", "LENG
UELLE"
2180 DATA "ULLA", "MONTE CEBREIRO
", "LUGO", 3, "DEZA", "ISO", "SAR"
2200 DATA 7, "CANTABRICA"
2210 DATA "BIDASOA", "MAYA", "NAVA
ARRA", 0

```

```

2220 DATA "EO", "MONTES DEL CADAB
O", "LUGO", 0
2230 DATA "NALON", "PUERTO TARNA"
, "OVIEDO", 4, "CAUDAL", "NARCEA", "N
ORA", "TRUBIA"
2240 DATA "NAVIA", "SIERRA RAÑADO
IRO", "LUGO", 0
2250 DATA "NERVIÓN", "PEÑA DE ORD
UNA", "ALAVA", 1, "IBAZABAL"
2260 DATA "ORIA", "MONTE AITZGORR
I", "GUIPUZCOA", 0

```

Programas

Como aplicación de las sentencias «READ/DATA», se presentan, en esta ocasión, dos programas de utilidad didáctica. Ambos están relacionados con un área muy importante dentro de la enseñanza: la Geografía.

El alfabeto español tiene una letra que no está incluida en el juego de caracteres del Spectrum, ésta es la «Ñ»; por tanto ha sido necesario diseñar un nuevo gráfico (GDU) con dicha forma. La técnica empleada para realizarlo será explicada en el capítulo dedicado a gráficos. Para acceder a la «Ñ» es necesario pasar a modo gráfico (**G**), como recordará, basta con pulsar simultáneamente la tecla «CAPS SHIFT» y «9», una vez en este modo, debe pulsarse la tecla «N».

ATENCIÓN

Hasta que el programa no se ejecute, *no se visualizará* el gráfico correspondiente a la «Ñ», por tanto no se preocupe si ésta no aparece al pasar al modo **G**.

Para retornar al modo anterior pulse la tecla «9».

El primer programa está dedicado al estudio de las *Autonomías*. Al ejecutar el programa, se presenta un menú con las distintas Autonomías de la Península. Cada una de ellas tiene a su izquierda un número de opción, éste sirve como

referencia para seleccionarla. Una vez elegida la opción, aparece en la parte superior de la pantalla el nombre de la Autonomía, y a continuación, el nombre de las provincias que la componen con sus correspondientes capitales, encerradas entre paréntesis.

La estructura del programa es:

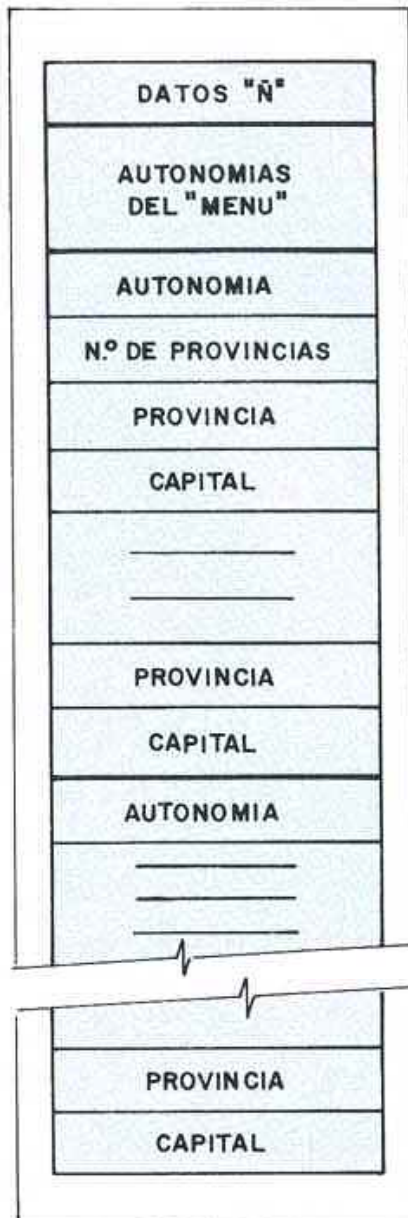
- 1Ø : Comentario con el nombre del programa.
- 2Ø : Asignación del color verde para fondo y papel y azul para los caracteres.
- 3Ø : Llamada a la subrutina que define como GDU, la «Ñ».
- 4Ø : Comienzo del programa principal.
- 5Ø-6Ø : Rótulo de presentación.
- 7Ø-13Ø : Bucle para visualizar el número de opción y el nombre de las Autonomías, éstas están contenidas en una tabla de datos.
- 14Ø-15Ø : Entrada opción y comprobación.
- 16Ø : Borrado pantalla.
- 17Ø : *Algoritmo* empleado para el cálculo del número de línea, dónde debe inicializarse el puntero de la tabla de datos.

- 18Ø : Inicialización del puntero (RESTORE).
- 19Ø-21Ø : Lectura y visualización del nombre de la Autonomía.
- 22Ø : Lectura del número de provincias.
- 23Ø-27Ø : Bucle para la lectura y visualización de las provincias y sus capitales.
- 28Ø : Entrada de la tecla «ENTER».
- 28Ø : Inicialización de la tabla.
- 3ØØ-31Ø : Borrado pantalla y regreso al menú principal.
- 1ØØØ-1Ø7Ø : Subrutina para definición de la tecla «Ñ».
- 11ØØ-113Ø : Datos del menú principal.
- 2ØØØ-35ØØ : Datos correspondientes a las provincias y sus capitales.

El segundo programa sirve para estudiar o consultar sobre la hidrografía española. Al ejecutarse presenta en pantalla el menú con las tres vertientes:

- ATLANTICA.
- CANTABRICA.
- MEDITERRANEA.

Al elegirse una de ellas aparece otro menú con los ríos que desembocan. Al seleccionar uno de ellos, se visualiza



Estructura de datos del programa "Autonomías".

en pantalla la información correspondiente a:

- Nombre del río.
- Nacimiento.
- Provincia.
- Afluentes más importantes (si los tiene).

Pulsando «ENTER» se retorna al menú principal. Si al elegir las opciones se selecciona una que no existe, aparece en la parte inferior de la pantalla un mensaje de error.

La estructura del programa es la siguiente:

- 10 : Comentario con el nombre del programa.
- 20 : Asignación del color azul para el borde, verde para el fondo y negro para los caracteres.
- 30 : Llamada a la subrutina que define la «Ñ».
- 40 : Comienzo del programa principal.
- 50-100 : Presentación del menú con las vertientes.
- 110-120 : Entrada de opción y comprobación.
- 130 : Algoritmo que calcula el número de línea donde debe inicializarse la lectura de los ríos.
- 140 : Inicialización del puntero.
- 150 : Leer número de ríos y vertiente.
- 160 : Llamada a la subrutina que lee y visualiza el menú con los nombres de los ríos.
- 170-180 : Entrada de la opción «río» y verificación.
- 190 : Algoritmo para calcular dónde debe inicializarse la lectura de los datos correspondientes al río elegido.
- 200 : Inicialización del puntero.
- 210 : Borrado pantalla.
- 220 : Llamada a la subrutina que lee y visualiza los datos correspondientes al dato elegido.



Estructura de datos del programa "Ríos".

- 230 : Entrada de la tecla «ENTER».
- 240-250 : Borrado pantalla y vuelta al menú principal (vertientes).
- 1000-1070 : Subrutina para definir la letra «Ñ».
- 1100-1130 : Subrutina de presentación del mensaje «ERROR».
- 1200-1290 : Subrutina para leer y visualizar los ríos.
- 1300-1410 : Subrutina para leer y visualizar los datos, (nacimientos, provincia, ...)
- 2000-2370 : Tabla de datos. ■

LECTURA DEL TECLADO Y TEMPORIZACIONES

INKEYS

Acceso al teclado

IN KEY \$



MODO **E**

OVER

Tipo de sentencia

Función de entrada.

Definición

La función «INKEY\$» permite leer un solo carácter introducido por teclado. Se diferencia básicamente de la sentencia «INPUT» en:

- INKEY\$ no produce *eco*, es decir, que no se visualiza el valor de la tecla pulsada.
- El dato tiene que estar preparado al ejecutarse la función «INKEY\$», es decir, que no espera, como en el caso de «INPUT», a que se teclee el dato: por tanto, si no está preparado se ejecuta la siguiente instrucción y la función retorna una cadena vacía (" ").

«INKEY\$» no tiene ningún argumento, por el contrario, forma parte de los argumentos de sentencias del tipo:

- LET.
- PRINT.
- IF... THEN...

Ejemplos:

```

10 REM *****
      INKEY$ 1
      *****
20 IF INKEY$="" THEN GO TO 20
30 PRINT INKEY$.
40 GO TO 10
    
```

- En el siguiente programa se visualiza el valor retornado por la función «INKEY\$» cuando se pulsa una tecla, es decir, cuando no es una cadena vacía. observe que «INKEY\$» diferencia entre mayúsculas (modo **C**), minúsculas (modo **L**) o aquellos símbolos que se pulsan conjuntamente con SYMBOL SHIFT.

```

10 REM *****
      INKEY$ 2
      *****
20 INPUT "Numero >>> ", numero
30 PRINT numero
40 PRINT #0, AT 1,0, "Quieres introducir mas (S/N)?"
50 IF INKEY$="" THEN GO TO 50
60 IF INKEY$="S" OR INKEY$="s" THEN GO TO 20
70 IF INKEY$="n" OR INKEY$="N" THEN STOP
80 GO TO 40
    
```

- En este caso, «INKEY\$» se utiliza para preguntar sobre una tecla determinada. Tiene la ventaja sobre la utilización de «INPUT», en que no es necesario pulsar «ENTER».

```

10 REM *****
      INKEY$ 3
      *****
    
```

```

20 PRINT AT 2,0, "MENU PRINCIPAL"
30 PRINT AT 3,0, "-----"
40 PRINT AT 8,10, "1 - EDITAR."
50 PRINT AT 10,10, "2 - LISTAR."
60 PRINT AT 12,10, "3 - BORRAR."
70 PRINT AT 14,10, "4 - ARCHIVAR."
80 PRINT AT 16,10, "5 - INSERTAR."
90 PRINT #0, AT 1,0, "Elija (a opcion) 1 - 5"
100 LET A$=INKEY$
110 IF A$="1" THEN GO TO 1000
120 IF A$="2" THEN GO TO SUB 2000
130 IF A$="3" THEN GO TO 1500
140 IF A$="4" THEN GO TO 1750
150 IF A$="5" THEN GO TO SUB 2100
160 GO TO 100
    
```

- Quando el valor retornado por «INKEY\$», se utiliza en sentencias posteriores a la pulsación de la tecla, conviene asignar este valor a una variable de cadena.

PAUSE

Acceso al teclado

PI



MODO **K**

INVERSE

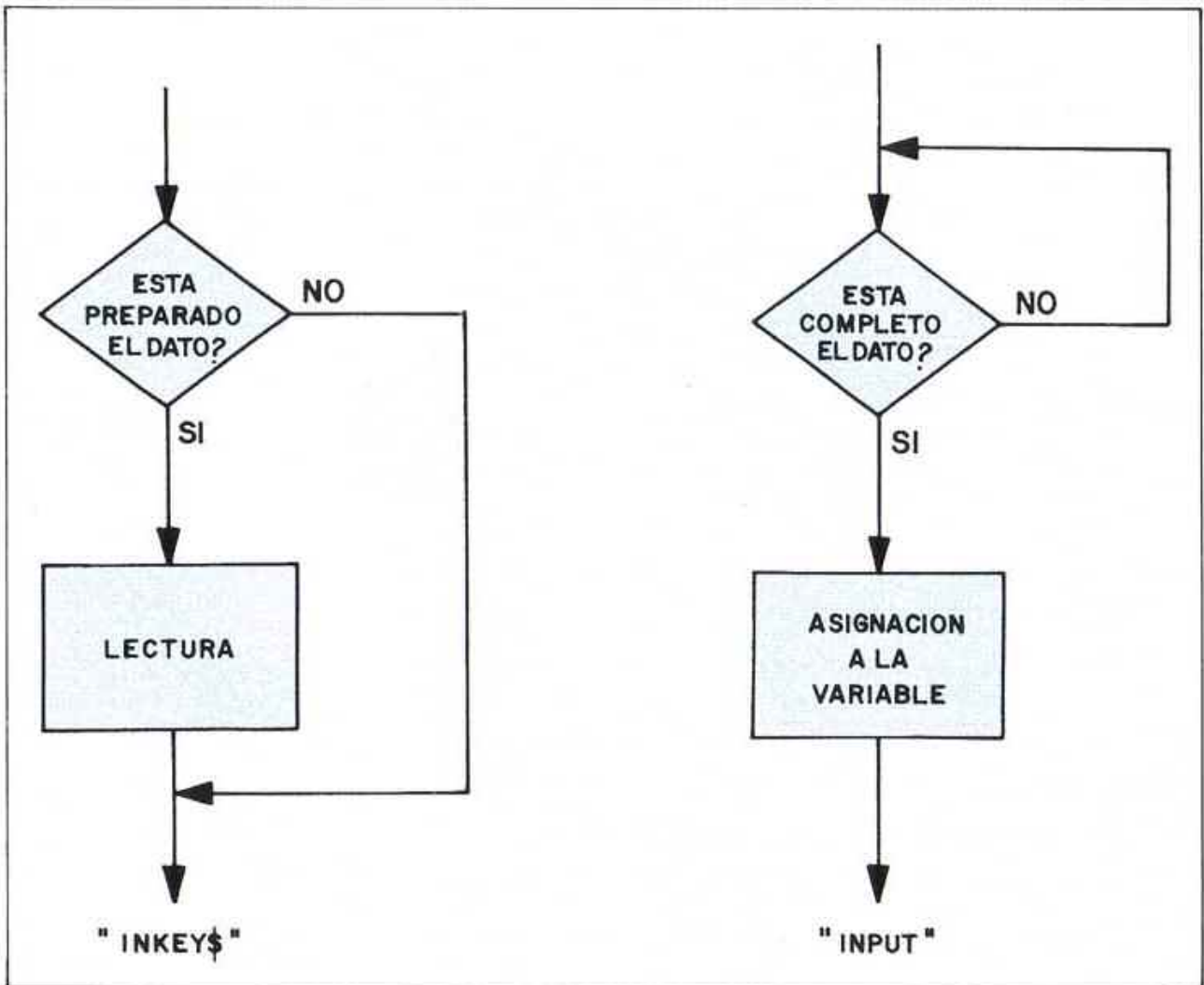
Tipo de sentencia

Comando de programación.

Definición

Esta sentencia suspende la ejecución de un programa durante un tiempo determinado, es decir, *temporiza* o hace una *pausa*.

El efecto que produce es similar al proporcionado por las sentencias:



Diferencias entre "INKEY\$" e "INPUT".

```
10 FOR x = 1 TO tiempo
20 NEXT x
```

La estructura de este comando es:

| SENTENCIA | ARGUMENTO |
|-----------|--------------------|
| PAUSE | expresión numérica |

Ejemplos:

- PAUSE 10.
- PAUSE 950.
- PAUSE 30 + 5.

Si se introduce un número fraccionario, éste se redondea al valor entero más cercano. El rango de valores está comprendido entre 0 y 65535, cuando se introduce uno fuera de margen aparece el mensaje.

B Integer out of range

La realización que existe entre el número del argumento y el tiempo de temporización, en segundos es:

$$\text{tiempo} = \frac{n}{50}$$

por lo tanto «PAUSE 100», detiene la ejecución del programa durante «2» segundos. Dividiendo entre 50 el valor máximo (65535) obtenemos el mayor tiempo de temporización:

$$65535/50 = 1310.7 \text{ sg.}$$

21 min. 50.7 sg.

ATENCIÓN

Si durante la ejecución de una sentencia «PAUSE» se presiona una tecla, la temporización se interrumpe y continúa el programa en la siguiente instrucción.

El valor «0», asignado a «PAUSE», provoca una temporización indefinida hasta que se pulsa una tecla.

Puede utilizarse una variable del tipo numérico como argumento de esta sentencia.

Ejemplo:

```
10 REM *****
   PAUSA VARIABLE *****
```



```

15 BORDER 4
20 INPUT "Temporizacion (1-10)
  " tiempo
30 IF tiempo<1 OR tiempo>10 TH
EN GO TO 20
40 LET pausa=tiempo*50
50 BORDER 1
60 PAUSE pausa
70 BORDER 4
80 PRINT #0,AT 1,0;"Quiere con
  tinuar (S/N)
90 PAUSE 0
100 LET as=INKEY$
110 IF as="s" OR as="S" THEN GO
  TO 20
120 IF as="n" OR as="N" THEN ST
  OP
130 GO TO 90

```

El valor asignado a la variable «tiempo» se multiplica por 50 para conseguir la temporización deseada, el resultado es asignado a la variable «pausa», utilizada como argumento.

En la línea 90 «PAUSE 0» detiene la ejecución del programa hasta que se pulse la tecla «S» o «N».

PROGRAMAS

El programa número «1» simula una máquina de escribir. los controles son:

- SYMBOL SHIFT + Q, borra el carácter anterior.
- SYMBOL SHIFT + W, provoca un retorno de carro.
- SYMBOL SHIFT + E, termina la edición de la página.

Una vez editada la página, puede sacarse una copia por impresora seleccionando la opción «L» o grabarse en cinta con «G». La opción «C» permite editar una nueva página o terminar.

La estructura es la siguiente:

- 10 : Comentario con el nombre del programa.
- 20 : Asignación del color rojo para borde, azul para el fondo y blanco para los caracteres.

- 30 : Inicialización del cursor en la posición 0, 0.
 - 40 : Visualización del cursor. Se utiliza uno de los gráficos predefinidos.
 - 50 : Pausa hasta que se pulsa una tecla.
 - 60-100 : Verificación de la tecla pulsada. Su código ASCII debe estar comprendido entre el número 32 (espacio) y el 122 (z) o ser uno de los códigos de control del programa:
 - " < =" (SYMBOL SHIFT + Q)
 - " < > " (SYMBOL SHIFT + W)
 - " > =" (SYMBOL SHIFT + E)
 - 110 : Visualización de la tecla pulsada.
 - 120-140 : Incremento de la posición del cursor y comprobación del final de línea y final de página.
 - 150 : Salto a la línea que visualiza el cursor.
 - 1000 : Empieza la rutina que borra el carácter situado a la izquierda del cursor (control = SYMBOL SHIFT + Q).
 - 1005 : Borra el cursor.
 - 1010-1030 : Decremento de la posición del cursor y comprobación del principio de línea y principio de página.
 - 1040 : Salto a la línea que visualiza el cursor.
 - 1100 : Comienzo de la rutina que hace saltar el cursor al comienzo de la siguiente línea, simulando el retorno de carro de una máquina de escribir (control = SYMBOL SHIFT + W).
 - 1105 : Borrado del cursor.
 - 1110 : Inicialización de la nueva posición del cursor.
 - 1120 : Salto a la línea que comprueba el final de página y línea.
 - 1205 : Borrado del cursor.
 - 1210 : Visualización de las opciones:
 - L - Listar
 - G - Grabar
 - C - Continuar
 - 1220-1250 : Comprobación de la opción elegida.
 - 1260-1300 : Imprimir el contenido de la pantalla. Se utiliza la sentencia «COPY».
 - 1310 : Salto a la visualización de las opciones.
 - 1350-1380 : Grabar en cinta el contenido de la pantalla. Debe introducirse previamente el nombre que deseamos asignarle. Se utiliza como argumento de «SAVE» la palabra clave «SCREEN\$».
 - 1390 : Salto a la visualización de opciones.
 - 1500-1550 : Decisión para editar una nueva página o no.
- El programa número «2» permite desplazar, con ayuda de unas teclas utilizadas como cursor, un asterisco (*) a través de la pantalla, que va dejando un rastro de puntos (.) por donde va pasando.

PROGRAMA 1

```

10 REM *****
*          *
*  CURSO/BASIC *
*          *
*****
*          *
*    MAQUINA   *
*          *
*****

20 BORDER 2: PAPER 1: INK 7: C
L5
30 LET x=0: LET y=x
40 PRINT AT y,x;"|"
50 PAUSE 0
60 LET a$=INKEY$
70 IF a$="<" THEN GO TO 1000
80 IF a$=">" THEN GO TO 1100
90 IF a$="=" THEN GO TO 1200
95 IF (a$>=" " AND a$<="z") TH
EN GO TO 110
100 GO TO 50
110 PRINT AT y,x;a$
120 LET x=x+1
130 IF x=32 THEN LET x=0: LET y
=y+1
140 IF y=22 THEN GO TO 1210
150 GO TO 40
1000 REM
*****
*          *
*  BORRADO    *
*          *
*****

1005 PRINT AT y,x;" "
1010 LET x=x-1
1020 IF x=-1 THEN LET x=31: LET
y=y-1
1030 IF y=-1 AND x=31 THEN LET y
=0: LET x=0
1040 GO TO 40
1100 REM
*****
*          *
*  RETORNO   *
*          *
*****

1105 PRINT AT y,x;" "
1110 LET x=0: LET y=y+1
1120 GO TO 140
1200 REM
*****
*          *
*  ALMACENAR *
*          *
*****

```

```

1205 PRINT AT y,x;" "
1210 PRINT #0;AT 1,1;"L-Lista /
G-Graba / C-Continua"
1220 IF INKEY$="" THEN GO TO 122
0
1230 IF INKEY$="l" OR INKEY$="L"
THEN GO TO 1260
1240 IF INKEY$="g" OR INKEY$="G"
THEN GO TO 1350
1245 IF INKEY$="c" OR INKEY$="C"
THEN GO TO 1500
1250 GO TO 1220
1260 REM
*****
*          *
*  IMPRESORA  *
*          *
*****

1262 INPUT 0
1265 PRINT #0;AT 0,2;"Conecte la
impresora y pulse"
1270 PRINT #0;AT 1,10;"una tecla"

1280 PAUSE 0
1285 INPUT 0
1290 COPY
1300 PRINT #0;AT 1,6;"Impresion
terminada": PAUSE 100
1310 GO TO 1210
1350 REM
*****
*          *
*  GRABACION  *
*          *
*****

1360 INPUT "Nombre (max 10 carac
teres) >>> "; LINE n$
1370 SAVE n$SCREEN$
1380 PRINT #0;AT 1,6;"Grabacion
terminada": PAUSE 100
1390 GO TO 1210
1500 REM
*****
*          *
*  CONTINUACION? *
*          *
*****

1510 INPUT 0
1520 PRINT #0;AT 1,0;"Quiere edi
tar otra pagina (S/N)"
1530 IF INKEY$="s" OR INKEY$="S"
THEN GO TO 10
1540 IF INKEY$="n" OR INKEY$="N"
THEN CLS: STOP
1550 GO TO 1530

```

Tenemos la posibilidad de elegir las teclas que vamos a utilizar como cursores. El programa tiene asignadas por defecto, es decir, si no se eligen otras, las siguientes:

- 7 - ARRIBA
- 6 - ABAJO
- 8 - DERECHA
- 9 - IZQUIERDA

La velocidad es otro de los parámetros que se puede elegir, su valor varía entre «1» (rápida) y «9» (lenta).

Aparte de las teclas utilizadas como cursor, existen otras dos que en combinación con SYMBOL SHIFT realizan una determinada función.

SYMBOL SHIFT + B, borra la pantalla.

SYMBOL SHIFT + C, fin del programa.

La estructura del programa es la siguiente:

1Ø : Comentario con el nombre del programa.

2Ø

: Asignación del color rojo para el borde, verde para el fondo y negro para los caracteres.

3Ø

: Llamada a la subrutina que presenta el menú con los cursores.

4Ø-9Ø

: Verificación de la opción elegida.

1ØØ

: Asignación de los valores por defecto.

PROGRAMA 2

```

10 REM *****
*
* CURSO/BASIC *
*
* *****
* MOVIMIENTO *
*
* *****

Ls 20 BORDER 2: PAPER 4: INK 0: C
25 REM *****
*
* CAMBIO CURSOR *
*
* *****

30 GO SUB 1000
40 PRINT #0; AT 1,0; "Quiere otr
os cursores (S/N) >>>"
50 IF INKEY$="" THEN GO TO 50
60 LET k$=INKEY$
70 IF k$="N" OR k$="n" THEN GO
TO 100
80 IF k$="S" OR k$="s" THEN GO
SUB 1100: GO TO 110
90 GO TO 50
100 LET a$="7": LET b$="6": LET
d$="8": LET i$="5"
110 PAUSE 50
115 GO SUB 1300
120 CLS
125 REM *****
*
* RECUADRO *
*
* *****

130 FOR n=0 TO 31
140 PRINT AT 0,n; "■"
150 NEXT n
160 FOR n=1 TO 20
170 PRINT AT n,0; "■"; AT n,31; "■"
180 NEXT n
190 FOR n=0 TO 31
200 PRINT AT 21,n; "■"
210 NEXT n
220 REM *****
*
* MOVIMIENTO *
*
* *****

230 LET posx=16
240 LET posy=11
250 GO SUB 1500
260 IF INKEY$="" THEN GO TO 260
270 IF INKEY$=a$ THEN PRINT AT
posy,posx; ".": LET posy=posy-1:
GO SUB 1500
280 IF INKEY$=b$ THEN PRINT AT
posy,posx; ".": LET posy=posy+1:
GO SUB 1500
290 IF INKEY$=d$ THEN PRINT AT
posy,posx; ".": LET posx=posx+1:
GO SUB 1500
300 IF INKEY$=i$ THEN PRINT AT
posy,posx; ".": LET posx=posx-1:
GO SUB 1500
310 IF INKEY$="*" THEN GO TO 34
0
320 IF INKEY$="?" THEN STOP
330 GO TO 260
340 GO SUB 1600
350 GO TO 230
1000 REM *****
*
* MOVIMIENTO *
*
* *****

```

```

1010 PRINT AT 2,0; "MOVIMIENTO"
1020 PRINT AT 6,9; "7 - ARRIBA."
1030 PRINT AT 9,9; "6 - ABAJO."
1040 PRINT AT 12,9; "8 - DERECHA."
1050 PRINT AT 15,9; "5 - IZQUIERD
A."
1060 PRINT AT 19,0; " "
1070 RETURN
1100 REM *****
*
* CURSORES *
*
* *****

1110 INPUT "Arriba >>> "; LINE a
$
1112 IF a$="" THEN GO TO 1110
1115 LET a$=a$(1)
1120 IF (a$>="0" AND a$<="9") OR
(a$>="a" AND a$<="z") OR (a$>="
A" AND a$<="Z") THEN PRINT AT 6,
9;a$: GO TO 1140
1130 GO TO 1110
1140 INPUT "Abajo >>> "; LINE b$
1142 IF b$="" THEN GO TO 1140
1145 LET b$=b$(1)
1150 IF ((b$>="0" AND b$<="9") O
R (b$>="a" AND b$<="z") OR (b$>="
A" AND b$<="Z")) AND b$<>a$ THE
N PRINT AT 9,9;b$: GO TO 1170
1160 GO TO 1140
1170 INPUT "Derecha >>> "; LINE
d$
1172 IF d$="" THEN GO TO 1170
1175 LET d$=d$(1)
1180 IF ((d$>="0" AND d$<="9") O
R (d$>="a" AND d$<="z") OR (d$>="
A" AND d$<="Z")) AND d$<>a$ AND
d$<>b$ THEN PRINT AT 12,9;d$: G
O TO 1200
1190 GO TO 1170
1200 INPUT "Izquierda >>> "; LIN
E i$
1202 IF i$="" THEN GO TO 1200
1205 LET i$=i$(1)
1210 IF ((i$>="0" AND i$<="9") O
R (i$>="a" AND i$<="z") OR (i$>="
A" AND i$<="Z")) AND i$<>a$ AND
i$<>b$ AND i$<>d$ THEN PRINT AT
15,9;i$: RETURN
1220 GO TO 1200
1300 REM *****
*
* VELOCIDAD *
*
* *****

1310 CLS
1320 PRINT AT 2,0; "VELOCIDAD"
1330 PRINT AT 6,9; "1 - RAPIDA."
1340 PRINT AT 9,9; "-----"
1350 PRINT AT 12,9; "-----"
1360 PRINT AT 15,9; "9 - LENTA."
1370 PRINT AT 19,0; " "
1380 INPUT "Velocidad >>> "; vel
ocidad
1390 IF velocidad<1 OR velocidad
>9 THEN GO TO 1380
1410 RETURN
1500 REM *****
*
* VERIFICACION *
*
* *****

```

```

1510 IF posx < 1 THEN LET posx = 1
1520 IF posx > 30 THEN LET posx = 30
1530 IF posy < 1 THEN LET posy = 1
1540 IF posy > 20 THEN LET posy = 20
1550 PRINT AT posy, posx; "*"
1560 FOR n = 1 TO velocidad
1570 NEXT n
1580 RETURN
1600 REM

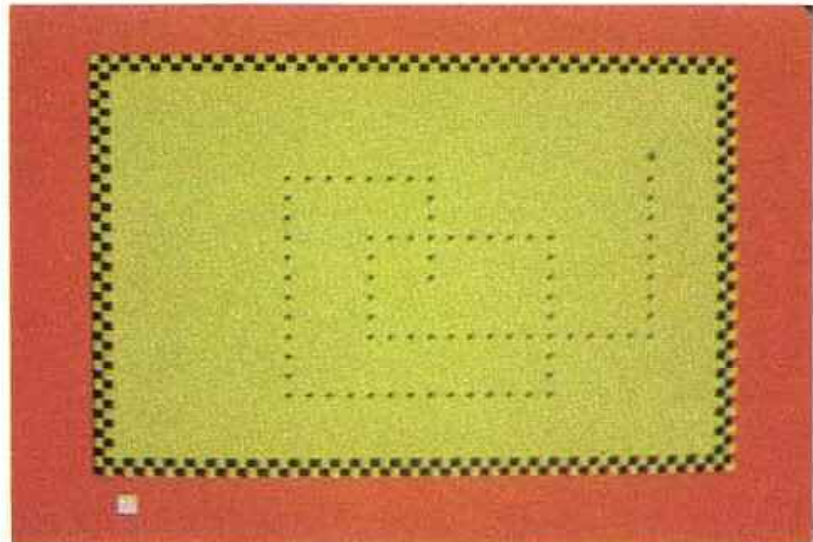
```

```

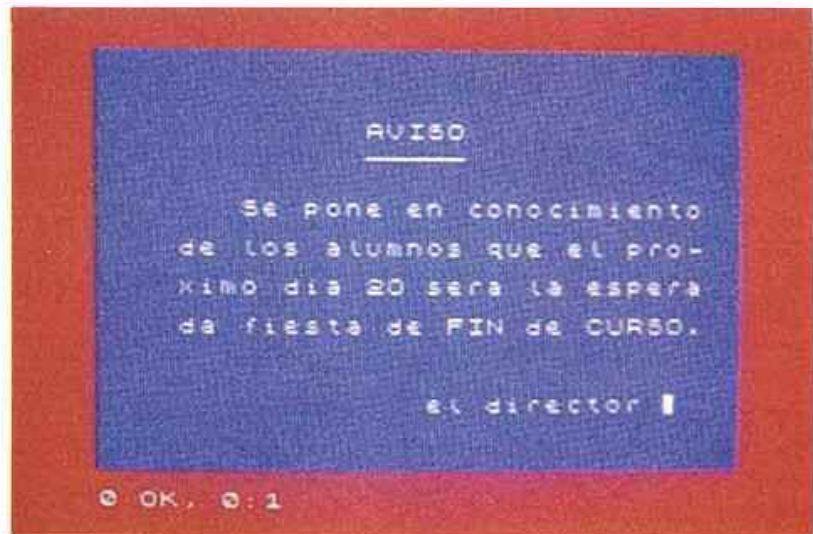
*****
* BORRADO *
*****
1610 FOR n = 1 TO 20
1620 PRINT AT n, 1; "
1630 NEXT n
1640 RETURN

```

- 110 : Retardo de un segundo aproximadamente. Llamada a la subrutina que pregunta la velocidad.
- 120 : Borrado de pantalla.
- 130-210 : Rutina que dibuja, con ayuda de los gráficos predefinidos, un recuadro.
- 230-240 : Inicialización de las coordenadas del asterisco.
- 250 : Llamada a la subrutina que dibuja el asterisco.
- 260-300 : Determinación de la tecla pulsada, cálculo de la nueva posición y llamada a la subrutina de visualización.
- 310 : Si la tecla pulsada es el asterisco (*), la opción de borrar es la elegida.
- 320 : Si la tecla pulsada es la interrogación (?), el programa termina su ejecución.
- 340 : Llamada a la subrutina de borrado.
- 1000-1070 : Subrutina que visualiza el menú con la asignación de cursores inicial.
- 1100-1220 : Subrutina utilizada para modificar la asignación inicial de cursores.
- 1300-1410 : Subrutina para la



Programa «Movimiento».



Programa «Máquina».

- 1500-1580 : Verificación de que las coordenadas del asterisco se encuentran dentro del recuadro, visualización

- del mismo y temporización variable, dependiente de la velocidad elegida.
- 1600-1640 : Borrado de la parte interior del recuadro. ■

FUNCIONES

Existen una serie de funciones definidas dentro del BASIC, que pueden clasificarse en:

- Numéricas.
- De cadena.

Aparte de estas funciones, ya fijas, el usuario puede definir las suyas propias.

En general, una función proporciona un resultado después de haber efectuado unos cálculos con un dato denominado *parámetro*; éste forma parte del argumento de la instrucción. Para cada valor, la función retorna un resultado distinto.

Aunque no es necesario incluir los parámetros de la función entre paréntesis, es conveniente hacerlo, en algunos casos, por motivos de claridad. También hay que tener en cuenta que las funciones tienen mayor prioridad que las operaciones. Las funciones no se utilizan directamente como comandos, por el contrario necesitan ir acompañadas de las palabras clave «PRINT» (visualización), «LET» (asignación), «IF... THEN», etc.

Funciones numéricas

Estas pueden clasificarse en los siguientes grupos:

- Matemáticas.
- Trigonométricas.
- Exponenciales.
- Logarítmicas.
- Aleatorias.
- Definidas.

ABS

Acceso al teclado

ABS



MODO E

}

Definición

Esta función retorna el *valor absoluto* del argumento.

Ejemplos:

- LET a = ABS 30.
- PRINT ABS (-100 + 2)
- IF ABS valor <> 30 THEN...
- FOR n = 1 TO ABS X

El valor absoluto de una expresión se calcula despreciando su signo, por tanto, el resultado de las siguientes instrucciones será el mismo.

```
PRINT ABS 3542
PRINT ABS -3542
PRINT ABS +3542
```

En el siguiente programa se visualiza el valor absoluto de cualquier número comprendido entre 99999999 y -99999999.

```
10 REM *****
      VALOR ABSOLUTO
      "ABS"
      *****
20 INPUT "Numero >>> "; numero
```

```
30 IF ABS numero > 99999999 THEN
GO TO 20
40 IF numero = 0 THEN LET as = "Si
n signo" GO TO 70
50 IF numero > 0 THEN LET as = "Po
sitivo" GO TO 70
60 LET as = "Negativo"
70 LET absoluto = ABS numero
80 FOR n = 5 TO 1 STEP 2
82 PRINT AT n, 10; "
"
84 NEXT n
90 PRINT AT 5, 0; "Numero .....
"
100 PRINT AT 7, 0; "Signo .....
"
110 PRINT AT 9, 0; "Absoluto ....
"
120 GO TO 20
```

En la línea 30 se ha utilizado la función «ABS» para averiguar si el número estaba comprendido dentro del rango; si no, hubiéramos tenido que hacerlo de la forma:

```
30 IF numero < -99999999 OR
numero > 99999999 THEN
GOTO 20
```

INT

Acceso al teclado

INT



MODO E

VERIFY

Definición

La función «INT» retorna el *valor entero* de una expresión:

Ejemplos:

- LET X = INT — Y
- PRINT INT (X + 3)
- IF INT n = n THEN ...
- FOR Z = INT t TO INT j

«INT» redondea *por defecto* el argumento, despreciando sus decimales. Redondear por defecto significa asumir el valor entero *inmediato inferior*, por tanto, se obtendrá el mismo resultado con cualquiera de las dos sentencias siguientes:

```
PRINT INT 3.00001
PRINT INT 3.99999
```

en ambos casos el resultado es «3».

Con los números negativos ocurre una cosa curiosa, ya que al efectuar el redondeo por defecto, aumenta su valor absoluto:

Ejemplo:

```
PRINT INT -3.00001
PRINT INT -3.99999
```

el resultado de ambas funciones es «-4».

El siguiente programa calcula si el número entero introducido por el teclado es *par* o *impar*.

```
10 REM *****
   : PAR O IMPAR :
   : ***** :
20 INPUT "Numero >>> "; numero
30 IF ABS numero > 99999999 OR INT numero < numero THEN GO TO 20
40 LET divisor=numero/2
50 LET entero=INT divisor
60 IF divisor=entero THEN LET
   : as="par" : GO TO 80
70 LET as="impar"
80 PRINT AT 5,0; "
90 PRINT AT 5,0; "El numero ";n
   : umero;" es ";as
100 GO TO 20
```

Con ayuda de la función «INT» se averigua en la línea 30 si la variable «número» tiene parte fraccionaria.

En el siguiente ejemplo se visualiza la parte entera y fraccionaria del número positivo que se introduce por teclado.

las pequeñas diferencias que pueden existir, entre las partes fraccionarias son debidas a los cálculos.

```
10 REM *****
   : PARTE ENTERA :
   : "INT" :
   : ***** :
20 INPUT "Numero >>> "; numero
30 IF numero < 0 OR numero > 99999
   : 999 THEN GO TO 20
40 LET entero=INT numero
50 LET fraccionario=numero-ent
   : ero
60 FOR n=5 TO 9 STEP 2
   : 70 PRINT AT n,16; "
   : "
80 NEXT n
90 PRINT AT 5,0; "Numero .....
   : ".numero
100 PRINT AT 7,0; "Parte entera
   : ".entero
110 PRINT AT 9,0; "Parte decimal
   : ".fraccionario
120 GO TO 20
```

SGN

Acceso al teclado

SGN



Definición

La función «SGN» nos indica qué signo tiene la expresión que estamos evaluando. Los posibles valores que puede retornar son:

- 1 : Si es positiva.
- 1 : Si es negativa.
- 0 : Si es 0

Ejemplos:

- PRINT SGN -30
- LET a = SGN C
- PRINT SGN (7 * (-5))
- LET X = 7 * SGN Z

El siguiente programa nos indica si el número introducido por teclado es positivo o negativo.

```
10 REM *****
   : SIGNO :
   : "SGN" :
   : ***** :
20 LET positivo=1
30 LET negativo=-1
40 LET cero=0
50 INPUT "Numero >>> "; numero
60 IF ABS numero > 99999999 THEN
   : GO TO 80
70 IF SGN numero=positivo THEN
   : LET as="es positivo" : GO TO 100
80 IF SGN numero=negativo THEN
   : LET as="es negativo" : GO TO 100
90 LET as="no tiene signo"
100 PRINT AT 5,0; "
110 PRINT AT 5,0; "El numero ";n
   : umbero;" es ";as
120 GO TO 50
```

SQR

Acceso al teclado

SQR



Definición

La función «SQR» calcula la raíz cuadrada del argumento.

Ejemplos:

- PRINT SQR 144
- LET r = SQR 625 + 13
- PRINT SQR raíz
- LET n = SQR (25 + a)

«SQR» sólo calcula raíces de tipo *real*, si se pretende calcular una *imaginaria*

($\sqrt{-144}$) se visualizará el mensaje de error:

A Invalid argument

Ejemplo:

— PRINT SQR -144

Cuando se evalúa una variable, ésta puede tomar un valor negativo, para asegurarnos que al calcular su raíz no nos dé error, podemos utilizar la función «ABS».

1Ø LET raíz = -144
2Ø PRINT SQR (ABS raíz)

Aunque el Spectrum sólo tiene una función de *radicación* (SQR), cuyo índice es 2, se pueden obtener raíces de cualquier orden, para ello hay que basarse en la siguiente igualdad:

$$\sqrt[r]{r} = r^{\frac{1}{r}}$$

donde «r» es el radicando e «i» el índice.

Ejemplo:

— Raíz cúbica de 27
PRINT 27 ↑ (1/3)
— Raíz quinta de 32
PRINT 32 ↑ (1/5)

BIN

Acceso al teclado

BIN



MODO E

BRIGHT

Definición

Permite la representación de los números en notación binaria, para una mayor aclaración conviene consultar el capítulo «CONSTANTES Y VARIABLES» (Pág. 30).

Ejemplos:

— LET a = BIN 1001
— PRINT -BIN 11011*2
— LET c = BIN 110/na
— PRINT SQR BIN 1111

El mayor número que se puede representar en este tipo de notación, es «65535».

En la página 25 se presenta el programa «CODEBIN» que realiza la transformación inversa, es decir, pasar de notación decimal a binaria.

PI

Acceso al teclado

PI



MODO E

INVERSE

Definición

«PI» es el nombre de una letra griega (π) usada como constante en multitud de cálculos matemáticos. Su valor aproximado es:

$$\pi \simeq 3,14159265\dots$$

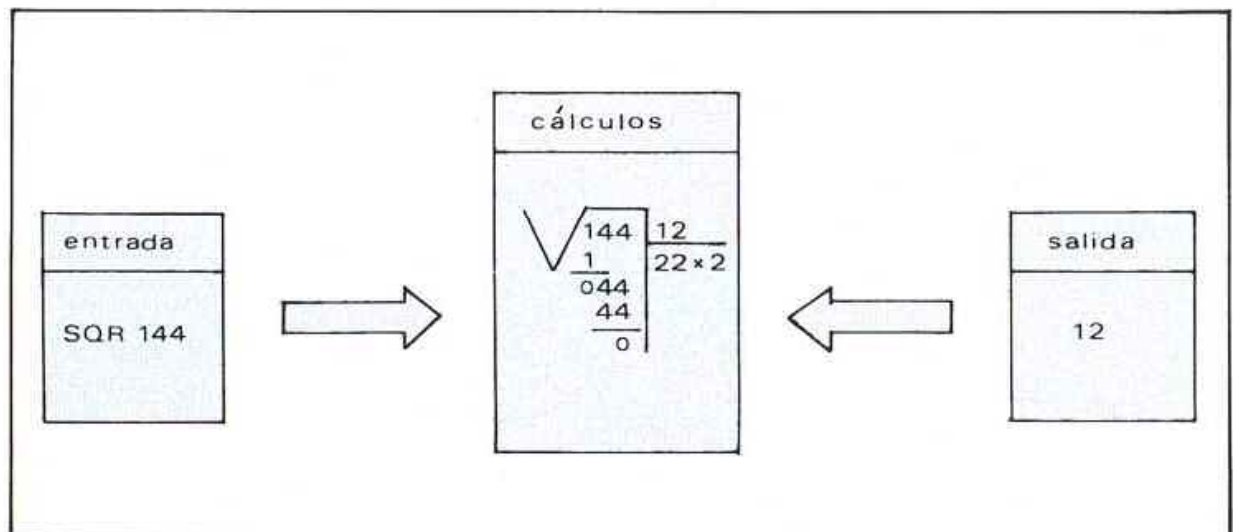
Ejemplos:

a) Cálculo de la longitud de una circunferencia de radio 6.

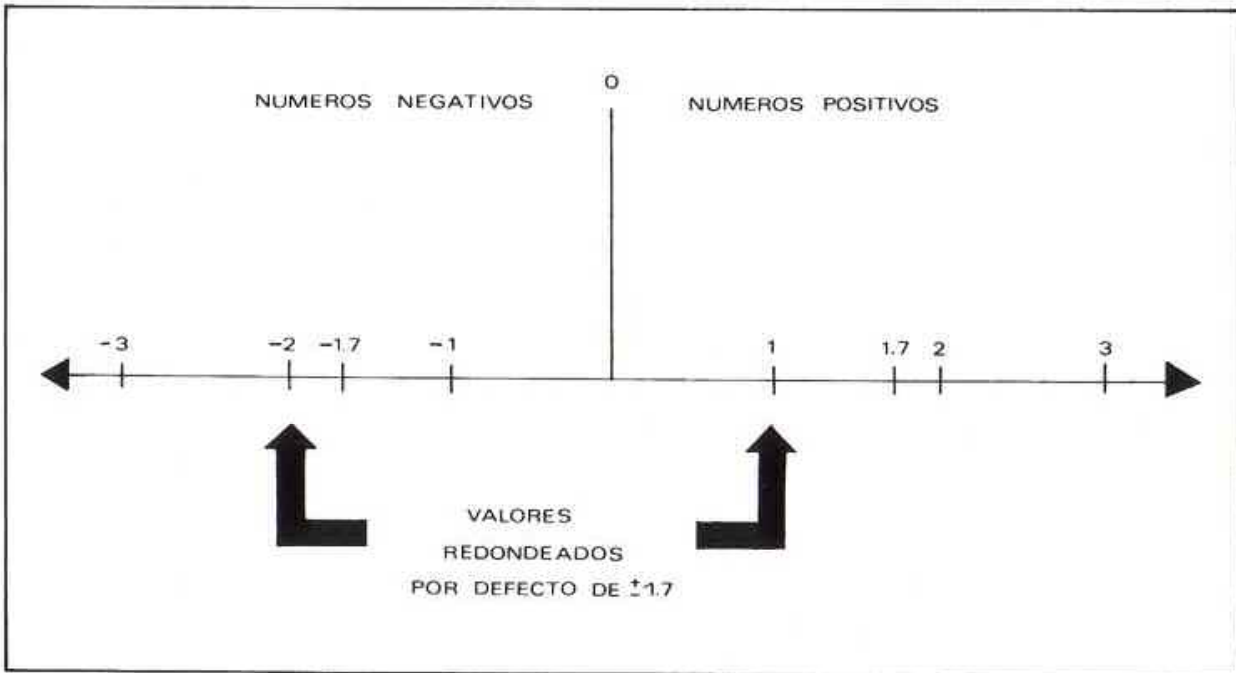
1Ø LET radio = 6
2Ø PRINT 2 * PI * radio

b) Cálculo de la impedancia que presenta una bocina de 0,5 Henrios a la frecuencia de 12 KHz.

1Ø LET L = 0,5
2Ø LET F = 12000



Ejemplo «Función».



Función «INT».

```
30 LET Z = 2 * PI * F * L
40 PRINT Z
```

El radián

Es una unidad de medida de ángulos. Se puede definir como el ángulo cuyo arco tiene la misma longitud que el radio de la circunferencia (ver figura).

La equivalencia entre el radián y los grados sexagesimales es:

$$\begin{aligned} 90^\circ &= \pi/2 \text{ radianes} \\ 180^\circ &= \pi \text{ radianes} \\ 270^\circ &= \pi 3/2 \text{ radianes} \\ 360^\circ &= 2\pi \text{ radianes} \end{aligned}$$

por tanto:

$$1 \text{ radián} = \frac{180}{\pi}$$

Para calcular en el Spectrum el seno, coseno o tangente de un ángulo, este deberá ser expresado en radianes. Para transformar grados a radia-

nes deberá usarse la siguiente fórmula:

$$\text{radianes} = \frac{\text{grados} \cdot \pi}{180}$$

y para transformar a la inversa:

$$\text{grados} = \frac{\text{radianes} \cdot 180}{\pi}$$

El siguiente programa transforma grados, minutos y segundos en radianes:

```
10 REM *****
   : RADIANTES
   : *****
20 INPUT "Grados >>> ";grados
30 IF grados<0 OR grados>360 THEN GO TO 20
35 CLS
40 PRINT "Grados ..... ";grados
50 INPUT "Minutos >>> ";minutos
60 IF minutos<0 OR minutos>60 THEN GO TO 50
70 PRINT "Minutos ..... ";minutos
80 INPUT "Segundos >>> ";segundos
90 IF segundos<0 OR segundos>60 THEN GO TO 80
100 PRINT "Segundos ..... ";segundos
110 LET total=grados+(minutos/60)+(segundos/3600)
120 IF total>360 THEN CLS : GO TO 20
130 LET radianes=total*PI/180
140 PRINT "Radianes ..... ";radianes
150 GO TO 20
```

Funciones trigonométricas

SIN

Acceso al teclado

SIN



MODO E

ASN

Definición

«SIN» calcula el seno de un ángulo expresado en radianes.

Ejemplos:

- PRINT SIN (90 * PI/180)
- LET C = SIN 1
- PRINT SIN (270 * PI/180)
- LET C = SIN ángulo

En un triángulo rectángulo, el seno de un ángulo es la razón que existe entre el cateto opuesto y la hipotenusa.

Para un ángulo comprendido entre «0» y «180» grados el valor del seno es positivo, y negativo entre «180» y «360».

COS

Acceso al teclado

COS



ACS

Definición

La función «COS» calcula el *coseno* de un ángulo, éste debe estar expresado en radianes.

Ejemplo:

- LET n = COS 2
- PRINT COS (17 * PI/180)
- LET valor = COS total
- PRINT COS (320 * PI/180)

El coseno de uno de los ángulos de un triángulo rectángulo es la razón que hay entre el cateto adyacente y la hipotenusa.

El valor del seno es positivo para un ángulo comprendido entre «0» y «90» o entre «270» y «360». Es negativo entre «90» y «270».

TAN

Acceso al teclado

TAN



ATN

152 MICROBASIC

Definición

«TAN» retorna la *tangente* de un ángulo expresado en radianes.

Ejemplos:

- PRINT TAN (45 * PI/180)
- LET C = TAN alfa
- PRINT TAN 1
- LET d = TAN (beta + 2)

La tangente de un ángulo es la razón que hay entre el cateto opuesto y el cateto adyacente, de un triángulo rectángulo.

ASN

Acceso al teclado

SIN



ASN

Definición

La función «ASN» calcula el *arcoseno*, es decir, el valor de un ángulo a partir de su seno. El valor retornado está expresado en radianes.

Ejemplos:

- LET a = ASN 0,5
- PRINT 180/PI * ASN 1
- LET C = ASN (alfa) * 180/PI
- PRINT ASN 0,7

El valor del argumento debe estar comprendido entre +1 y -1, de lo contrario, se visualizará el mensaje de error.

A Invalid argument

ACS

Acceso al teclado

COS



ACS

Definición

«ACS» es la función que retorna el *arcoseno*, ángulo calculado a partir del coseno. Las unidades son expresadas en radianes.

Ejemplos:

- LET d = ACS 0,9
- PRINT ACS (0,35) * 180/PI
- LET K = ACS beta
- PRINT ACS alfa

Al igual que la función «ASN», el argumento de «ACS» debe estar comprendido dentro del rango de +1 y -1.

ATN

Acceso al teclado

TAN



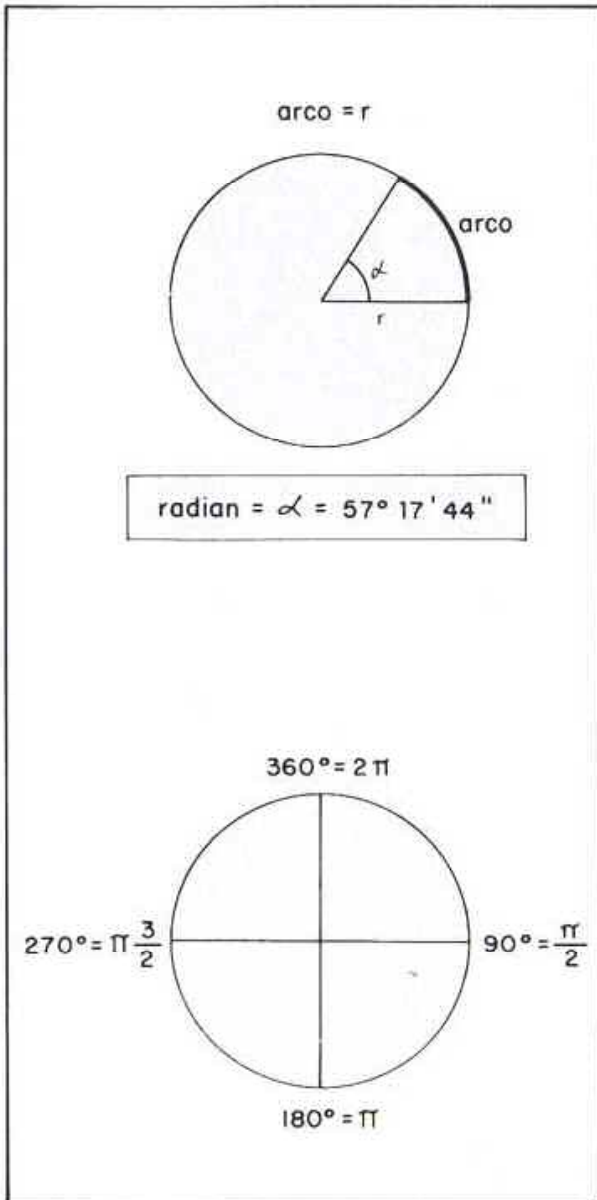
ATN

Definición

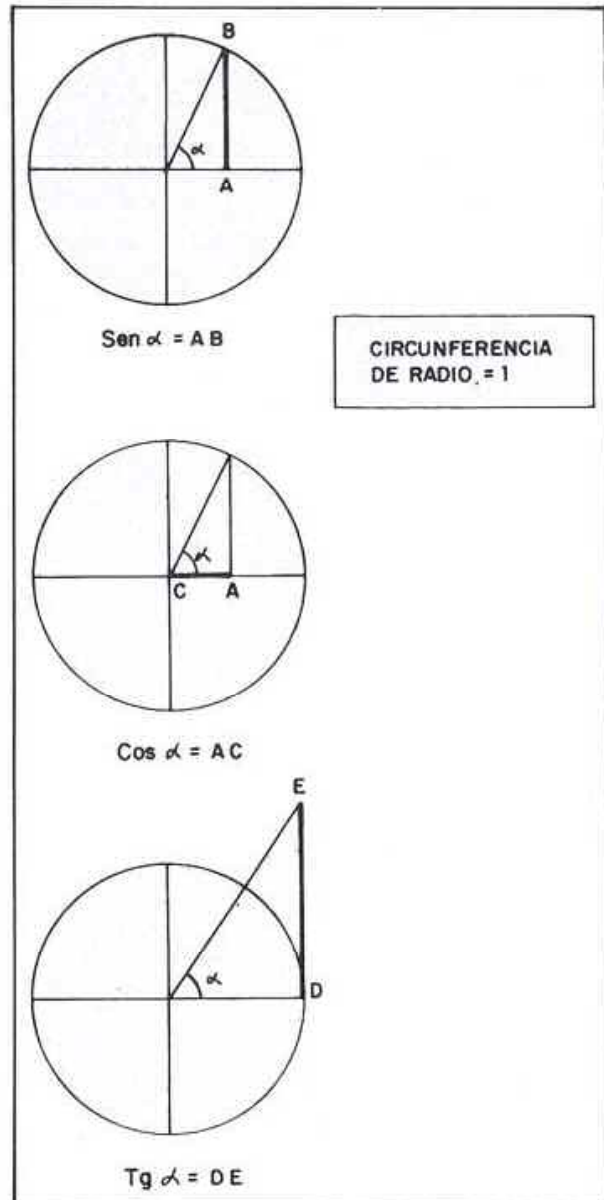
La función «ATN» calcula el *arcotangente*, es decir, el valor de un ángulo en radianes, a partir de su tangente.

Ejemplo:

- LET f = ATN gamma
- PRINT ATN 10
- LET C = 180/PI * ATN n
- PRINT ATN (10 + alfa)



Radianes



Funciones trigonométricas

Aplicación de la trigonometría

Los programas que a continuación se presentan, son una pequeña muestra de las múltiples aplicaciones de la trigonometría.

a) Calcular la altura que alcanza una escalera apoyada en una pared, conociendo su longitud y el ángulo que forma con el suelo.

```
10 REM *****
  * ESCALERA *
  *****
```

```
20 BORDER 4: PAPER 4: INK 0: C
LS
30 INPUT "Longitud escalera >>" longitud
32 CLS
40 IF longitud<1 OR longitud>3
0 THEN GO TO 30
42 PRINT "Longitud .....",longitud
50 INPUT "Inclinacion (grados) >>" grados
60 IF grados<1 OR grados>89 TH
EN GO TO 50
62 PRINT "Inclinacion ...",grados
70 GO SUB 1000
80 LET altura=longitud*SIN rad
90 PRINT "Altura .....",alt
ura
100 GO TO 30
1000 REM *****
1010 LET radianes=grados*PI/180
1020 RETURN
```

b) Calcular la altura de una torre, conociendo la distancia que nos separa de ella y la visual hasta su parte más alta.

```
10 REM *****
  * TORRE *
  *****
20 BORDER 4: PAPER 4: INK 0: C:LS
30 INPUT "Distancia (max. 500) >>" distancia
32 CLS
40 IF distancia<1 OR distancia>500 THEN GO TO 30
42 PRINT "Distancia .....",distancia
50 INPUT "Visual (grados) >>" grados
60 IF grados<1 OR grados>89 TH
EN GO TO 50
62 PRINT "Visual .....",grados
70 GO SUB 1000
80 LET altura=distancia*TAN rad
90 PRINT "Altura .....",alt
ura
100 GO TO 30
1000 REM *****
1010 LET radianes=grados*PI/180
1020 RETURN
```

c) Calcular la longitud de la sombra que proyectará un árbol, conociendo su altura.

ra y el ángulo que forma el sol con el horizonte.

```

10 REM *****
   *  ARBOL  *
   *  *****  *
20 BORDER 4: PAPER 4: INK 0: C
L5
30 INPUT "Altura arbol (max. 50)"; altura
32 CLS
40 IF altura<1 OR altura>50 TH
EN GO TO 30
42 PRINT "Altura arbol .."; alt
ura:
50 INPUT "Inclinacion sol (gra
dos)"; grados
60 IF grados<1 OR grados>89 TH
EN GO TO 30
62 PRINT "Inclinacion ..."; gra
dos:
70 GO SUB 1000
80 LET sombra=altura/TAN radia
nes:
90 PRINT "Sombra ....."; som
bra:
100 GO TO 30
1000 REM *****
1010 LET radianes=grados*PI/180
1020 RETURN
  
```

Función exponencial

Se llaman así aquellas funciones en las que el exponente es un número variable.

Ejemplo:

$$y = a^x$$

donde «a» es la base y «x» el exponente variable.

Cuando la base es mayor que uno ($a > 1$), la función exponencial es una *función creciente*.

Ejemplo:

```

10 LET base = 3
20 FOR x = 1 TO 22
30 PRINT base ^ x
40 NEXT x
  
```

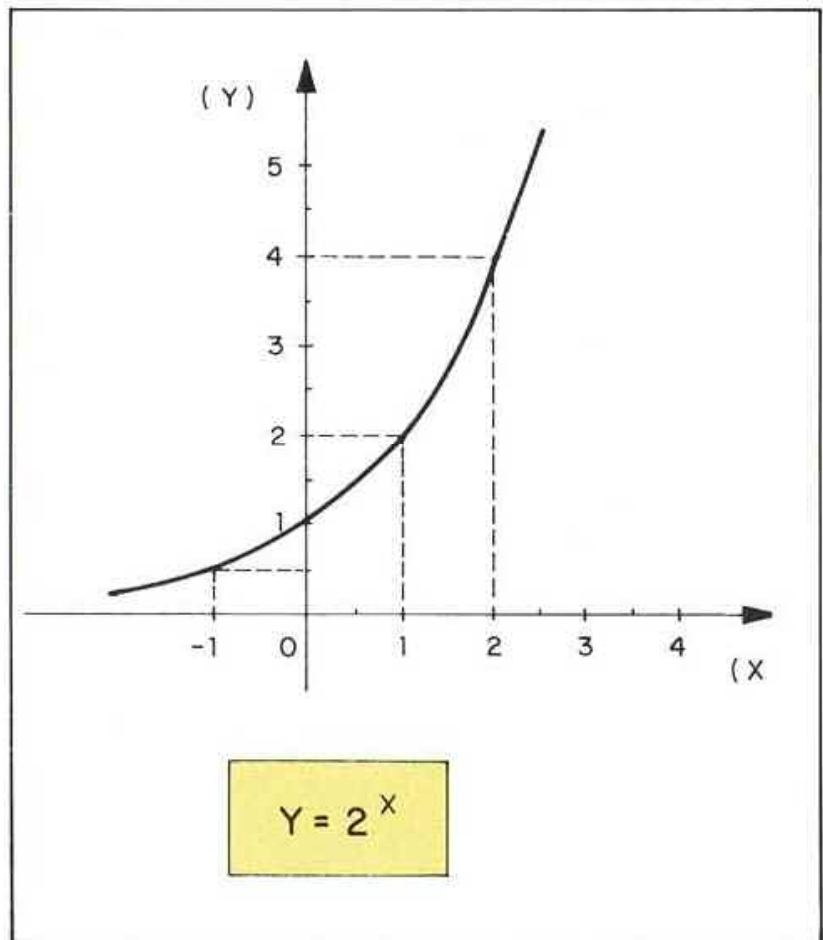
Observe cómo los diversos valores que toma la función de base «3» y exponente variable, entre «1» y «22», van aumentando.

Cuando por el contrario, el valor de la base se halla comprendido entre 0 y 1 ($0 < a < 1$), se trata de una *función decreciente*.

Ejemplo:

```

10 LET base = 0.99
20 FOR x = 1 TO 22
30 PRINT base ^ x
40 NEXT x
  
```



Función exponencial creciente

En este caso se observa que el valor de la función va decreciendo.

De los diversos valores que puede tomar la base «a», hay uno que se utiliza frecuentemente en Matemáticas, Física y otras ciencias, es el denominado *número «e»*, base de los *logaritmos neperianos* y cuyo valor aproximado es:

$$e = 2,7182818$$

EXP

Acceso al teclado

EXP



MODO E

Definición

«EXP» retorna el valor de la función exponencial creciente, teniendo como base el número «e».

Ejemplo:

- LET c = EXP 4
- PRINT EXP total
- LET k = c/EXP 8
- PRINT EXP (k * 3)

La función «EXP» cumple la siguiente igualdad:

$$\text{EXP } x = e^x = e \wedge x$$

Una forma de visualizar el valor numérico de «e», es:

```
PRINT EXP 1
```

ya que todo número elevado al

exponente unidad da como resultado el propio número:

$$n^1 = n$$

Ejemplo:

PRINT 27 ↑ 1

Función logarítmica

Es la inversa de la función exponencial.

Ejemplo:

$$y = \log_a X$$

La expresión anterior se lee de la siguiente forma: «y» es igual al logaritmo de «x» en base «a»; donde «a» es la base de los logaritmos.

Se denomina logaritmo de un número al exponente a que es preciso elevar la base para obtener dicho número.

$$\log_a X = y$$

$$a^y = X$$

Las bases de los logaritmos más utilizadas son la *decimal* y la *neperiana*. Los logaritmos de base decimal son también conocidos como logaritmos *vulgares* o de *Briggs*.

Ejemplo:

$$y = \log_{10} X$$

Los logaritmos neperianos deben su nombre a John Neper (matemático inglés) y tienen como base el conocido número «e». También son llamados logaritmos *naturales*.

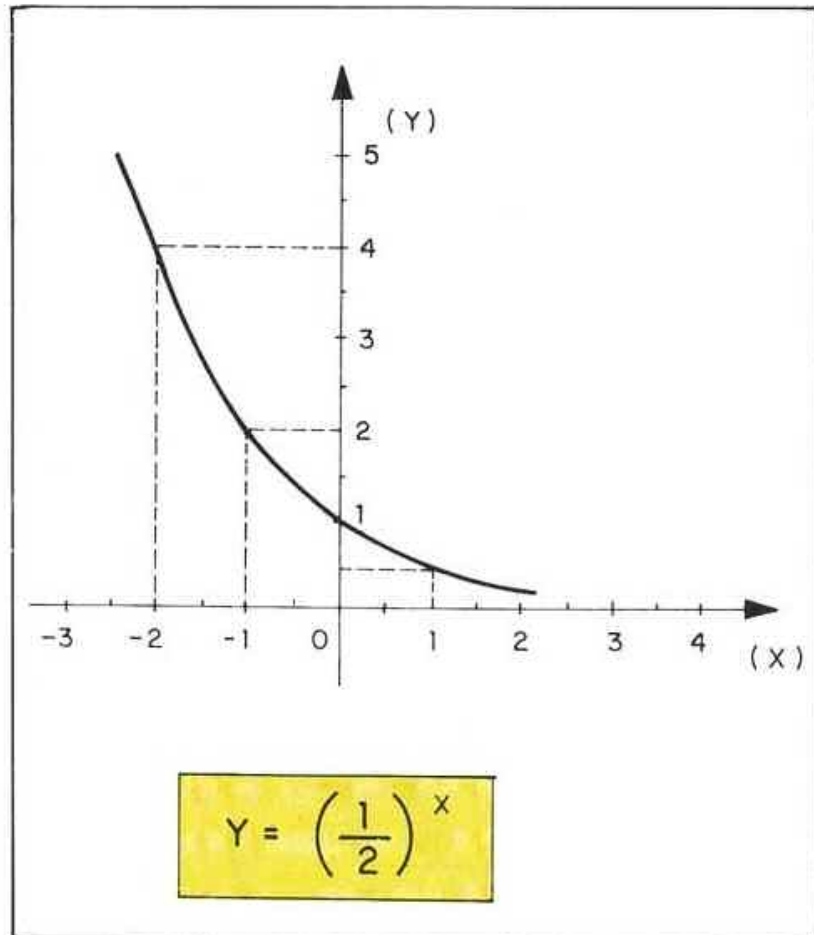
Ejemplo:

$$y = \log_e X$$

abreviadamente, puede escribirse el logaritmo neperiano como «ln».

Ejemplo:

$$y = \ln X$$



Función exponencial decreciente

La función logarítmica incluida en el juego de sentencias del Spectrum, es la de base neperiana.

Ejemplos:

- LET a = LN 7
- PRINT LN suma
- LET b = 2 * LN k
- PRINT LN (17/n)

Cuando el argumento es igual o inferior a «0», se visualiza el mensaje:

A Invalid argument

La utilización de los logaritmos es de uso frecuente en calculadoras de las denominadas científicas, y cómo no, en el Spectrum, ya que basándose en dos de los teoremas de dicha función se realizan los cálculos internos de potenciación y radicación.

- a) El logaritmo de una potencia es igual al producto del exponente por el logaritmo de la base.

LN

Acceso al teclado

LN

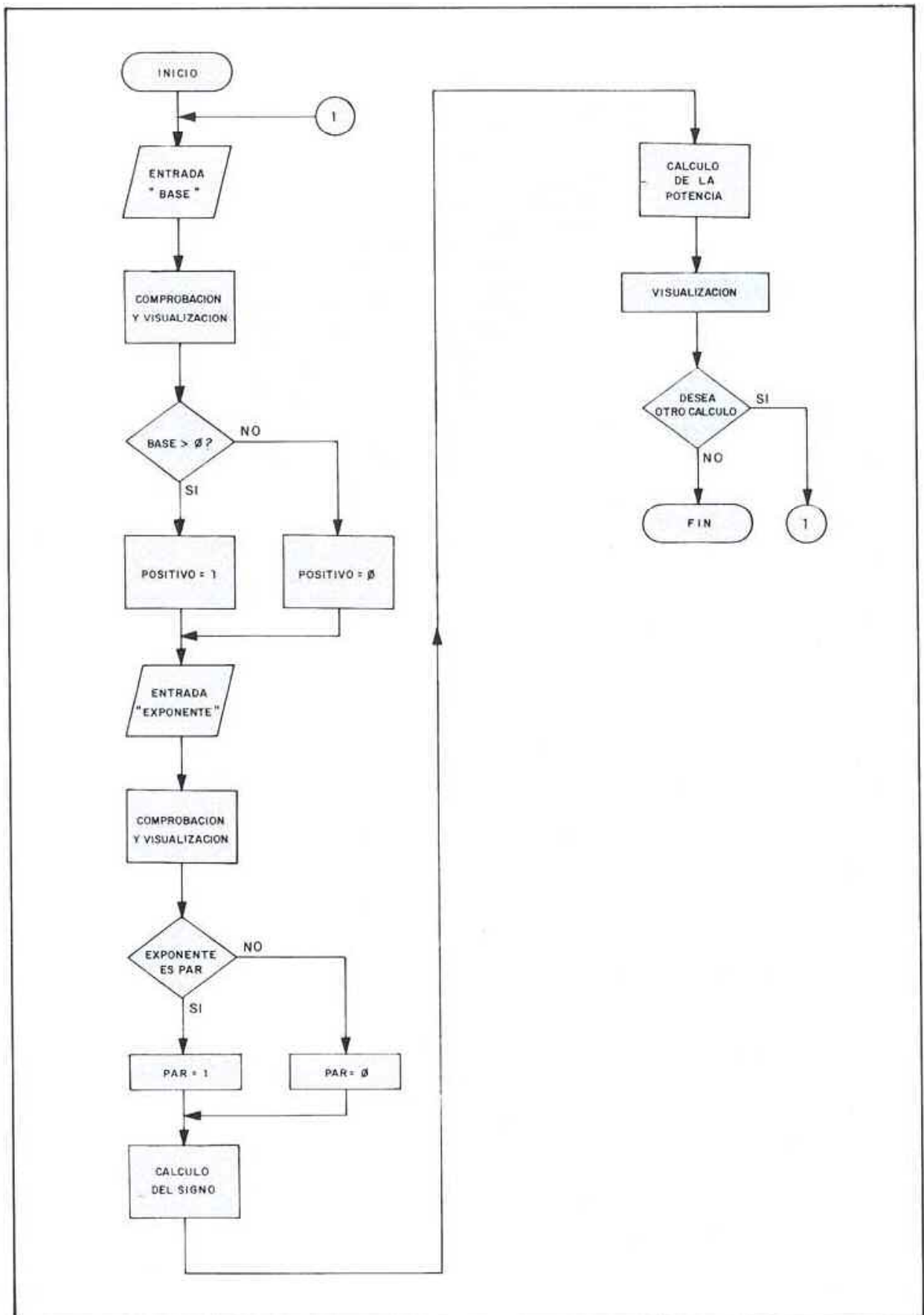


MODO E

BEEP

Definición

La función «LN» retorna el logaritmo neperiano del argumento.



Programa «potencia».

PROGRAMA 1

```

10 REM *****
    *
    * POTENCIACION *
    *
    *****

20 INPUT "Base >>> "; base
30 IF (ABS base>999) OR (base=
0) THEN GO TO 20
40 PRINT "Base .....", bas
e'
50 IF base>0 THEN LET positivo
=1: GO TO 70
60 LET positivo=0
70 INPUT "Exponente >>> "; expo
nente
80 IF ABS exponente>10 OR INT
exponente<>exponente THEN GO TO
70
90 PRINT "Exponente .....", exp
onente'
100 LET divisor=exponente/2
110 LET entero=INT divisor
120 IF divisor=entero THEN LET
par=1: GO TO 140
130 LET par=0
140 IF (positivo=0 AND par=1) O
R (positivo=1) THEN LET s$="+":
GO TO 160
150 LET s$="-"
160 LET resultado=EXP (exponent
e*LN (ABS base))
170 PRINT base;" elevado a ";ex
ponente;" es igual a:"
180 PRINT " "s$;" ";resultado
190 PRINT #0;"Desea otro calcul
o (S/N)"
200 IF INKEY$="" THEN GO TO 200
210 LET a$=INKEY$
220 IF a$="s" OR a$="S" THEN CL
S : GO TO 10
230 IF a$="n" OR a$="N" THEN CL
S : STOP
240 GO TO 210

```

$$\ln a^b = b \cdot \ln a$$

por tanto

$$a^b = \text{exponencial}(b \cdot \ln a)$$

Para comprobar que el ordenador realiza la potenciación con ayuda de los logaritmos, introduzca el siguiente comando directo:

$$\text{PRINT } 2 \uparrow 13$$

observará que el resultado es «8192»; si restamos ambos valores, el resultado debería ser «0».

$$\text{PRINT } 2 \uparrow 13 - 8192$$

¿Por qué razón no es «0»? esto se debe a que el ordenador

no multiplica «13» veces el número «2», sino que por el contrario, lo calcula de acuerdo con el teorema expuesto anteriormente.

Introduzca:

$$\text{PRINT EXP}(13 \cdot \text{LN } 2)$$

realice la siguiente operación:

$$\text{PRINT } 2 \uparrow 13 - (\text{EXP}(13 \cdot \text{LN } 2))$$

en este caso el resultado sí es cero.

Al realizar la operación $2 \uparrow 13$ no se visualiza «8192.000022888184» ya que, debido al sistema de presentación del Spectrum, este valor queda redondeado a «8192».

b) El otro teorema dice que el logaritmo de una raíz es igual al cociente entre el logaritmo del radicando y el índice de la raíz.

$$\ln \sqrt[b]{a} = \ln(a)/b$$

luego

$$\sqrt[b]{a} = \text{exponencial}(\ln(a)/b)$$

Al igual que en el caso anterior, la radicación también se realiza con ayuda de los logaritmos.

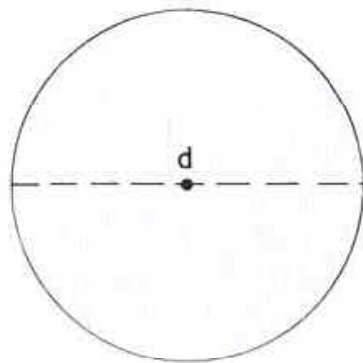
Realice los siguientes pasos:

```

PRINT SQR 144
PRINT SQR 144 - 12
PRINT EXP (LN 144/2)
PRINT SQR 144 - (EXP (LN 144/2))

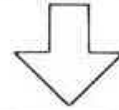
```

Las explicaciones dadas en el caso de la potenciación también son válidas para la potenciación.

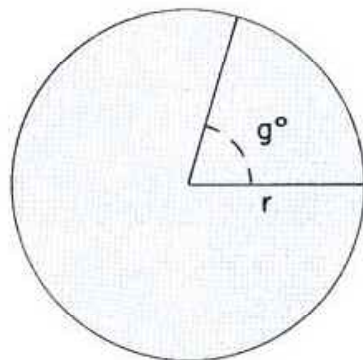


AREA DEL CIRCULO

$$S = \pi \left(\frac{d}{2} \right)^2$$

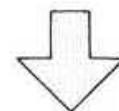


```
DEF FN S(d) = PI * (d/2.) ^ 2
```



AREA DEL SECTOR CIRCULAR

$$S = \frac{\pi r^2}{360} \cdot g^\circ$$



```
DEF FN C(r,g) = PI * r ^ 2 / 360 * g
```

Definición de funciones

Compare los resultados anteriores con los proporcionados por «144 ↑ (1/2)».

Aunque el Spectrum solamente calcula, de forma directa, los logaritmos de base neperiana, pueden calcularse en cualquier otra base, teniendo en cuenta la siguiente igualdad:

$$\log_a X = \frac{\ln X}{\ln a}$$

que dice, que el logaritmo de un número «X» en base «a» es igual al cociente entre el logaritmo neperiano del número y el logaritmo neperiano de la base.

Ejemplos:

- a) Calcular el logaritmo en base 10 de 4 ($\log_{10} 4$).

```
PRINT LN (4)/LN (10)
```

- b) Calcular el logaritmo de 15 en base 2 ($\log_2 15$).

```
PRINT LN 15/LN 2
```

El siguiente programa calcula el logaritmo de cualquier número comprendido entre «1» y «99999999», la base puede elegirse entre «2» y «20».

```

10 REM *****
      LOGARITMOS
*****
20 INPUT "Base >>> ";base
25 CLS
30 IF base<=1 OR base>20 THEN
GO TO 20
40 PRINT "Base .....":bas
e
50 INPUT "Número >>> ";numero
60 IF numero<=0 OR numero>9999
9999 THEN GO TO 50
70 PRINT "Número .....":num
ero
80 LET logaritmo=LN numero/LN
base
90 PRINT "El logaritmo de ";nu
mero
100 PRINT "en base ";base," es
"; logaritmo
110 GO TO 20

```

Debido a que las potencias se calculan con ayuda de los logaritmos neperianos, no se puede calcular una potencia de base negativa, ya que daría el mensaje de error:

A Invalid argument

Ejemplo:

```
PRINT (-2) 3
```

Con ayuda del programa número «1» se puede calcular hasta la potencia décima de cualquier número comprendido entre «999» y «-999», excepto el cero.

Definición de funciones

El sistema operativo del Spectrum permite al usuario definir sus propias funciones, éstas pueden definirse en cualquier parte del programa, pero por motivos de claridad conviene hacerlo al comienzo del mismo.

Para poder utilizar estas funciones posteriormente, es necesario que tengan asignado

un nombre, éste está compuesto por una sola letra en el caso de funciones numéricas y por una letra seguida del signo «\$» en el caso de funciones de cadena.

DEF FN

Acceso al teclado



Definición

Esta sentencia permite la definición de las funciones de usuario.

Su estructura general es:

| SENTENCIA | ARGUMENTO |
|-----------|-----------------------------|
| DEF FN | nombre (variable) = función |

Ejemplos:

- DEF FN a (X) = 7 * X + SIN X
- DEF FN c (y) = COS (y — 200)
- DEF FN j (n) = PI + n
- DEF FN t (u) = 20 + (2 + u)

Las variables también tienen que estar formadas por una sola letra.

En los ejemplos anteriores se incluía una sola variable en la definición de la función, pero ésta puede contener hasta veintiséis distintas, una por cada letra.

Ejemplos:

- DEF FN n (a, b) = (a + b)/100
- DEF FN r (1, m, n) = EXP n * (mm + LN 1)

Llamadas a funciones definidas



Acceso al teclado

RED
CAPS LOCK



Definición

Esta sentencia permite utilizar las funciones definidas previamente.

Su estructura general es:

FN nombre (parámetros)

Ejemplos:

- PRINT FN a (10)
- LET x = FN n (total)
- PRINT FN X (37,23)
- LET j = FN p (c, 3)

Veamos unos ejemplos prácticos:

- a) Calcular el área de un círculo en función de su diámetro. Primero es necesario definir la función:

```
10 DEF FN S (d) = PI * (d/2) 2
```

Veamos los valores que retorna la función para distintos diámetros (entre 1 y 20).

```
20 FOR n = 1 TO 20
30 PRINT "Diámetro: "; n,
PRINT "Área: "; FN S (n)
40 NEXT n
```

- b) Calcular el área de un sector circular en función de su radio y del número de grados.

Definición de la función:

```
10 DEF FN c (r, g) = PI * r ^ 2 / 360 * g
```

Un ejemplo de utilización podría ser el siguiente:

```
20 INPUT "Radio: "; radio
30 INPUT "grados: "; grados
40 PRINT FN c (radio, grados)
```

Errores

Hay una serie de mensajes típicos de error que se visualizan al manejar incorrectamente las funciones.

- a) Al intentar manejar una función no definida previamente aparece el mensaje:

```
P FN without DEF
```

Ejemplo:

```
10 REM *****
   ERROR "P"
   *****
20 INPUT "temperatura? ", calor
30 LET x=FN a(calor)
```

- b) Cuando se hace una llamada a una función definida, y no coincide el número de parámetros aparece el mensaje:

```
Q Parameter error
```

Este mensaje también aparece cuando alguno de los parámetros no es del mismo tipo que los definidos en la función; es decir, un parámetro de cadena en

una función numérica o viceversa.

Ejemplo:

```
10 REM *****
   ERROR "Q"
   *****
20 DEF FN r (c, v) = (c+v) / c*v
30 INPUT "Numero 1 > ", n1
40 INPUT "Numero 2 > ", n2
50 PRINT FN r (n1)
```

en este caso falta un parámetro en la llamada de la función. También daría error las siguientes instrucciones:

```
50 PRINT FN r (n1, n2, 10)
```

(sobra un parámetro)

```
50 PRINT FN r (n1, "pepe")
```

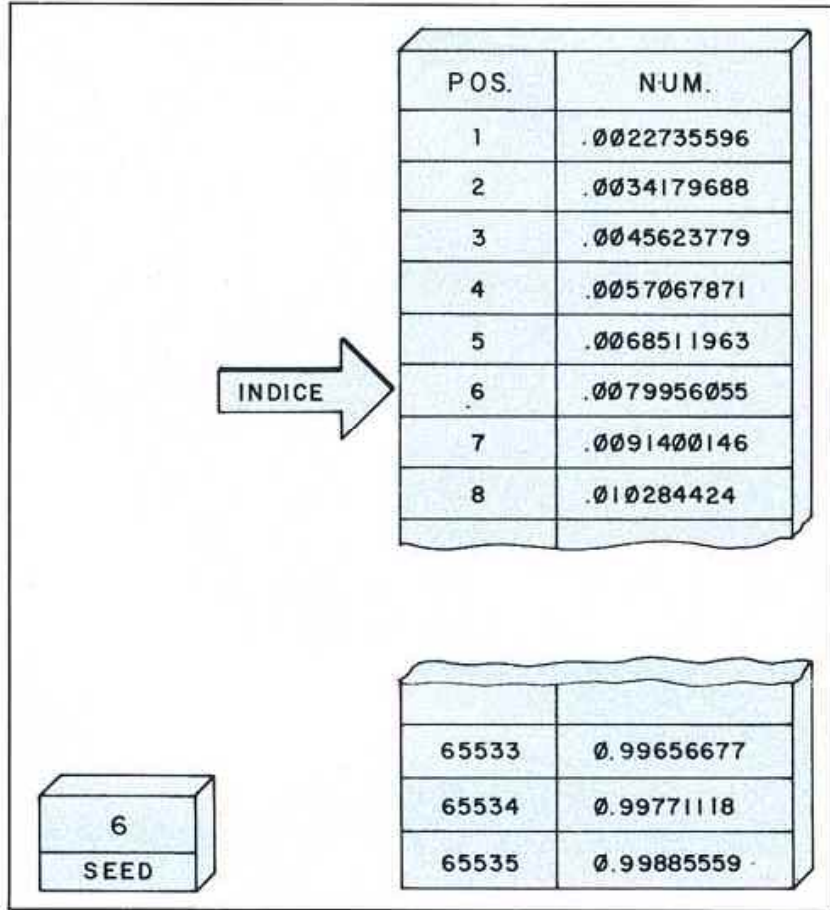
(parámetro de cadena) ■

FUNCION ALEATORIA

Como su propio nombre indica, una función aleatoria es aquella que retorna un valor al azar, es decir, aquel que no sigue ninguna ley o algoritmo; en realidad esto no es del todo cierto ya que es bastante difícil, por no decir imposible, implementar una función de este tipo en un ordenador, por tanto, éstos incluyen una función llamada *pseudoaleatoria*.

La función pseudoaleatoria, siguiendo un algoritmo, elige adecuadamente ciertos números para que parezcan aleatorios. En el caso del Spectrum existe una secuencia ciclica formada por «65536» números distintos.

Las funciones aleatorias tienen un extenso campo de aplicación en los juegos y en los programas didácticos.



RND

Acceso al teclado



Definición

«RND» retorna uno de los

Variable «SEED».

«65536» números que forman la secuencia de números aleatorios. Al ser una función debe ir acompañada de sentencias tales como «PRINT», «LET», «IF»... etc.

Ejemplos:

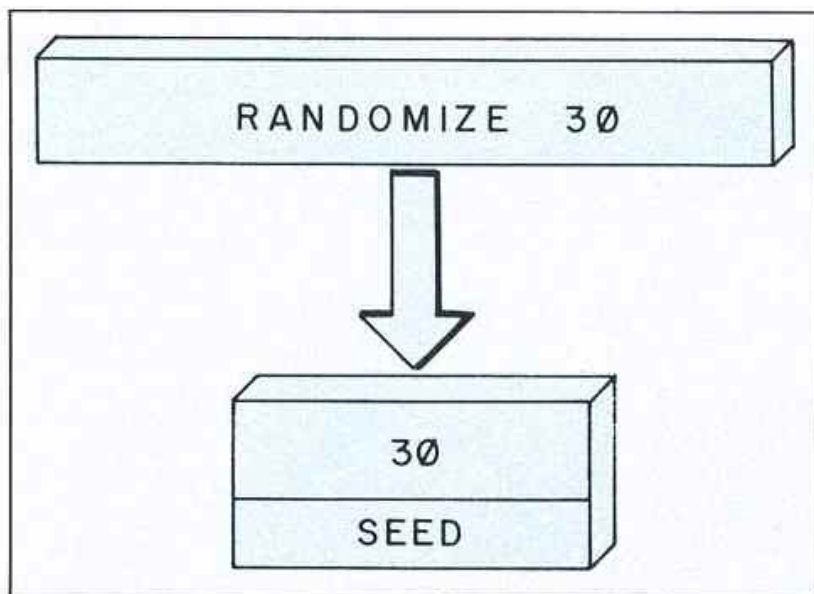
- LET a = RND
- LET b = INT (RND + 5)
- PRINT RND
- PRINT 3 + RND

Ejecute el siguiente programa:

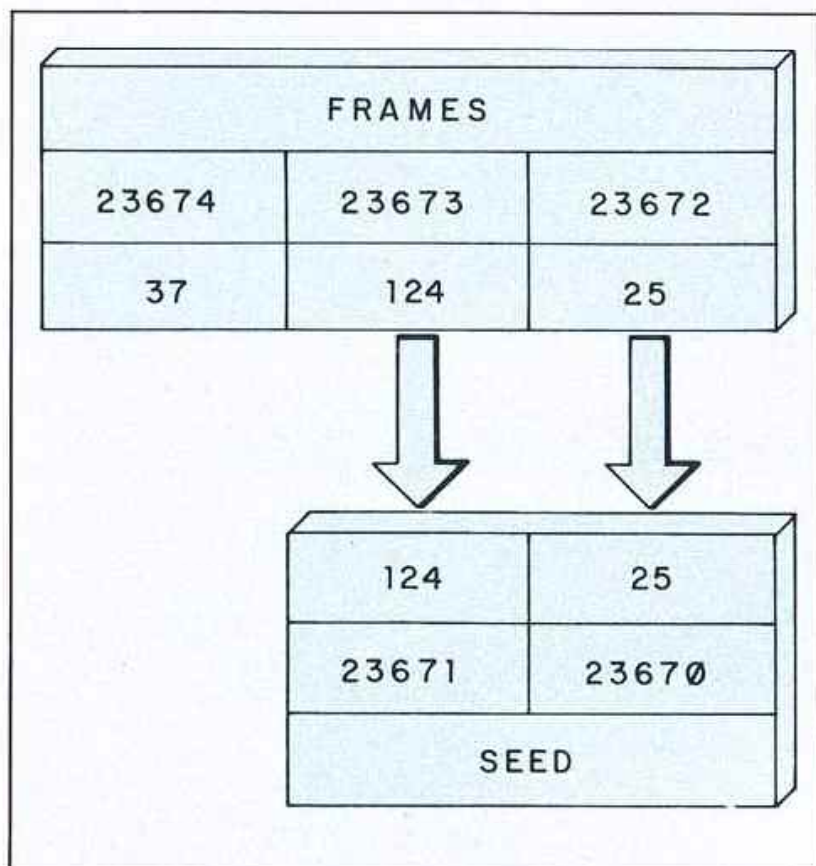
```
10 FOR n = 1 TO 44
20 PRINT RND
30 NEXT n
```

observará que todos los números son inferiores a «1», ya que el valor retornado por «RND» está comprendido entre «0» y «1», en alguna ocasión puede valer «0», pero nunca «1».

Para conseguir números aleatorios comprendidos entre otros rangos, por ejemplo,



Randomize «n».



Randomize «0».

de «0» a «10», podríamos utilizar:

```
10 FOR n = 1 TO 44
20 PRINT RND * 10
30 NEXT n
```

pero si lo que se desea es obtener números aleatorios enteros, utilizaríamos:

```
10 FOR n = 1 TO 44
20 PRINT INT (RND * 10),
30 NEXT n
```

Existe un pequeño algoritmo para obtener números aleatorios enteros comprendidos entre dos cualesquiera, ambos inclusive:

```
INT ((Y-X+1) * RND) + X
```

donde «X» es el menor e «Y» el mayor, veamos unos ejemplos:

```
10 LET y = INT (22 * RND)
20 LET x = INT (32 * RND)
30 PRINT AT y, x: (*)
40 GOTO 10
```

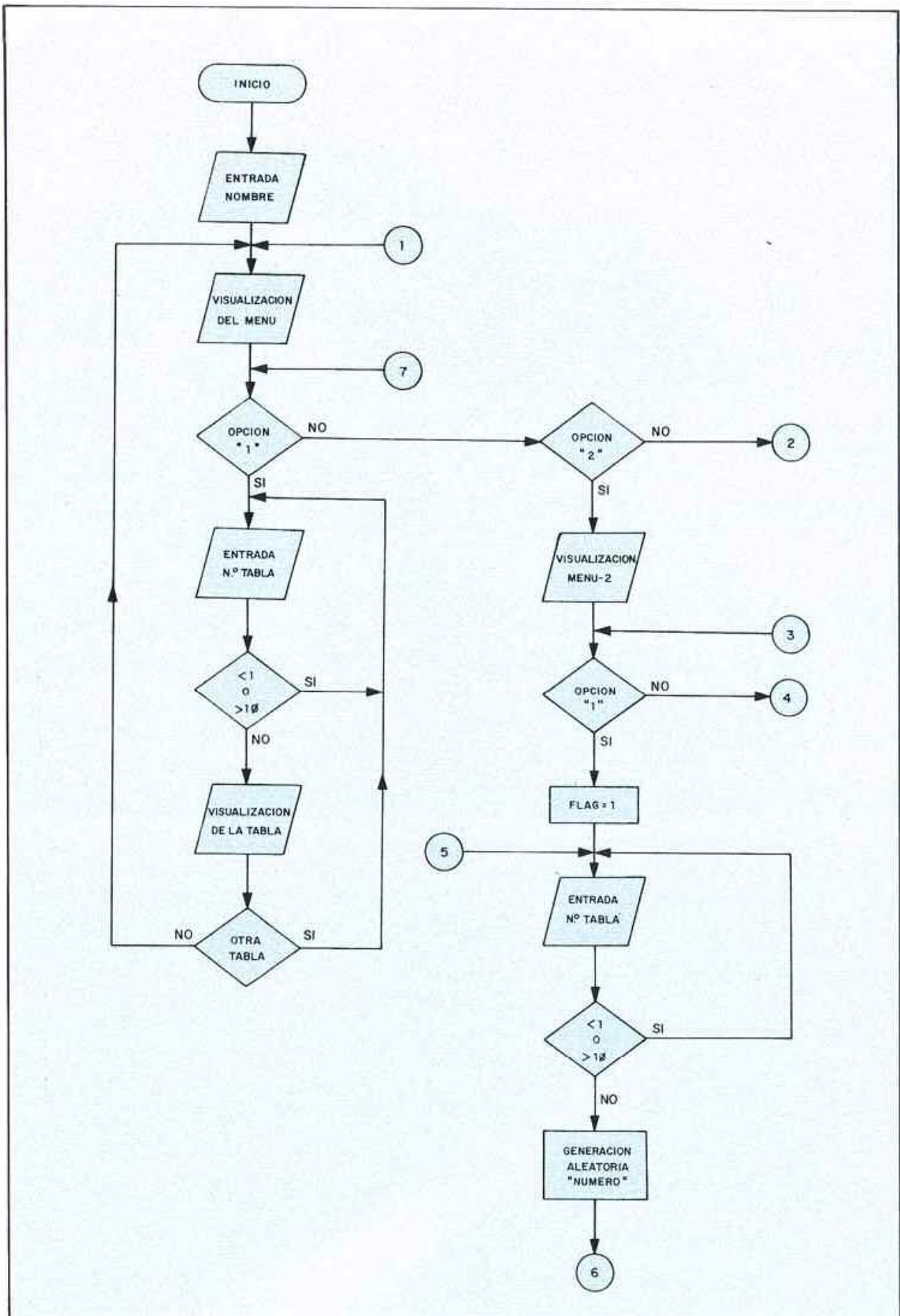
Este programa visualiza un asterisco (*) en una posición aleatoria de la pantalla, si tiene paciencia, al cabo de un rato, verá la pantalla llena de este símbolo, en total «704».

Una variante de este programa podría consistir en introducir estas tres líneas suplementarias:

```
5 INPUT (Tiempo >); X
32 PAUSE X
34 PRINT AT y, x: (*)
```

la variable «X» controla el tiempo de visualización del asterisco. Introduzca, por ejemplo, el valor «5» y parecerá que hay una mosca revoloteando sobre la pantalla de su televisor.

Dentro del Sistema Operativo del Spectrum está implementado un algoritmo que calcula sucesivamente cual de los «65536» números aleatorios será el siguiente en ser presentado. El resultado de este algoritmo se almacena en dos posiciones consecutivas de memoria, éstas forman una de las variables del sistema conocida por el nombre de «SEED»; el contenido de esta variable es, por



Programa «TABLA».

tanto, la dirección dentro de la tabla de números aleatorios donde se encuentra el siguiente a presentar.

Edite las siguientes líneas de programa; al ejecutarlas, aparecerá en una columna el contenido de la variable SEED y en la otra el número aleatorio correspondiente.

La línea «50» se encarga de visualizar el contenido de dicha variable.

El programa número «1» está basado en el conocido juego de los *barquitos*, pero en esta ocasión en vez de tener que hundir una completa flota enemiga compuesta por varias unidades, tan solo será necesario acertar la posición de una lancha que está situada en una retícula de «10» por «10».

Las instrucciones del juego son sencillas, deberá introducir primero la coordenada vertical (y), posteriormente la horizontal (x). Cuando una de las coordenadas coincide aparece un mensaje de «alerta» indicando que vamos por buen camino.

La estructura general del programa es la siguiente:

- 10 : Presentación del programa.
- 20 : Asignación de los colores azul para el borde, verde para el fondo y negro para los caracteres.
- 30 : Inicialización de la variable (record).
- 60-140 : Bucles utilizados para dibujar la retícula. Se han utilizado las sentencias (PLOT) y (DRAW), éstas serán explicadas en el capítulo dedicado al dibujo.
- 150-180 : Visualización de las posiciones de la retícula.
- 220-250 : Cálculo de la posición del barco. Se utiliza (RANDOMIZE), de esta forma, la posi-

ción depende del tiempo que lleve conectado el ordenador.

- 260 : Comienzo del juego.
- 270-320 : Inicialización y visualización de variables.
- 330-400 : Introducción del valor de las coordenadas, éste tiene que ser un número entero y estar comprendido (dentro de los márgenes (0 a 9).
- 410 : Dibujo de la posición disparada.
- 425-427 : Incremento y visualización del número de intentos.
- 430 : Comprobación si las coordenadas han sido acertadas.
- 440 : Comprobación si una de las coordenadas ha sido averiguada.
- 450-500 : Indicación de disparo fallado.
- 510 : Petición de nuevas coordenadas.
- 520-600 : Indicación de (acierto). Visualización de la puntuación y actualización del record.
- 670-730 : ¿Continuación del juego?

RANDOMIZE

Acceso al teclado

RND



Tipo de sentencia

Comando de programación.

Definición

Si intenta buscar la palabra

clave «RANDOMIZE» no la encontrará, ya que en su lugar está «RAND», forma simplificada de la anterior.

Su estructura general es:

| SENTENCIA | ARGUMENTO |
|-----------|------------------|
| RANDOMIZE | expresión numér. |

Ejemplos:

- RANDOMIZE 70
- RANDOMIZE dato
- RANDOMIZE 10 * alta/3
- RANDOMIZE

«RANDOMIZE» sin argumento toma, por defecto, el valor «0».

El argumento debe estar comprendido entre «0» y «65535», de lo contrario aparecerá el mensaje:

B Integer out of range

Existen dos formas de utilizar «RANDOMIZE»:

a) Cuando su argumento tiene un valor comprendido entre «1» y «65535».

El argumento se utiliza para definir en qué número de la secuencia va a comenzar la generación de números aleatorios.

Ejemplo:

```
10 RANDOMIZE 50
20 FOR n = 1 TO 44
30 PRINT RND,
40 NEXT n
```

Observará que cada vez que se ejecuta el programa la secuencia de número aleatorio es exactamente la misma, ya que siempre se inicializa con el valor «50».

Una forma de poder visualizar la lista completa de números aleatorios, podría ser la siguiente:

PROGRAMA 1

```

10 REM *****
*          *
*  CURSO  *
*  BASIC  *
*          *
*  *****
*  !!! AGUA !!!
*          *
*  *****
LS
20 BORDER 1: PAPER 4: INK 0: C
30 LET record=0
60 REM *****
*          *
*  DIBUJO *
*          *
*  *****

70 FOR n=72 TO 152 STEP 8
80 PLOT 24,n

90 DRAW 80,0
100 NEXT n
110 FOR n=24 TO 104 STEP 8
120 PLOT n,72
130 DRAW 0,80
140 NEXT n
150 REM *****
*          *
*  COORDENADAS *
*          *
*  *****

160 FOR n=3 TO 12
170 PRINT AT n,1;n-3
175 PRINT AT 1,n;n-3
180 NEXT n
220 REM *****
*          *
*  SITUACION *
*          *
*  *****

230 RANDOMIZE
240 LET barcoy=INT (10*RND)
250 LET barcox=INT (10*RND)
260 REM *****
*          *
*  COMIENZO *
*          *
*  *****

270 LET intentos=0
272 LET puntuacion=0
280 PRINT AT 3,16;"RECORD "
record
290 PRINT AT 5,16;"PUNTUACION "
puntuacion
300 PRINT AT 7,16;"INTENTOS "
intentos
310 PRINT AT 10,16;"COORD y:"
320 PRINT AT 12,16;"COORD x:"
330 REM *****
*          *
*  DISPARO *
*          *
*  *****

```

```

340 PRINT AT 10,25;" ";AT 12,25
";
350 INPUT "Coordenada y >>> ";p
osy
360 IF (INT posy<>posy) OR (pos
y<0) OR (posy>9) THEN GO TO 350
370 PRINT AT 10,25;posy
380 INPUT "Coordenada X >>> ";p
osx
390 IF (INT posx<>posx) OR (pos
x<0) OR (posx>9) THEN GO TO 370
400 PRINT AT 12,25;posx
410 PRINT AT posy+3,posx+3;"■"
420 REM *****
*          *
*  COMPROBACION *
*          *
*  *****

425 LET intentos=intentos+1
427 PRINT AT 7,27;intentos
430 IF (posy=barcoy) AND (posx=
barcox) THEN GO TO 520
440 IF (posy=barcoy) OR (posx=b
arcox) THEN PRINT FLASH 1;AT 14,
3;" ALERTA " : GO TO 460
450 PRINT AT 14,3;" "
460 PRINT AT 18,10;"!!! AGUA !!
!"
490 PAUSE 50
500 PRINT AT 18,10;" "
510 GO TO 340
520 REM *****
*          *
*  ACIERTO *
*          *
*  *****

530 PRINT AT 14,3;" "
540 PRINT AT 18,9;"!!! ACERTO !
!"
550 PRINT AT posy+3,posx+3; FLA
SH 1;"■"
560 IF intentos>100 THEN LET pu
ntuacion=0: GO TO 580
570 LET puntuacion=100-intentos
580 PRINT AT 5,27;puntuacion
590 IF puntuacion>record THEN L
ET record=puntuacion
600 PRINT AT 3,27;record
670 REM *****
*          *
*  CONTINUACION *
*          *
*  *****

680 PRINT #0;AT 1,2;"Desea juga
r otra vez (S/N) "
690 PAUSE 0
700 LET a$=INKEY$
710 IF a$="S" OR a$="s" THEN CL
S : GO TO 60
720 IF a$="N" OR a$="n" THEN CL
S : STOP
730 GO TO 690

```

```

10 FOR n = 1 TO 65535
20 RANDOMIZE n
30 PRINT n, RND
40 NEXT n

```

Si tiene la suficiente paciencia podrá averiguar cual

es el número de secuencia que hace que la función «RND» retorne el valor «0», pero si no, compruébelo con:

```

10 RANDOMIZE 45438
20 PRINT RND

```

Cuando se ejecuta una sentencia del tipo «RANDOMIZE n» lo que en realidad sucede es que la variable «SEED» asume el valor del argumento; el siguiente programa lo demuestra:

PROGRAMA 2

```

10 REM *****
*
* CURSO BASIC *
*
*****
* LA TABLA *
*
*****

20 BORDER 4: PAPER 4: INK 0: C
LS
99 REM
*****
*
* MENU *
*
*****

100 INPUT "Como te llamas >>> "
; LINE a$
110 PRINT AT 1,1;a$;" elige una

de las tres"" opciones:"
120 PRINT AT 9,8;"1 - REPASAR"
130 PRINT AT 12,8;"2 - EXAMEN"
132 PRINT AT 15,8;"3 - FIN"
140 IF INKEY$="" THEN GO TO 140
150 IF INKEY$="1" THEN GO TO 10
00
160 IF INKEY$="2" THEN GO TO 20
00
162 IF INKEY$="3" THEN CLS : ST
OP
170 GO TO 140
1000 REM
*****
*
* REPASAR *
*
*****

1002 BORDER 1
1004 CLS
1010 INPUT "Que tabla deseas rep
asar >>> ";tabla
1020 IF tabla<1 OR tabla>10 THEN
GO TO 1010
1022 PRINT AT 1,2;"TABLA DE MULT
IPlicAR DEL """;tabla;" """"
1030 FOR n=1 TO 10
1040 LET resultado=tabla*n
1050 PRINT AT n+6,10;tabla;" x "
;n;
1060 IF n<>10 THEN PRINT " ";
1070 PRINT "=" ;resultado
1080 NEXT n
1090 PRINT #0;" Deseas repasar o
tra (S/N)"
1100 IF INKEY$="" THEN GO TO 110
0
1110 LET b$=INKEY$
1120 IF b$="S" OR b$="s" THEN GO
TO 1004
1130 IF b$="N" OR b$="n" THEN BO
RDER 4: CLS : GO TO 110
1140 GO TO 1100

2000 REM
*****
*
* MENU 2 *
*
*****

2010 BORDER 2
2020 CLS
2022 FOR n=1 TO 25: NEXT n
2030 PRINT AT 1,1;a$;" elige el
tipo:"
2040 PRINT AT 8,8;"1 - PARCIAL"
2050 PRINT AT 12,8;"2 - GENERAL"
2060 IF INKEY$="" THEN GO TO 206
0

```

```

2070 LET b$=INKEY$
2080 IF b$="1" THEN LET flag=1:
GO TO 2200
2090 IF b$="2" THEN LET flag=0:
GO TO 2200
2100 GO TO 2070
2200 REM
*****
*
* PREGUNTAS *
*
*****

2210 CLS
2212 RANDOMIZE
2220 PRINT AT 2,1;"PREGUNTAS : "
2230 PRINT AT 4,1;"ACIERTOS : "
2240 PRINT AT 6,1;"PORCENTAJE:"
2250 LET pregunta=1: LET acierto
=0: LET porcentaje=0
2260 IF flag=0 THEN GO TO 2350
2270 INPUT " Que tabla deseas >>
";tabla
2280 IF tabla<1 OR tabla>10 THEN
GO TO 2270
2290 LET numero=INT (RND*10)+1
2300 GO TO 2400
2350 LET tabla=INT (RND*10)+1

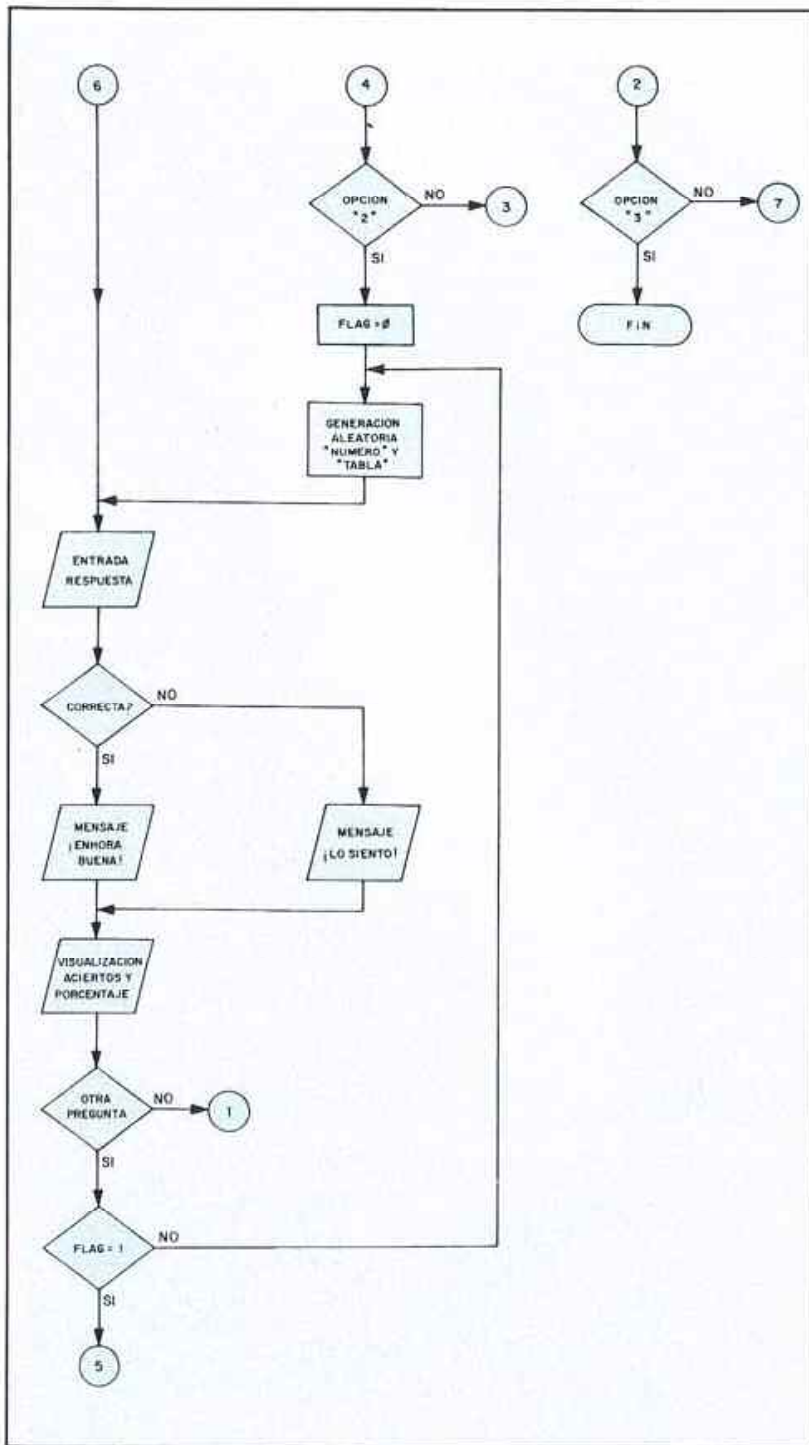
2360 LET numero=INT (RND*10)+1
2400 LET resultado=tabla*numero
2405 PRINT AT 2,13;pregunta
2410 PRINT AT 10,1;tabla;" x ";n
umero;" =
2420 INPUT (" Cuantas son ";tabl
a;" x ";numero;" >>> ");respuest
a
2440 PRINT AT 10,9;respuesta
2450 REM
*****
*
* ANALISIS *
*
*****

2460 IF respuesta=resultado THEN
GO TO 2480
2470 GO TO 2510
2480 PRINT AT 15,7;"!!! ENHORABU
ENA !!!"
2490 LET acierto=acierto+1

2500 GO TO 2530
2510 PRINT AT 15,7;"!!! LO SIENT
O !!!"
2520 PRINT AT 19,1;a$+" ";tabla;
" x ";numero;" son ";resultado
2530 PRINT AT 4,13;acierto
2540 LET porcentaje=INT (acierto
/pregunta*10000)/100
2550 PRINT AT 6,13;"
"
2560 PRINT AT 6,13;porcentaje;"
%"
2570 PRINT #0;" Deseas continuar
(S/N)"
2580 PAUSE 0
2590 LET b$=INKEY$
2600 IF b$="S" OR b$="s" THEN GO
TO 2530
2610 IF b$="N" OR b$="n" THEN BO
RDER 4: CLS : GO TO 110
2620 GO TO 2580
2630 LET pregunta=pregunta+1
2640 FOR n=15 TO 21

2650 PRINT AT n,0;"
"
2660 NEXT n
2670 IF flag=1 THEN GO TO 2290
2680 GO TO 2350

```



Programa «TABLA»

```

10 REM *****
      : VARIABLE "FRAMES" :
      : *****
20 PRINT "Frames", "Seed"
30 PRINT " "
40 RANDOMIZE
50 PRINT (PEEK 23673+256) + (PEEK
K 23672) * (PEEK 23671+256) + (PEEK
23670)
60 GO TO 40

```

b) Cuando su argumento es cero.

Cada vez que se conecta el ordenador, se ejecuta una sentencia «NEW» o se activa el «RESET», la generación de números aleatorios se realiza siempre a partir del mismo valor.

Edite el siguiente programa después de introducir «NEW» como comando directo:

```

10 PRINT RND
20 GOTO 10

```

anote alguno de los valores obtenidos. Vuelva a introducir «NEW» y edite otra vez el programa. Comprobará que los resultados son idénticos.

Para conseguir una función más aleatoria debe introducirse previamente:

```
RANDOMIZE 0
```

o en su defecto,

```
RANDOMIZE
```

Ejemplo:

```

10 RANDOMIZE
20 PRINT RND
30 go to 20

```

En este caso el valor retornado por «RND» está en función del tiempo que lleva conectado el ordenador. Existe otra variable del sistema conocida por el nombre de «FRAMES» que almacena indirectamente este tipo. La variable «FRAMES» ocupa tres posiciones de memoria, veamos un ejemplo de lectura de dicha variable:

```

10 PRINT
      PEEK 23674+65536)+
      PEEK (23673+256)+
      PEK 23672
20 GOTO 10

```

Cuando se ejecuta «RANDOMIZE 0» la variable «SEED» asume el valor de las dos posiciones de memoria menos significativas de «FRAMES». Veamos un ejemplo:


```

10 REM *****
   RANDOMIZE *
*****
12 PRINT "Randomize", "Seed"
14 PRINT
20 INPUT "Numero " ; azar
30 RANDOMIZE azar
40 PRINT azar, PEEK 23670+PEEK
23671+256
50 GO TO 20

```

```

10 REM *****
   VARIABLE SEED *
*****
12 LET seed=23670
20 PRINT "Variable ""SEED""",
Numero Aleatorio"
30 PRINT " "
40 FOR n=1 TO 19
50 PRINT (PEEK seed)+(PEEK (se
ed+1)*256)
60 PRINT RND
70 NEXT n

```

Puede observar que, en ocasiones, existe una diferencia entre ambas, de una unidad debido al retardo que hay entre la ejecución de «RANDOMIZE» y la visualización del contenido de la variable «FRAMES»; recuerde que «FRAMES» se incrementa según transcurre el tiempo y «SEED» se actualiza al ejecutarse «RANDOMIZE».

El programa número «2» pertenece al grupo denominado «UTILIZADES» y tiene aplicación en el campo de la enseñanza, ya que permite a los principiantes estudiantes de la EGB repasar la tabla de multiplicar o contestar las preguntas que el ordenador realice sobre el tema.

Lo primero que hace el ordenador es preguntarnos nuestro nombre, ya que en diversas ocasiones hace referencia a él. Posteriormente

se presenta un menú con tres opciones:

```

1 - REPASAR
2 - EXAMEN
3 - FIN

```

La opción «1» permite repasar la tabla de multiplicar del número que elijamos, ya que se nos visualiza ésta en la pantalla.

La opción «EXAMEN» tiene otras dos opciones:

```

1 - PARCIAL
2 - GENERAL

```

Eligiendo la primera, el ordenador nos pregunta la tabla de multiplicar de un número determinado, en cambio con la segunda, las preguntas son sobre cualquier número.

La opción «3» permite parar la ejecución del programa.

La estructura general es:

- 10 : Comentario con el nombre del programa.
- 20 : Asignación del color verde para fondo y borde, y negro para los caracteres.
- 100 : Entrada del nombre.
- 110-132 : Presentación del menú.
- 140-170 : Detección de la opción elegida.
- 1000 : Comienzo de la opción (1) (REPASAR).
- 1002 : Borde de color azul.
- 1010-1020 : Entrada y comprobación del número de tabla.

- 1022-1080 : Visualización de la tabla.
- 1090-1140 : Selección de una nueva tabla, o salto al menú principal.
- 2000 : Comienzo de la opción (2).
- 2010 : Borde de color rojo.
- 2022-2050 : Presentación del menú secundario.
- 2060-2100 : Detección de la opción elegida.
- 2200 : Comienzo de las preguntas.
- 2212 : Inicialización de la secuencia de números aleatorios, ésta depende del tiempo que lleve conectado el ordenador.
- 2220-2250 : Visualización de rótulos e inicialización de variables.
- 2260-2360 : Generación aleatoria de la pregunta, dependiendo de la opción elegida.
- 2400 : Cálculo del resultado.
- 2405-2420 : Visualización de la pregunta y entrada de la respuesta.
- 2440 : Visualización de la respuesta.
- 2450 : Comienzo del análisis.
- 2460 : Comparación de los resultados.
- 2480-2490 : Mensaje de acierto.
- 2510-2520 : Mensaje de fallo.
- 2530-2560 : Visualización del número de aciertos y porcentaje.
- 2570-2620 : Selección de otra pregunta o salto al menú principal.
- 2630 : Incremento de la variable (pregunta).
- 2640-2660 : Borrado del mensaje.
- 2670-2680 : Selección de la línea de salto para una nueva pregunta. ■

FUNCIONES DE CADENA

En capítulos anteriores se explicaron las funciones matemáticas y la pseudoaleatoriedad; en éste, por el contrario, van a ser estudiadas las *funciones de cadena*.

Conviene, antes de leer este capítulo, si no se tiene el concepto de *cadena* lo suficientemente claro, repasar el dedicado al «Código ASCII» (pag. 37) y «operaciones con

cadena» (pag. 42), en este último se tratan temas tales como:

- CONCATENACION
- SUBCADENAS
- FRAGMENTACION
- ORDENACION

Las funciones de cadena implementadas dentro del juego de sentencias del

Spectrum son:

- LEN
- STR\$
- VAL
- VAL\$
- CHR\$
- CODE

también pueden definirse funciones de cadena por el usuario con la sentencia «DEF FN».

PROGRAMA 1

```

10 REM *****
   *          *
   *  CURSO/BASIC  *
   *          *
   * *****      *
   *          *
   *  BUSQUEDA     *
   *          *
   * *****      *
   *          *

20 PRINT "Introduce una cadena
(max. 30)"
30 INPUT "> "; LINE a$
40 LET longitud=LEN a$
50 IF longitud<1 OR longitud>30
 THEN GO TO 30
60 CLS
70 PRINT "CADENA: "
80 PRINT a$
90 PRINT "LONGITUD: "; longitud
100 PRINT "CARACTER A BUSCAR: "

110 INPUT "> "; LINE b$
120 IF b$="" THEN GO TO 110
130 LET b$=b$(1)
140 PRINT b$
150 LET contador=0
160 FOR x=1 TO longitud
170 LET c$=a$(x)
180 IF c$<>b$ THEN PRINT c$; : G
O TO 205
190 LET contador=contador+1
200 PRINT FLASH.1;c$;
205 PAUSE 10
210 NEXT x
220 PRINT "
230 IF contador=0 THEN PRINT "C
aracter no encontrado." : GO TO 2
50
240 PRINT "Caracter encontrado
" : contador; " veces."
250 PRINT #0; "Quiere buscar otr
o caracter S/N"

260>PAUSE 0
270 LET i$=INKEY$
280 IF i$="S" OR i$="s" THEN CL
S : GO TO 70
290 IF i$="N" OR i$="n" THEN ST
OP
300 GO TO 260
  
```

PROGRAMA 2

```

10 REM *****
   *          *
   *  CURSO/BASIC  *
   *          *
   * *****      *
   *          *
   *  INSERTAR     *
   *          *
   * *****      *
   *          *

20 PRINT "Introduce una cadena
(max. 30)"
30 INPUT "> "; LINE a$
40 LET longitud=LEN a$
50 IF longitud<1 OR longitud>30
 THEN GO TO 30
60 CLS
70 PRINT "CADENA: "
80 PRINT a$
100 PRINT "CADENA A INSERTAR: "
110 INPUT "> "; LINE b$

120 IF b$="" THEN GO TO 110
130 IF LEN b$>30 THEN GO TO 110
140 PRINT b$
150 PRINT "POSICION: ";
160 INPUT "> "; LINE n$
170 LET posicion=VAL n$
180 IF posicion<1 OR posicion>l
ongitud THEN GO TO 160
185 PRINT posicion
190 LET c$=a$(1 TO posicion)
200 LET fin=longitud-posicion
210 IF fin=0 THEN LET d$=""; GO
TO 230
220 LET d$=a$(posicion+1 TO )
230 LET a$=c$+b$+d$
240 PRINT c$;
250 FOR x=1 TO LEN b$
260 PRINT b$(x);
265 PAUSE 10
270 NEXT x
280 PRINT d$
285 LET longitud=LEN a$

285>LET longitud=LEN a$
290 PRINT #0; "Mas caracteres a
insertar S/N"
300 PAUSE 0; LET i$=INKEY$
310 IF i$="S" OR i$="s" THEN GO
TO 60
320 IF i$="N" OR i$="n" THEN ST
OP
330 GO TO 300
  
```

LEN

Acceso al teclado

LEN



MODO E

SCREEN \$

Definición

La función «LEN» retorna un valor equivalente al número de caracteres de una cadena; hay que tener presente que los espacios en blanco también cuentan.

La estructura general de esta sentencia es:

| FUNCION | ARGUMENTO |
|---------|---------------------|
| LEN | expresión de cadena |

Ejemplos:

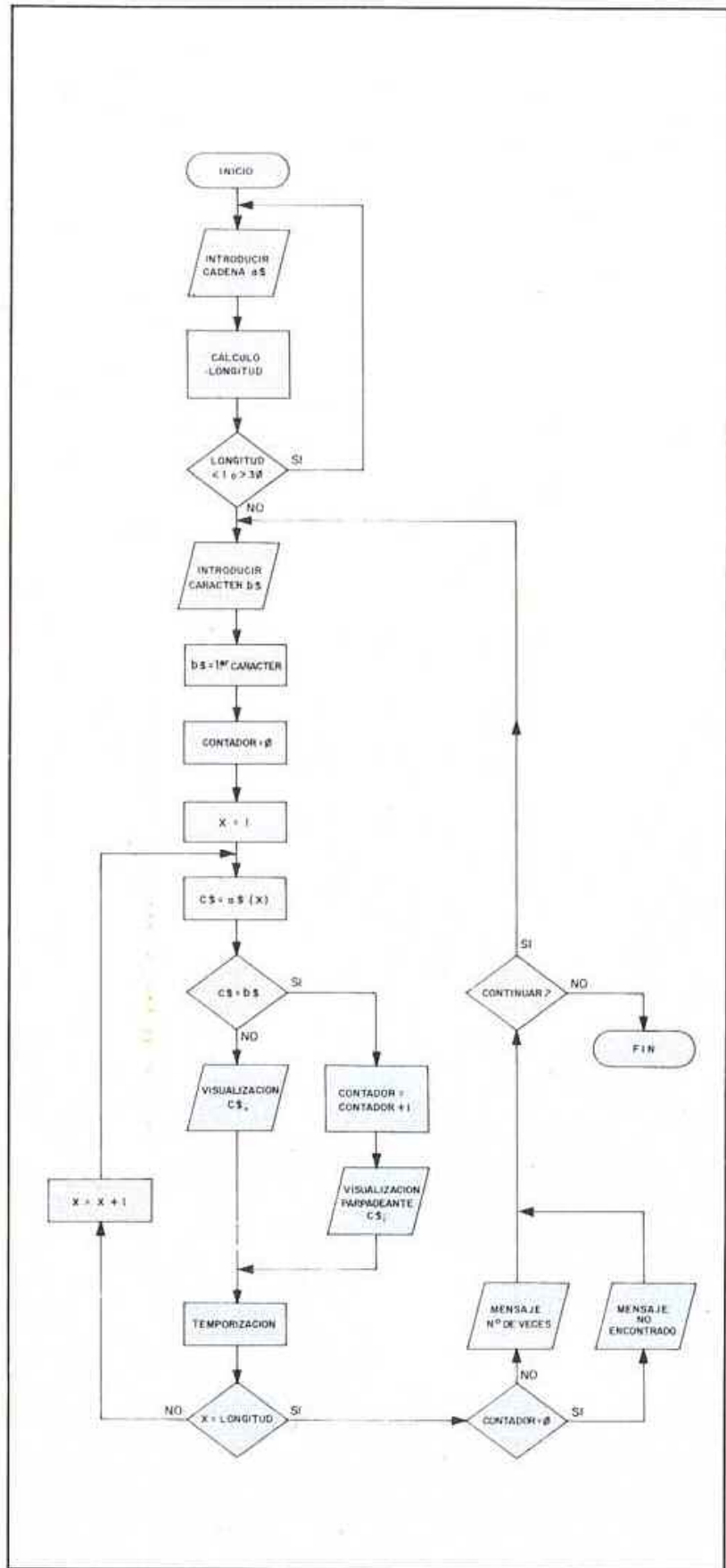
- PRINT LEN "BASIC"
- PRINT LEN a\$
- LET C\$ = LEN (b\$ + j\$)
- LET X\$ = LEN a\$ (3 TO)

El siguiente programa es un ejemplo de utilización de la sentencia «LEN»:

```

10 REM *****
   * FUNCION LEN *
   *****
20 PRINT " Introduce tu nombre
 y dos ape- " (1idos."
30 INPUT " " : LINE a$
40 LET longitud=LEN a$
50 FOR n=1 TO longitud
60 IF a$(n) = " " THEN LET longi-
 tud=longitud-1
70 NEXT n
75 CLS
80 PRINT "Tu nombre es: " a$
90 PRINT "Y tiene " longitud ;
 caracteres."

```



Estructura programa «BUSQUEDA».

éste, calcula la cantidad total de caracteres que forman el nombre introducido por el teclado. El bucle formado por las líneas «50» a «70» detecta los espacios en blanco y son restados de la longitud total.

Cuando se evalúa una cadena vacía, el valor retornado es igual a «0».

STR\$

Acceso al teclado

STR \$



MODO E

[

Definición

Asigna a una variable de cadena el resultado de una expresión numérica. Su estructura general es:

| FUNCIÓN | ARGUMENTO |
|---------|--------------------|
| STR\$ | expresión numérica |

Ejemplos:

- PRINT STR\$ 7
- PRINT STR\$ X
- LET n = STR\$ (7 * 12)
- LET x = STR\$ (v - n)

Veamos el siguiente programa:

```
10 REM *****
      *          *
      *  CURSO/BASIC  *
      *          *
      * *****      *
      *          *
      *   ANULAR      *
      *          *
      * *****      *
*****
```

```
20 INPUT "Valor numerico >>> "
numero
30 LET b$=STR$ numero
40 LET a$="Valor introducido"
45 CLS
50 PRINT a$+b$
60 GO TO 20
```

En la línea «30» se asigna a la variable de cadena «b\$» el valor de la variable numérica «numero», de esta manera puede realizarse la concatenación de la línea «50».

VAL

Acceso al teclado

VAL



MODO M

VAL \$

PROGRAMA 3

```
10 REM *****
      *          *
      *  CURSO/BASIC  *
      *          *
      * *****      *
      *          *
      *   ANULAR      *
      *          *
      * *****      *
*****
20 PRINT "Introduce una cadena
(max. 30)"
30 INPUT "> "; LINE a$
40 LET longitud=LEN a$
50 IF longitud<1 OR longitud>3
0 THEN GO TO 30
60 CLS
70 PRINT "CADENA: ""
80 PRINT a$
150 PRINT "POSICION CARACTER A
BORRAR:"
160 INPUT "> "; LINE n$
```

```
170 LET posicion=VAL n$
180 IF posicion<1 OR posicion>
ongitud THEN GO TO 160
185 PRINT posicion
190 LET c$=a$(1 TO posicion-1)
200 LET fin=longitud-posicion
210 IF fin=0 THEN LET d$="": GO
TO 230
220 LET d$=a$(posicion+1 TO )
230 LET a$=c$+d$
240 PRINT a$
260 LET longitud=LEN a$
290 PRINT #0;"Mas caracteres a
borrar S/N"
300 PAUSE 0: LET i$=INKEY$
310 IF i$="S" OR i$="s" THEN GO
TO 60
320 IF i$="N" OR i$="n" THEN ST
OP
330 GO TO 300
```

CADENA:
MICROHOBBY SEMANAL

LONGITUD: 18

CARACTER A BUSCAR: M

MICROHOBBY SEMANAL

Caracter encontrado 2 veces.

CADENA:
MICROHY SEMANAL

CADENA A INSERTAR:

OBB

POSICION: 6

MICROHOBBY SEMANAL

Búsqueda de caracteres.

Definición

Evalúa una cadena cuyo contenido son números y operadores matemáticos, y el resultado lo asigna a una variable numérica.

Su estructura general es:

| FUNCION | ARGUMENTO |
|---------|---------------------|
| VAL | expresión de cadena |

Ejemplos:

- PRINT VAL "23"
- LET a = VAL a\$
- PRINT VAL ("24" + "/"2")
- LET n = VAL b\$ + c\$

Para realizar la evaluación, la función «VAL» suprime las comillas de la cadena.

El siguiente programa realiza las cuatro operaciones básicas entre dos números, éstos están asignados a las variables «a\$» y «b\$». El signo de la operación depende de la letra minúscula introducida por teclado (s – suma, r – resta, m – multiplicación y d – división) y queda asignada a la variable «C\$».

Para calcular el resultado se utiliza la función «VAL» en la línea 100.

Inserción de cadenas.

Ejemplos:

- PRINT VAL\$ """BASIC"""
- LET a\$ = VAL\$ "b\$"
- PRINT VAL\$ "a\$ + c\$"
- LET b\$ = VAL\$ "C\$" + "PEPE"

Conversiones de código

Hay dos funciones de cadena que permiten realizar cualquier conversión entre los códigos «ASCII» y «DECIMAL», éstas son:

- CHR\$
- CODE

Ver tabla del Código ASCII en la pag. 41.

CHR\$

Acceso al teclado

CHR\$



Definición

La función «CHR\$» transforma un número decimal en

```

10 REM *****
   * FUNCION VAL *
   * *****
20 INPUT "numero 1 >>> ";a$
25 CLS
30 INPUT "numero 2 >>> ";b$
40 INPUT "Operacion (s/r/m/d)
>>> ";o$
50 IF o$="s" THEN LET c$="+":
GO TO 100
60 IF o$="r" THEN LET c$="-":
GO TO 100
70 IF o$="m" THEN LET c$="*":
GO TO 100
80 IF o$="d" THEN LET c$="/":
GO TO 100
90 GO TO 40
100 LET resultado=VAL (a$+c$+b$)

110 PRINT a$;" ";c$;" ";b$;" =
";resultado
120 GO TO 20

```

VAL\$

Acceso al teclado

VAL



VAL\$ Definición

La función «VAL\$» evalúa variables, constantes o expresiones de cadena siempre que estén encerradas entre paréntesis; el valor retornado es otra cadena.

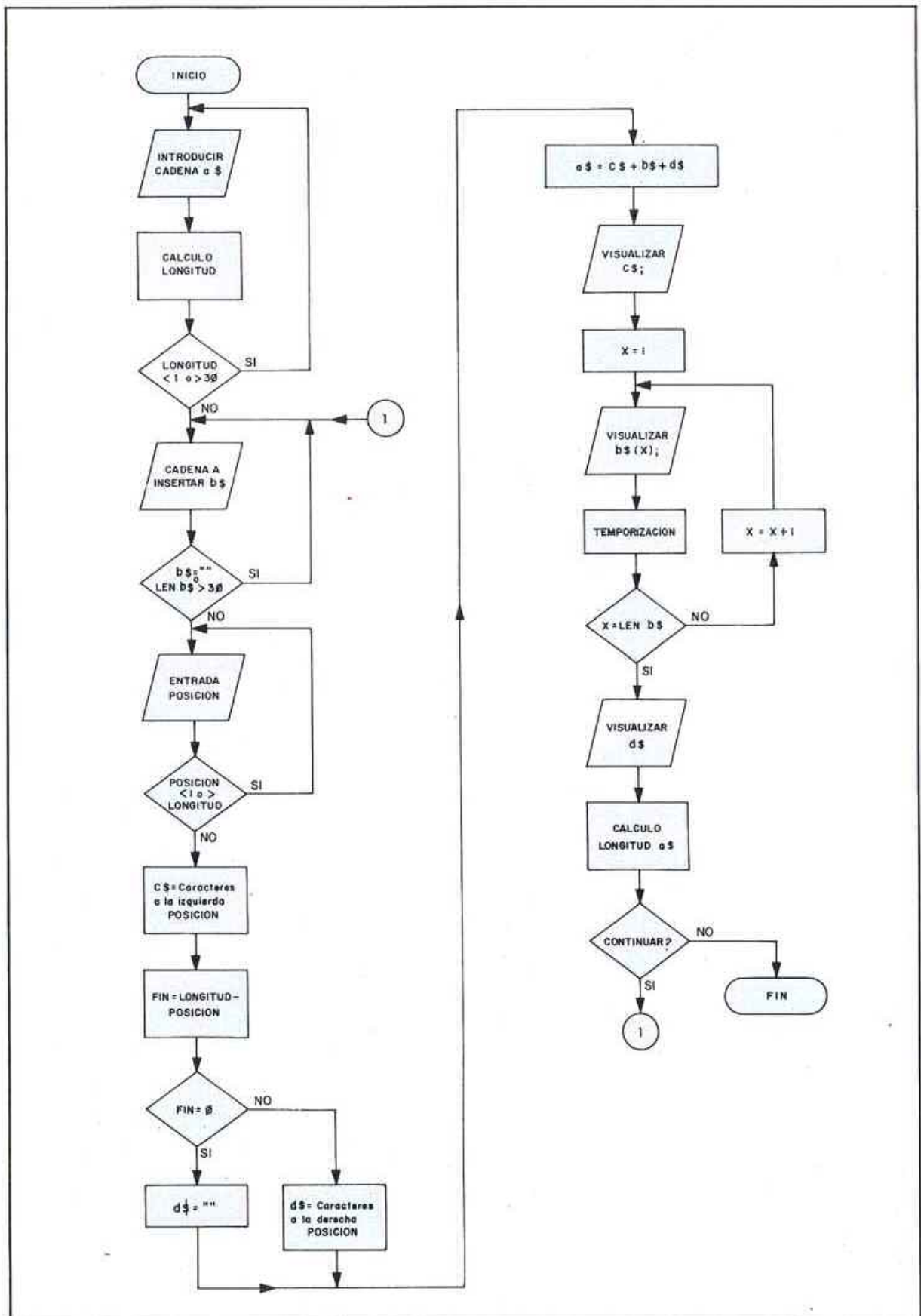
Su estructura general es:

| FUNCION | ARGUMENTO |
|---------|-----------------------|
| VAL\$ | (expresión de cadena) |

MODULO E

SYMBOL
SHIFT

MODULO E



Estructura programa «INSERTAR».

| FUNCIÓN | ARGUMENTO |
|---------|---------------------|
| CODE | expresión de cadena |

Ejemplos:

- PRINT CODE «A»
- LET n = CODE a\$
- PRINT CODE b\$(X)
- LET X 3 CODE "RAIZ"

Al retornar «CODE» el código del primer carácter, los siguientes comandos directos darán el mismo resultado (65):

- PRINT CODE "A"
- PRINT CODE "Alazan"
- PRINT CODE "Almería"

La palabra clave «CODE» también es utilizada junto con «LOAD», «SAVE» y «VERIFY», pero esta particularidad será vista en otro capítulo.

En este programa se visualizan las letras mayúsculas de la «A» a la «Z» con su correspondiente código decimal. Observe que la función «CODE» se utiliza para controlar el bucle:

```
10 FOR n = CODE "a" TO CODE "z"
20 PRINT n; " "; CHR$(n)
30 NEXT n
```

En este otro, debe introducirse una cadena como máximo de «15» caracteres: la instrucción 70 permite centrar la visualización de dicha cadena. El bucle compuesto por las líneas «80» a «100» permite representar el código decimal de cada carácter, ya que se utiliza la función «CODE» junto con la fragmentación de la cadena.

Cuando el argumento es una cadena vacía ("") la función «CODE» retorna el valor «0».

```
10 REM *****
      * FUNCION CODE *
      *****
20 PRINT AT 21,0;"Introduce un
a cadena (Max. 15)"
30 INPUT "":a$
40 LET longitud=LEN a$
50 IF longitud>15 OR longitud<1
5 THEN GO TO 30
60 CLS
70 PRINT AT 0,(16-longitud)/2;
a$
80 PRINT " "
90 FOR n=1 TO longitud
90 PRINT "a$(n);"
CODE a$(n)
100 NEXT n
110 GO TO 20
```

Funciones definidas de cadena

De la misma manera que el usuario puede definir sus propias funciones numéricas, también puede hacerlo con las de cadena.

La estructura general es:

| SENTENCIA | ARGUMENTO |
|-----------|--------------------------------|
| DEF FN | letra \$ (variables) = función |

Ejemplos:

- DEF FN a\$(b) = CHR\$(b) + " : "
- DEF FN b\$(a\$, c\$) = a\$ + c\$
- DEF FN c\$(a\$, n) = a\$(n)
- DEF FN d\$(X) = STR\$(X)

La llamada de las funciones sigue la estructura:

```
FN letra$(parámetros)
```

Ejemplos:

- PRINT FN a\$(59)
- PRINT FN b\$("Pepe", "Juan")
- PRINT FN c\$("Camión", 3)
- PRINT FN d\$(4 * 56)

Errores

Dos errores típicos en el

manejo de cadenas son:

a) C Nonnense in BASIC

Ocurre en las siguientes situaciones:

```
LET a$ = "1374a2"
PRINT VAL a$
```

ya que la variable «a\$» contiene un carácter no numérico. Sin embargo

```
LET a$ = "a13742"
PRINT VAL a$
```

da el error:

```
2 variable not found
```

ya que la función «VAL» al suprimir las comillas y detectar que el primer carácter es una letra, el sistema operativo interpreta que se trata de una variable numérica y al no encontrarla visualiza el error anterior. Observe la diferencia con las siguientes instrucciones:

```
LET valor = 30
LET a$ = "valor"
PRINT VAL a$
```

Con la función «VAL\$» también se produce el error «C»:

```
LET c$ = "oasis"
PRINT VAL$ c$
```

ya que o bien se modifica:

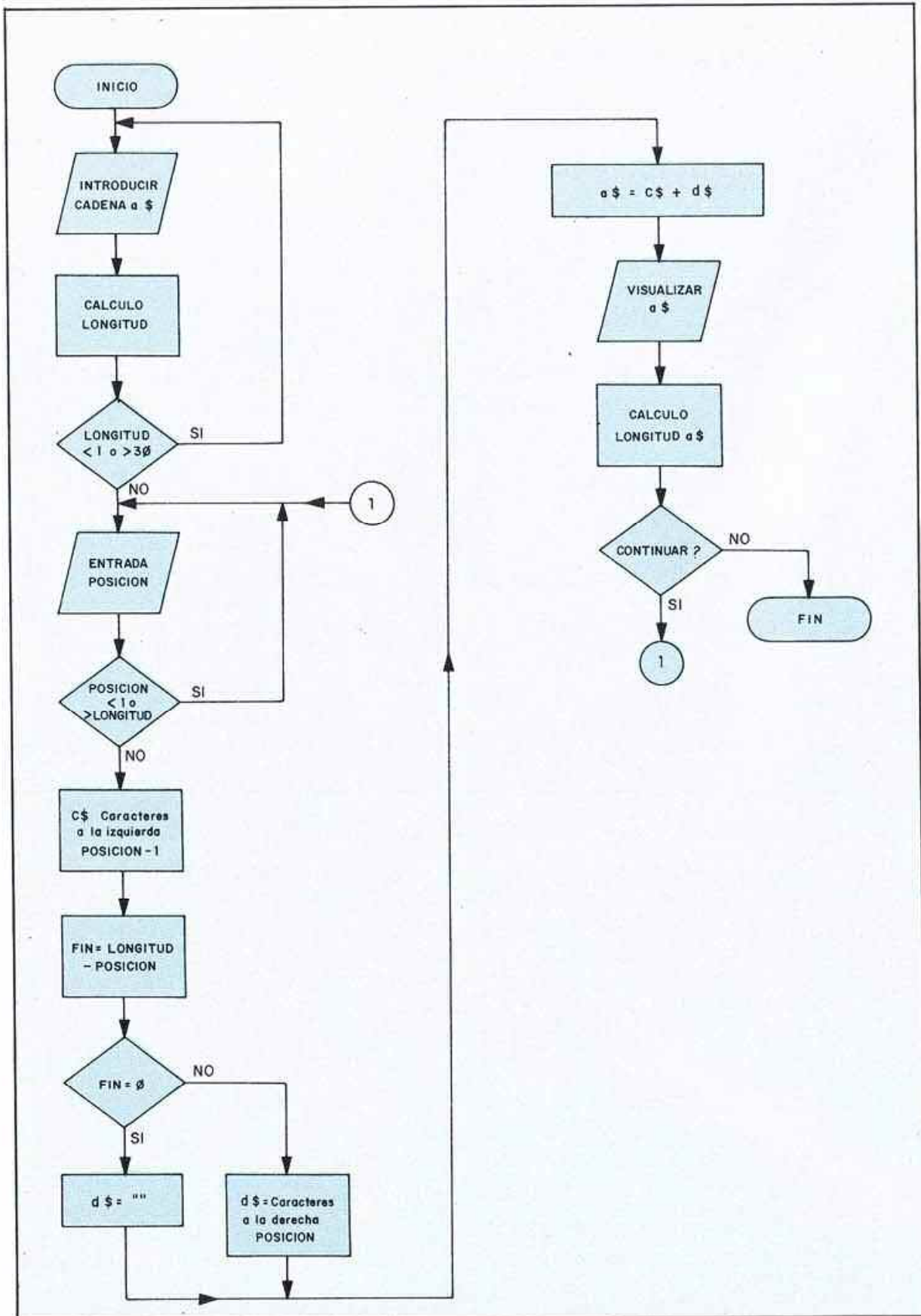
```
LET c$ = ""oasis""
```

o por el contrario

```
PRINT VAL$ "C$"
```

b) 3 Subscript wrong.

Sucede cuando el índice



Estructura programa «ANULAR».

de una fragmentación tiene un valor mayor que la longitud total de la cadena.

Ejemplo:

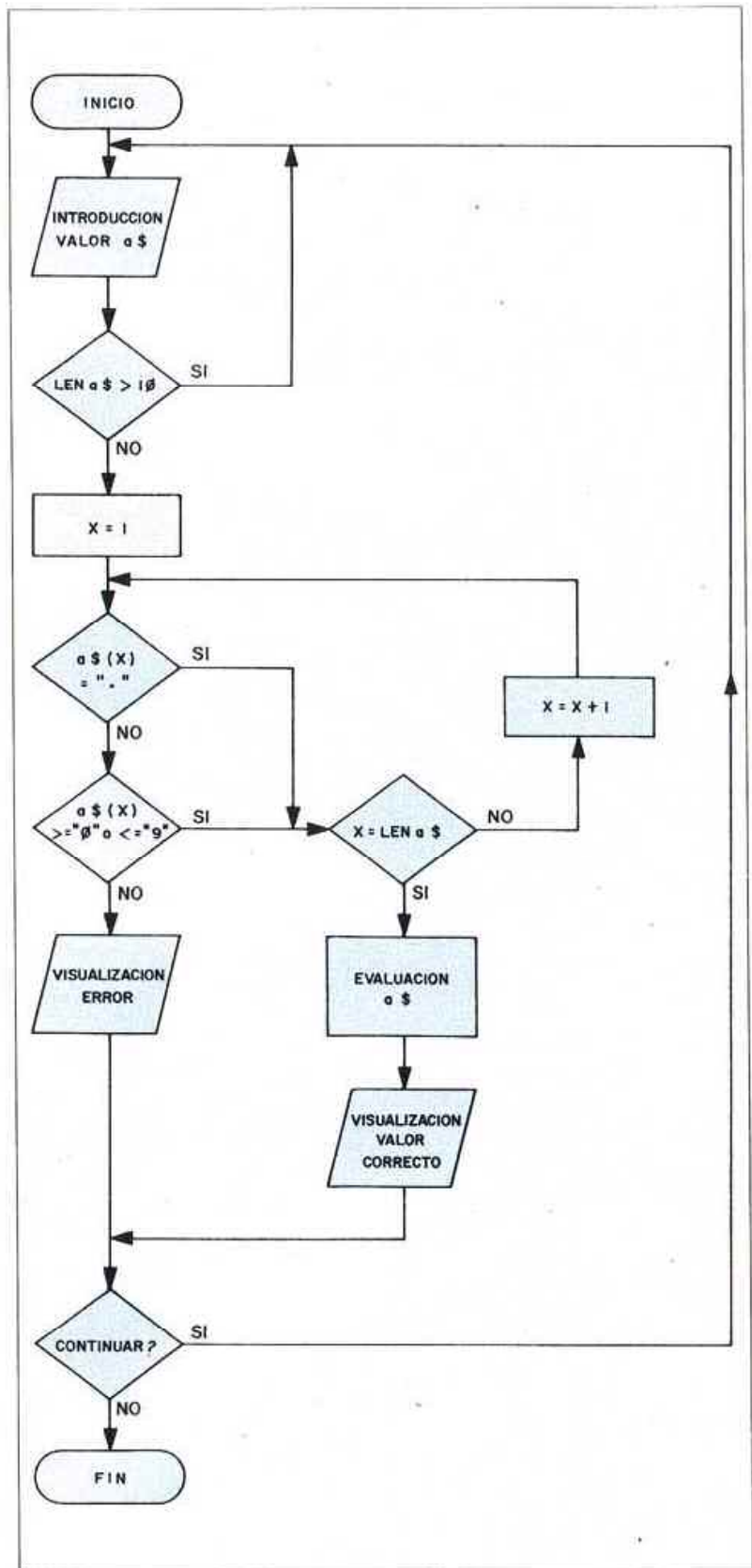
```
LET a$ = "sandra"
PRINT a$ (7)
```

Programa

El programa n.º «1» (BUSQUEDA) permite introducir una cadena con un máximo de 30 caracteres y, posteriormente, buscarnos un caracter determinado; si lo encuentra éste empezará a parpadear.

La estructura general es:

- 10 : nombre del programa.
- 20 : Mensaje explicativo.
- 30 : Entrada de la cadena (a\$).
- 40 : Cálculo de la longitud de la cadena.
- 50 : Comprobación de la longitud.
- 60 : Borrado pantalla.
- 70-90 : Visualización de la cadena y su longitud.
- 100-110 : Entrada del caracter a buscar (b\$).
- 120 : Comprobación de que no es una cadena vacía.
- 130 : Asignación a la cadena b\$ de su primer caracter.
- 140 : Visualización del caracter a buscar.
- 150 : Inicialización de la variable (contador) a cero.
- 160 : Comienzo del bucle de búsqueda.
- 170 : Asignación a la variable C\$ del caracter muestreado.
- 180 : Verificación si el caracter muestreado es igual al buscado; si no lo es, se visualiza el caracter.
- 190-200 : Si el caracter es igual se incrementa la variable



Estructura programa «IMPULS».

- (contador) en una unidad y se visualiza parpadeante (FLASH) el caracter.
- 205 : Temporización.
- 210 : Incremento de la variable de control del bucle.
- 230 : Si la variable (contador) es igual a cero se visualiza el mensaje: (Caracter no encontrado).
- 240 : Visualización del contenido de la variable (contador) en el caso de que sea distinta de cero.
- 250-300 : Análisis de las teclas (S) y (N) para buscar otro caracter o no.

El programa n.º 2 (INSERTAR) inserta un caracter o una serie de ellos dentro de otra cadena a partir de la posición que se quiera.

La estructura general del programa es:

- 10 : Comentario con el nombre del programa.
- 20-30 : Entrada de la cadena (a\$).
- 40 : Cálculo de su longitud.
- 50 : Verifica si está dentro de los límites.
- 70-80 : Visualización de la cadena.
- 100-110 : Entrada de la cadena a insertar (h\$).
- 120 : Comprueba que b\$ no es una cadena vacía.
- 130 : Comprueba que no tiene más de treinta caracteres.
- 140 : Visualización de la cadena a insertar.
- 150-170 : Entrada de la posición.
- 180 : Comprueba que la posición no es cero, ni mayor que la longitud de la cadena a\$.
- 185 : Visualización de la posición.
- 190 : Asignación a la variable c\$ de los caracteres situados a la izquierda de (posición).
- 200 : Calcula cuantos caracteres hay situados a la derecha de (posición).

- 210 : Si no hay ninguno, asigna a la variable d\$ una cadena vacía.
- 220 : Si hay, asigna a d\$ los caracteres situados a la derecha de (posición).
- 230 : Concatenación de las variables c\$, b\$ y d\$.
- 240 : Visualización de c\$.
- 250-270 : Visualización, a continuación de los caracteres, de la variable b\$ (temporizados).
- 280 : Visualización de la variable d\$.
- 285 : Nuevo cálculo de la longitud de la cadena.
- 290-330 : Análisis de las teclas (S) y (N) para insertar otra cadena o no.

El programa n.º 3 (ANULAR) borra el caracter indicado por su posición, dentro de una cadena que tenga «30» caracteres como máximo.

La estructura del programa es:

- 10 : Nombre del programa.
- 20-30 : Entrada de la cadena a\$.
- 40 : Cálculo de su longitud.
- 50 : Comprueba si está dentro de los límites (1-30).
- 60 : Borrado de la pantalla.
- 70-80 : Visualización de la cadena a\$.
- 150-170 : Entrada de la posición a borrar.
- 180 : Comprueba que no se encuentra situada fuera de la cadena.
- 185 : Visualización de la posición.
- 190 : Asignación de los caracteres situados a la izquierda de (posición) a la variable c\$.
- 200 : Calcula si el caracter a borrar es el último.
- 210 : Si lo es, asigna a la variable d\$ el valor de una cadena vacía.
- 220 : Si no lo es, asigna a d\$ los caracteres situados a la derecha de posición.

- 230 : Asigna a la variable a\$ el nuevo valor (c\$ + d\$ sin el caracter borrado).
- 240 : Visualización de la variable a\$.
- 280 : Actualización de la longitud de a\$.
- 290-330 : Análisis de las teclas (S) y (N) para anular un nuevo caracter o no.

El programa n.º (INPUT) detecta si dentro de una sentencia «INPUT» se ha introducido un caracter numérico; de esta manera se consigue que por error no aparezca el conocido mensaje:

2 Variable not found

La estructura del programa es:

- 10 : Comentario con el nombre del programa.
- 20 : Entrada del valor numérico asignado a una variable de cadena (a\$).
- 30 : Comprobación de que la longitud no excede de 10 caracteres.
- 40 : Comienzo del bucle de análisis.
- 50 : Si el código del caracter es un (punto) continúa el análisis de caracteres.
- 60 : Si el código pertenece a un número comprendido entre (0) y (9) se continúa con el análisis.
- 70 : El caracter al no ser ni un punto ni un número hace que la ejecución del bucle se interrumpa.
- 80 : Incremento de la variable de control del bucle.
- 90-100 : Si todos los caracteres son correctos se evalúa la variable (valor) y se visualiza su contenido.
- 120-130 : Visualización de un mensaje de error y del primer caracter no numérico introducido en la cadena.
- 140-190 : Análisis de las teclas (S) y (N) para introducir un nuevo valor o no. ■

MATRICES

La *matriz*, *tabla* o *array*, es una estructura utilizada en programación que permite almacenar los datos (constantes o variables) de un programa.

Los elementos de una matriz tienen en común el mismo nombre y se diferencian en el subíndice, de esta manera se tiene el acceso directo a ellos.

Utilizando correctamente los subíndices pueden asignarse o leerse los contenidos de los elementos de una matriz.

Las matrices pueden ser de dos tipos:

- NUMERICAS
- DE CADENA

Dimensionado de matrices

Para reservar una serie de posiciones de memoria (tabla) es necesario, previamente, *dimensionar* una matriz; tarea que consiste en definir la cantidad de elementos de que va a constar así como su distribución; es decir, podríamos dimensionar una matriz, de 40 elementos, de varias formas:

- a) Una, en que los elementos están consecutivos.
- b) También podrían estar distribuidos como en una especie de «tablero de ajedrez», que tuviera «5» filas por «8» columnas o «10» filas por «4»

columnas, etc.

c) Otra solución sería distribuirlos en dos *planos* formados cada uno por dos «tablas» de 4 * 5.

d) etc.

Dependiendo de la distribución las matrices, reciben diversos nombres:

- UNIDIMENSIONALES
- BIDIMENSIONALES
- TRIDIMENSIONALES
- MULTIDIMENSIONALES

ADVERTENCIA

Las matrices antes de ser utilizadas deben estar «dimensionadas»

PROGRAMA 1

```

10 REM *****
   * MANEJO DE TABLAS *
   * *****
20 INPUT "Numero de elementos
>>> ", elementos
30 IF elementos <> INT elementos
THEN GO TO 20
40 IF elementos < 1 OR elementos
> 40 THEN GO TO 20
50 DIM a (elementos)
60 REM *****
   * ASIGNACION VALORES *
   * *****
65 RANDOMIZE
70 FOR n=1 TO elementos
80 LET contenido=INT (RND*100)
+1 90 LET a(n)=contenido
100 NEXT n
110 REM
  
```

```

*****
* VISUALIZACION *
* *****
120 FOR n=1 TO elementos
130 PRINT n;
132 IF n<10 THEN PRINT " ";
134 PRINT "> "; a(n);
140 NEXT n
150 REM *****
   * BUSQUEDA *
   * *****
160 INPUT "Valor a encontrar >>
> ", valor
170 IF valor <> INT valor THEN GO
TO 160
180 IF valor < 1 OR valor > 100 THE
N GO TO 160
190 LET veces=0
200 DIM b(elementos)
210 REM
  
```

```

*****
*
* EXPLORACION *
*
*****
220 FOR n=1 TO elementos
230 IF a(n)<>valor THEN GO TO 2
60
240 LET veces=veces+1
250 LET b(veces)=n
260 NEXT n
270 REM
*****
*
* VISUALIZACION *
*
*****
280 CLS
282 IF veces=0 THEN PRINT "Valor
r";valor;" no encontrado": GO T
0 350
290 PRINT "El valor: ";valor
300 PRINT "ha sido encontrado:
";veces;" veces"
310 PRINT "en las posiciones:"
320 FOR n=1 TO veces
330 PRINT "> ";b(n)
340 NEXT n
350 PRINT #0;"Otro valor (S/N)"
360 PAUSE 0
370 LET z$=INKEY$
380 IF z$="S" OR z$="s" THEN GO
TO 150
390 IF z$="N" OR z$="n" THEN GO
TO 410
400 GO TO 360
410 CLS
412 FOR n=1 TO elementos
414 PRINT n;
416 IF n<10 THEN PRINT " ";
420 PRINT "> ";a(n);
430 NEXT n
440 REM
*****
*
* PAR/IMPAR *
*
*****
450 PRINT #0;"Pulsa una tecla p
ara continuar"
460 PAUSE 0
470 CLS
480 LET par=0
490 LET impar=0
500 DIM c(elementos)
510 DIM d(elementos)
520 REM
*****
*
* BUSQUEDA *
*
*****
530 FOR n=1 TO elementos
540 LET division=a(n)/2
550 IF INT (division)*2=a(n) TH
EN GO TO 590
560 LET impar=impar+1
570 LET d(impar)=a(n)

```

```

580 GO TO 610
590 LET par=par+1
600 LET c(par)=a(n)
610 NEXT n
620 REM
*****
*
* VISUALIZACION *
*
*****
625 CLS
630 PRINT "Numeros pares encont
rados: ";par
640 FOR n=1 TO par
650 PRINT "> ";c(n);
660 NEXT n
670 PRINT #0;"Pulsa una tecla p
ara continuar"
680 PAUSE 0
685 CLS
690 PRINT "Numeros impares enco
ntrados: ";impar
700 FOR n=1 TO impar
710 PRINT "> ";d(n);
720 NEXT n
730 PRINT #0;"Vuelvo a visualiz
arlos (S/N)"
740 PAUSE 0
750 LET z$=INKEY$
760 IF z$="S" OR z$="s" THEN GO
TO 620
770 IF z$="N" OR z$="n" THEN GO
TO 790
780 GO TO 740
790 CLS
800 FOR n=1 TO elementos
802 PRINT n;
804 IF n<10 THEN PRINT " ";
810 PRINT "> ";a(n);
820 NEXT n
830 REM
*****
*
* ORDENA *
*
*****
840 PRINT #0;"Pulsa una tecla p
ara continuar"
850 PAUSE 0
860 CLS
870 FOR x=1 TO elementos-1
880 LET menor=x
890 FOR y=x+1 TO elementos
900 IF a(menor)>a(y) THEN LET m
enor=y
910 NEXT y
920 LET cambio=a(x)
930 LET a(x)=a(menor)
940 LET a(menor)=cambio
950 PRINT x;
960 IF x<10 THEN PRINT " ";
970 PRINT "> ";a(x);
980 NEXT x
990 PRINT elementos;
1000 IF elementos<10 THEN PRINT
" ";
1010 PRINT "> ";a(elementos)
1020 PAUSE 0

```

DIM

Acceso al teclado

DATA



MODOS **K**

Tipo de sentencia

Comando de programación.

Definición

La sentencia «DIM» permite dimensionar tanto las matrices de tipo numérico como las de cadena.

Matrices numéricas

La estructura del dimensionado de matrices numéricas es:

| SENTENCIA | ARGUMENTO |
|-----------|--------------------------|
| DIM | letra (lista de valores) |

Ejemplos:

a) Matriz «unidimensional».

DIM X (20)

La matriz «X» se dimensiona de 20 elementos consecutivos. El subíndice varía, por tanto, entre 1 y 20:

X (1), X (2)... X (20)

b) Matriz «bidimensional».

DIM J (5, 7)

La matriz «J» se dimensiona de 35 elementos distribuidos en 5 filas por 7 columnas. Los subíndices varían desde:

J (1, 1), J (1, 2)... J (5, 6), J (5, 7)

c) Matriz «tridimensional».

DIM N (3, 4, 2)

Matriz llamada «N» constituida por 24 elementos distribuidos en tres planos de 4 filas por 2 columnas. Los subíndices varían entre:

N (1, 1, 1), N (1, 1, 2)... N (3, 4, 2)

Los subíndices también pueden estar constituidos por variables y expresiones de tipo numérico.

Ejemplos:

- DIM C (7 * 3, 6)
- DIM P (Valor)
- DIM F (SIN cr, 5)
- DIM J (20 * X, SQR 9)

Como habrá podido observar, el nombre de una matriz está formado por una sola letra que la distingue de las demás, por tanto, no puede haber dos matrices con el mismo nombre aunque estén dimensionadas de distinta forma, ya que al dimensionar una matriz se borra cualquier otra que tuviera el mismo

| MATRIZ " B " | |
|--------------|--|
| 1 | |
| 2 | |
| 3 | |
| 4 | |
| 5 | |
| 6 | |
| 7 | |
| 8 | |
| 9 | |
| 10 | |
| 11 | |
| 12 | |
| 13 | |
| 14 | |
| 15 | |
| 16 | |
| 17 | |
| 18 | |

SUBINDICE

Matriz unidimensional B(18), elemento B(10).

nombre. Por el contrario, puede haber una variable numérica con el mismo nombre que una matriz, ya que, el sistema operativo distingue una de la

otra, por que, en su caso se utilizan los subíndices y en el otro no.

Cuando se dimensiona una matriz numérica el contenido

de sus elementos queda inicializado el valor cero.

Ejemplo:

```
10 DIM X(21)
20 FOR n = 1 TO 21
30 PRINT n, X (n)
40 NEXT n
```

Asignación y visualización

Para asignar un valor a un elemento determinado de una matriz, es necesario *posicionar* correctamente los subíndices o punteros. Por ejemplo:

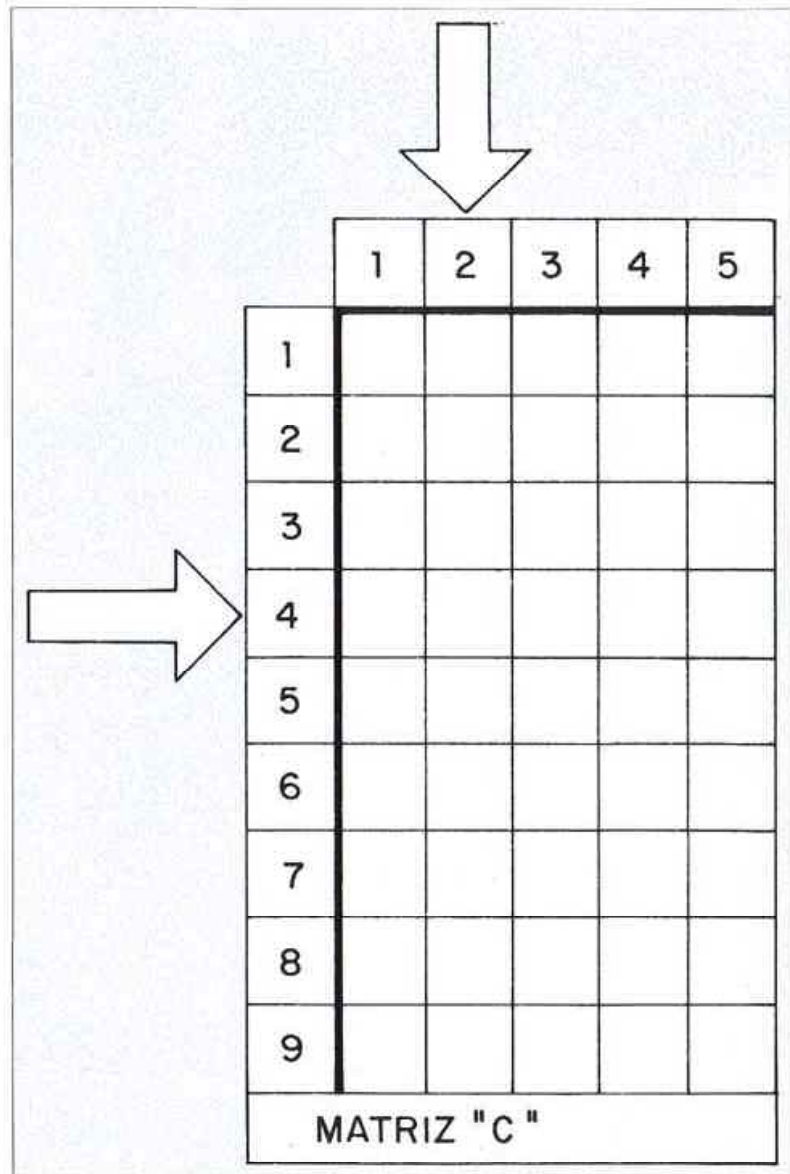
a) Asignar el valor 30 a elemento 7 de una matriz «Z».

```
LET Z(7) = 30
```

b) Asignar al elemento situado en las coordenadas «y = 3» y «x = 5» de una matriz «J», el valor de la variable «patron».

```
LET J (3,5) = patron
```

Hay que tener presente que los punteros deben estar comprendidos dentro de los valores especificados al dimensionar la matriz.



Matriz bidimensional C(9,5), elemento C(4,2).

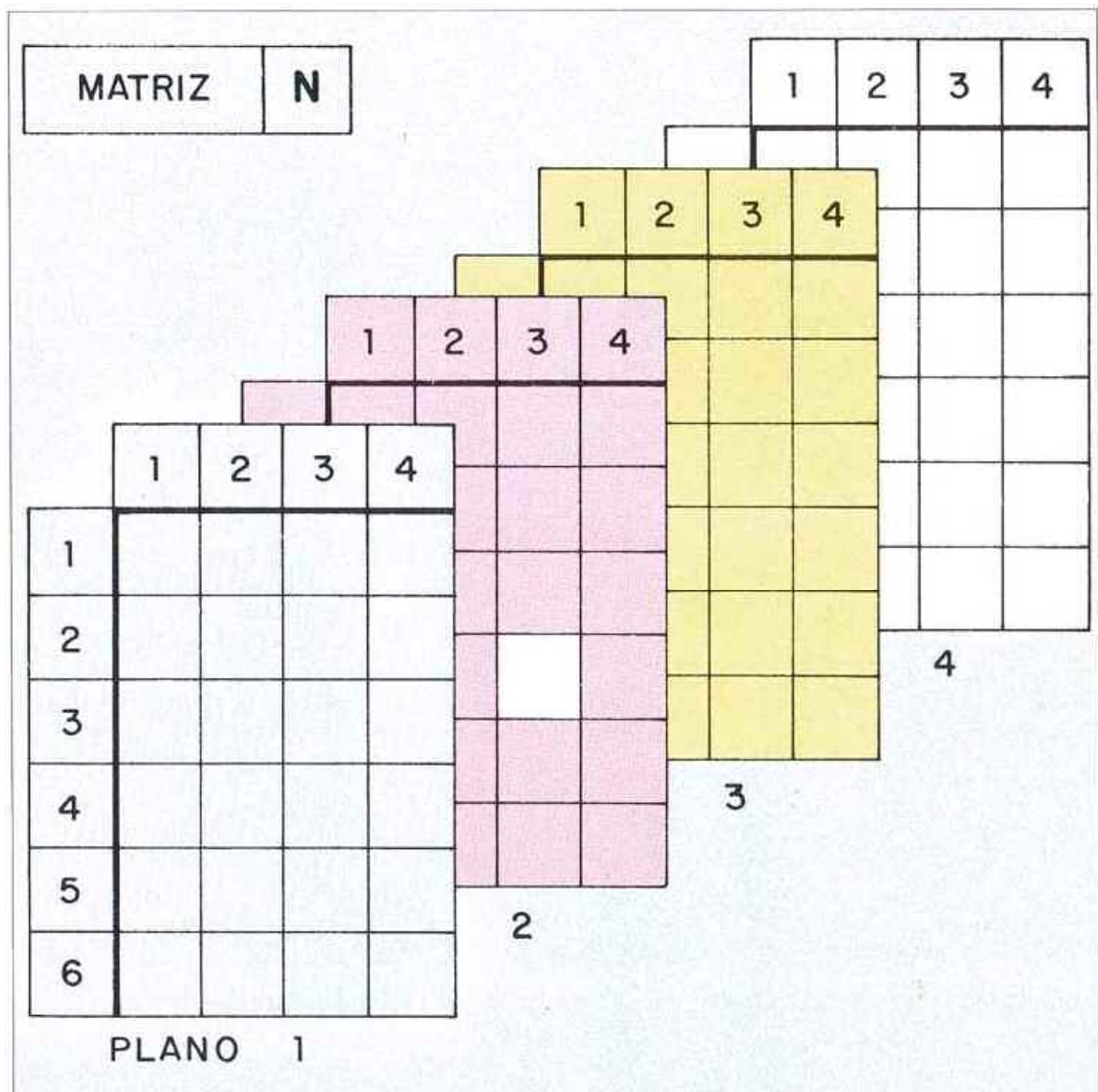
```
1 > 53
2 > 20
3 > 31
4 > 81
5 > 56
11 > 10
13 > 34
15 > 34
17 > 42
19 > 9
```

```
2 > 41
4 > 36
6 > 54
8 > 57
10 > 85
12 > 4
14 > 18
16 > 92
18 > 49
20 > 3
```

```
21 > 15
23 > 7
25 > 55
27 > 18
29 > 98
31 > 53
33 > 71
35 > 18
37 > 7
39 > 64
```

```
22 > 19
24 > 11
26 > 62
28 > 70
30 > 88
32 > 88
34 > 88
36 > 87
38 > 92
40 > 70
```

Programa 1. Generación aleatoria.



Matriz tridimensional $N(4,6,4)$, elemento $N(2,4,3)$.

Recíprocamente, puede asignarse a una variable numérica el contenido de un determinado elemento. Ejemplos:

a) Asignar a la variable «número» el contenido del elemento 3 de la matriz «h».

```
LET número = h (3)
```

b) Asignar a la variable «cálculo» la raíz cuadrada del elemento situado en las coordenadas 7 y 2 de la matriz «p»

```
El valor: 18
ha sido encontrado: 3 veces
en las posiciones:
```

```
> 14
> 27
> 35
```

Programa 1. Búsqueda.

LET cálculo = SQR p (7,2)

La visualización de elementos se realiza de forma similar a la asignación:

- PRINT Z (7)
- PRINT J (3,5)

Cuando se desea asignar o visualizar todos los elementos de una matriz, son muy útiles los bucles «FOR...NEXT».

Ejemplo:
- Dimensionar una matriz de 10 elementos, asignarles valor y posteriormente visualizarlos:

```

10 REM *****
   UNIDIMENSIONAL
*****
20 DIM a(10)
30 REM *****
   ASIGNACION
*****
40 FOR x=1 TO 10
50 INPUT "Valor ";(x); " ";:let
elemento
60 LET a(x)=elemento
70 NEXT x
80 REM *****
   VISUALIZACION
*****
90 FOR x=1 TO 10
100 PRINT "Elemento ";(x);": ";a(x)
110 NEXT x
120 REM *****

```

```

*****
   VISUALIZACION
*****
90 FOR x=1 TO 10
100 PRINT "Elemento ";(x);": ";a(x)
110 NEXT x

```

Para asignar valores a una matriz bidimensional, es necesario utilizar dos bucles anidados, uno que «barra» las filas y otro las columnas.

Ejemplo:
- Idem. matriz «n» de 4 * 3 elementos.

```

10 REM *****
   BIDIMENSIONAL
*****
20 DIM n(4,3)
30 REM *****
   ASIGNACION
*****
40 FOR y=1 TO 4
45 FOR x=1 TO 3
50 INPUT "Valor ";(y);(x);": ";:let
elemento
60 LET n(y,x)=elemento
70 NEXT x
75 NEXT y
80 REM *****
   VISUALIZACION
*****
90 FOR y=1 TO 4
95 FOR x=1 TO 3
100 PRINT "Elemento ";(y);(x);": ";n(y,x)
110 NEXT x
120 NEXT y

```

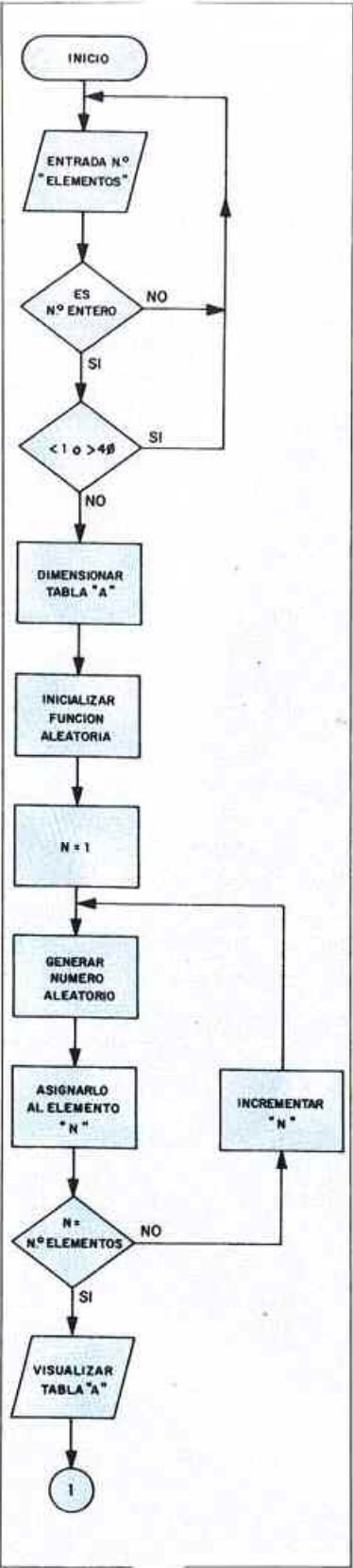
Numero pares encontrados: 19

| | |
|------|------|
| > 20 | > 36 |
| > 54 | > 56 |
| > 10 | > 4 |
| > 34 | > 18 |
| > 34 | > 92 |
| > 42 | > 62 |
| > 18 | > 98 |
| > 88 | > 68 |
| > 18 | > 92 |
| > 64 | |

Numero impares encontrados: 21

| | |
|------|------|
| > 53 | > 41 |
| > 31 | > 81 |
| > 67 | > 85 |
| > 49 | > 9 |
| > 3 | > 15 |
| > 19 | > 7 |
| > 11 | > 55 |
| > 79 | > 53 |
| > 71 | > 85 |
| > 87 | > 7 |
| > 79 | |

Programa 1. Pares/impares.



Manejo tablas Dimensionado y asignación

Manejo de tablas

Las tablas de datos o matrices son quizás uno de los recursos más utilizados a la hora de elaborar un programa, por este motivo es imprescindible que el programador conozca a fondo su manejo.

El programa n.º «1» realiza diversos tratamientos con las matrices, por este motivo va a ser explicado con bastante detalle.

Las funciones básicas que realiza el programa son:

- Dimensionar una matriz en función de una variable.
- Asignar a la matriz valores aleatorios.
- Buscar un número determinado y visualizar cuántas veces aparece y en qué posiciones.
- Calcular cuáles son los números pares e impares y visualizarlos en dos listas separadas.
- Ordenar la tabla, generada aleatoriamente, de menor a mayor.

a) Dimensionado y asignación:

La línea 20 del programa permite introducir un número por teclado, éste queda asignado a la variable «elementos»; este número hace referencia a la cantidad de elementos de que va a constar nuestra tabla.

La línea 30 comprueba que el número introducido es entero, es decir, que no tiene decimales, de no ser así, el programa vuelve a pedir que introduzcamos otro número.

En la 40 se comprueba que el número está comprendido dentro del rango «1» a «40».

El dimensionado de la tabla se realiza en la línea 50. El nombre asignado es «a» y con

```
10 REM *****
* MANEJO DE TABLAS *
* *****
20 INPUT "Numero de elementos
>>> "; elementos
30 IF elementos<>INT elementos
THEN GO TO 20
40 IF elementos<1 OR elementos
>40 THEN GO TO 20
50 DIM a(elementos)
60 REM *****
* ASIGNACION VALORES *
* *****

65 RANDOMIZE
70 FOR n=1 TO elementos

80 LET contenido=INT (RND*100)
+1
90 LET a(n)=contenido
100 NEXT n
110 REM *****
* VISUALIZACION *
* *****

120 FOR n=1 TO elementos
130 PRINT n;
132 IF n<10 THEN PRINT " ";
134 PRINT "> ";a(n),
140 NEXT n
```

una dimensión definida en la variable «elementos».

En la línea 65 se realiza la inicialización de la función aleatoria, ésta depende del tiempo que lleve conectado el ordenador (variable FRAMES).

Dentro del bucle comprendido entre las líneas 70 a 100 se realiza la asignación de valores aleatorios.

La generación del número aleatorio se realiza en la línea 80. Debido al algoritmo empleado, la función retorna valores comprendidos entre los números 1 y 100. El valor queda asignado a la variable «contenido».

En la línea 90 se realiza la asignación del valor de la variable «contenido», al elemento apuntado por la variable de control del bucle (n).

La visualización del contenido de la matriz lo lleva a cabo en el bucle formado por las líneas 120 a 140.

La línea 130 visualiza el n.º de subíndice. La 132 visualiza a continuación un espacio en blanco, si el número de subíndice es menor que 10; esto permite que la visualización de los números quede alineada.

La visualización de los contenidos de la matriz, seleccionados por la variable de con-

trol «n», lo realiza la línea 134. El formato de salida es a dos columnas.

b) Búsqueda:

En la línea 160 se realiza la entrada del valor a buscar.

La línea 170 comprueba si el número introducido es entero o no, y en la 180 si está comprendido dentro del rango 1 a 100.

En la 190 se inicializa la variable «veces» con el valor «0». En la línea 200 se dimensiona la matriz «b», ésta va a contener las direcciones de la matriz «a» correspondientes al valor buscado.

El bucle de exploración de los elementos de la matriz «a» está localizado entre las líneas 220 y 260.

En la línea 230 se comprueba si el elemento direccionado por la variable de control de bucle (n) es igual al contenido de la variable «valor».

Si son iguales en la línea 240 se incrementa la variable «veces», que contiene el número de ocasiones en que el valor es encontrado.

En la 250 se asigna al elemento de la matriz «b», seleccionado por la variable «veces», el valor de la variable de control del bucle (n), que indica la dirección de la tabla «a» donde el valor ha sido encontrado.

La línea 280 «limpia» la pantalla para poder visualizar los resultados de la búsqueda.

En la 282 se comprueba si el número ha sido encontrado. En caso negativo se visualiza un mensaje indicando el hecho.

Si al menos en una ocasión el valor se encuentra, las líneas 290 a 300 se encar-

```

150 REM
      *****
      *
      * BUSQUEDA *
      *
      *****

160 INPUT "Valor a encontrar >>
> ";valor
170 IF valor<>INT valor THEN GO
TO 160
180 IF valor<1 OR valor>100 THE
N GO TO 160
190 LET veces=0
200 DIM b(elementos)
210 REM
      *****
      *
      * EXPLORACION *
      *
      *****

220 FOR n=1 TO elementos
230 IF a(n)<>valor THEN GO TO 2
50
240 LET veces=veces+1
250 LET b(veces)=n
260 NEXT n
270 REM
      *****
      *
      * VISUALIZACION *
      *
      *****

280 CLS
282 IF veces=0 THEN PRINT "Ualo
r ";valor;" no encontrado": GO T
O 350
290 PRINT "El valor: ";valor
300 PRINT "ha sido encontrado:
";veces;" veces"
310 PRINT "en las posiciones:"

320 FOR n=1 TO veces
330 PRINT "> ";b(n)
340 NEXT n
350 PRINT #0;"Otro valor (S/N)"
360 PAUSE 0
370 LET z$=INKEY$
380 IF z$="S" OR z$="s" THEN GO
TO 150
390 IF z$="N" OR z$="n" THEN GO
TO 410
400 GO TO 360
410 CLS
412 FOR n=1 TO elementos
414 PRINT n;
416 IF n<10 THEN PRINT " ";
420 PRINT "> ";a(n);
430 NEXT n

```

gan de visualizar el «valor» y el número de «veces»; y el bucle formado por las líneas 320 a 340 de visualizar las posiciones.

Las líneas 350 a 400 comprueban si se desea buscar otro valor o no.

En caso negativo se «limpia» la pantalla y se vuelve a visualizar el contenido de la matriz de origen.

c) Par-Impar:

Las líneas 450 a 460 temporizan hasta que se pulsa una tecla y la 470 «limpia» la pantalla.

La 480 y 490 asignan a las variables «par» e «impar» el valor inicial 0.

La matriz «C» se dimensiona en la línea 500 y se utiliza para almacenar los números pares encontrados. La 510 dimensiona la tabla «D» que hace lo mismo con los números impares.

El bucle de búsqueda de números pares o impares está implementado entre las líneas 530 y 610.

La línea 540 asigna a la variable «división» el cociente entre el elemento de la matriz «a», direccionado por la variable de control (n), y el número dos.

La forma de calcular si el n.º es par o no se realiza en la línea 550. Un número es par cuando es divisible por dos, por lo tanto, si se multiplica la parte entera del cociente («división») por el divisor (2) y el resultado es igual al dividendo (a(n)), es que el resto es igual a cero, y por tanto, par.

Si el número es impar en las líneas 560 y 570 se incrementa la variable «impar», que almacena el número total de impares, y se asigna el número al elemento de la matriz «d», direccionada por el puntero

```

440 REM *****
      *                               *
      * PAR/IMPAR *
      *                               *
      *****
450 PRINT #0;"Pulsa una tecla p
ara continuar"
460 PAUSE 0
470 CLS
480 LET par=0
490 LET impar=0
500 DIM c(elementos)
510 DIM d(elementos)
520 REM *****
      *                               *
      * BUSQUEDA *
      *                               *
      *****
530 FOR n=1 TO elementos
540 LET division=a(n)/2
550 IF INT (division)*2=a(n) TH
EN GO TO 590
560 LET impar=impar+1
570 LET d(impar)=a(n)
580 GO TO 610
590 LET par=par+1
600 LET c(par)=a(n)
610 NEXT n
620 REM *****
      *                               *
      * VISUALIZACION *
      *                               *
      *****
625 CLS
630 PRINT "Numeros pares encont
rados: ";par
640 FOR n=1 TO par
650 PRINT "> ";c(n),
660 NEXT n
670 PRINT #0;"Pulsa una tecla p
ara continuar"
680 PAUSE 0
685 CLS
690 PRINT "Numeros impares enco
ntrados: ";impar
700 FOR n=1 TO impar
710 PRINT "> ";d(n),
720 NEXT n
730 PRINT #0;"Vuelvo a visualiz
arlos (S/N)"
740 PAUSE 0
750 LET z%=INKEY$
760 IF z%="S" OR z%="s" THEN GO
TO 620
770 IF z%="N" OR z%="n" THEN GO
TO 790
780 GO TO 740
790 CLS
800 FOR n=1 TO elementos
802 PRINT n;
804 IF n<10 THEN PRINT " ";
810 PRINT "> ";a(n),
820 NEXT n

```

«impar». En las líneas 590 y 600 se realiza la misma operación para los números pares.

La línea 625 «limpia» la pantalla. En la línea 630 se visualiza el número de pares encontrados. El bucle formado por los números 640 a 660 visualiza los valores de la matriz «c».

Las líneas 670 a 685 realizan la función de pausa y borrado.

La visualización del total de impares y sus valores, está implementado en las líneas 690 a 720.

Las líneas 730 a 780 comprueban si se desea volver a visualizar los números. En caso negativo, las líneas 790 a 820 nos vuelven a visualizar la tabla original.

d) Ordenación:

El último paso que queda es la ordenación de la tabla, para ello se ha utilizado el siguiente procedimiento.

Se compara el primer número de la tabla con el resto, si resulta que es el más pequeño, se queda donde está, pero si no, el valor más pequeño se sitúa en la primera posición y el primer número en su lugar. En el siguiente ciclo se realiza la misma operación con el número dos, este ciclo se repite tantas veces como elementos tenga la tabla menos uno, ya que el último valor queda automáticamente ordenado.

Las líneas 840 a 860 temporizan y posteriormente borran la pantalla cuando se pulsa una tecla.

El bucle formado por las líneas 870 a 980 es el dedicado a «barrer» todos los números menos uno.

La variable «menor» se inicializa en la línea 880 con el

```

830 REM
      *****
      * ORDENA *
      *****
840 PRINT #0;"Pulsa una tecla P
ara continuar"
850 PAUSE 0
860 CLS
870 FOR x=1 TO elementos-1
880 LET menor=x
890 FOR y=x+1 TO elementos
900 IF a(menor)>a(y) THEN LET m
enor=y
910 NEXT y
920 LET cambio=a(x)
930 LET a(x)=a(menor)
940 LET a(menor)=cambio
950 PRINT x;
960 IF x<10 THEN PRINT " ";
970 PRINT "> ";a(x),
980 NEXT x
990 PRINT elementos;
1000 IF elementos<10 THEN PRINT
" ";
1010 PRINT "> ";a(elementos)
1020 PAUSE 0

```

valor correspondiente de la variable de control del bucle (x).

El bucle de comparación se encuentra implementado en las líneas 890 a 910.

Al final del bucle la variable «menor» contiene el menor número de los comparados.

En las líneas 920 a 940 se realiza el intercambio entre los elementos direccionados

por los subíndices «x» y «menor»; este intercambio se apoya en la variable «cambio» para poder realizarlo.

La visualización de la posición y del valor ordenado lo ejecutan las líneas 950 a 970.

Para finalizar la visualización del último elemento (mayor), lo llevan a cabo las líneas 990 a 1010.

| | | | | | |
|----|---|----|----|---|----|
| 1 | > | 3 | 2 | > | 4 |
| 3 | > | 7 | 4 | > | 7 |
| 5 | > | 9 | 5 | > | 10 |
| 7 | > | 11 | 6 | > | 15 |
| 9 | > | 18 | 10 | > | 18 |
| 11 | > | 18 | 12 | > | 19 |
| 13 | > | 20 | 14 | > | 31 |
| 15 | > | 34 | 15 | > | 34 |
| 17 | > | 36 | 18 | > | 41 |
| 19 | > | 42 | 20 | > | 49 |
| 21 | > | 50 | 22 | > | 53 |
| 23 | > | 54 | 24 | > | 55 |
| 25 | > | 56 | 26 | > | 62 |
| 27 | > | 64 | 28 | > | 71 |
| 29 | > | 79 | 30 | > | 71 |
| 31 | > | 79 | 32 | > | 77 |
| 33 | > | 81 | 34 | > | 85 |
| 35 | > | 85 | 36 | > | 87 |
| 37 | > | 88 | 38 | > | 90 |
| 39 | > | 92 | 40 | > | 90 |

Programa 1. Ordenación.

PROGRAMA 2

```

10 REM *****
* CURSO/BASIC *
*****
* CARTAS *
*****

15 BORDER 1: PAPER 4: INK 0: C
Ls
20 PRINT AT 3,0;"(C) MICRO
HOBBY SEMANAL"
22 PRINT AT 6,13;"JUEGO"
24 PRINT AT 12,7;"LAS SIETE y
MEDIA"
26 PRINT AT 18,0;"Por Rafa
el Prades"
28 PAUSE 150
30 REM *****
* CREDITO/APUESTA *
*****

40 LET credito=10000
50 LET cred ordenador=credito
60 LET cred Jugador=credito
70 LET apuesta=100
75 LET Manos=1
80 REM *****
* DIMENSIONADO *
*****

90 DIM m(40,2)
100 DIM b(10,4)
110 DIM c(40,2)
115 DIM v(10)
120 DIM t$(10,4,20)
125 DIM p$(1,6)
130 REM *****
* ASIGNACION *
*****

140 FOR p=1 TO 4
142 RESTORE 250+p
144 READ a$
150 RESTORE 260
160 FOR n=1 TO 10
170 READ b$
180 LET t$(n,p)=b$+" de "+a$
190 NEXT n
200 NEXT p
250 REM *****
* DATOS DE CARTAS *
*****

251 DATA "OROS"
252 DATA "COPAS"
253 DATA "ESPADAS"
254 DATA "BASTOS"
260 DATA "AS", "DOS", "TRES", "CUA
TRO", "CINCO", "SEIS", "SIETE", "SOT
A", "CABALLO", "REY"
300 REM *****
* VALOR *
*****

310 FOR x=1 TO 7
320 LET v(x)=x
330 NEXT x
340 FOR x=8 TO 10
350 LET v(x)=0.5
360 NEXT x

```

```

365 GO SUB 2100
370 REM *****
* BARAJEAR *
*****

372 RANDOMIZE
380 CLS
390 PRINT AT 3,0;" ESPERA"
400 PRINT AT 18,0;" VOY A BA
RAJEAR"
410 FOR x=1 TO 40
420 GO SUB 490
430 IF b(numero,palo)=1 THEN GO
TO 420
440 LET m(x,1)=numero
450 LET m(x,2)=palo
460 LET b(numero,palo)=1
470 NEXT x

480 GO TO 530
490 REM *****
* CARTA ALEATORIA *
*****

500 LET numero=INT (RND*10)+1
510 LET palo=INT (RND*4)+1
520 RETURN
530 REM *****
* CORTAR EL MAZO *
*****

540 CLS
550 PRINT AT 3,0;" !!! YA ES
TA !!!"
560 PRINT AT 20,1;"Por que nume
ro quieres cortar?"
570 INPUT "> "; LINE n$
572 IF n$="" THEN GO TO 570
574 FOR x=1 TO LEN n$
576 IF n$(x)("<0" OR n$(x)(">9" T
HEN GO TO 570
578 NEXT x
580 LET numero=VAL n$
590 IF numero<1 OR numero>40 TH
EN GO TO 570
592 CLS
594 PRINT AT 3,0;" ESPERA"
600 LET resto=40-numero
610 FOR x=1 TO numero
620 LET c(x+resto,1)=m(x,1)
630 LET c(x+resto,2)=m(x,2)
640 NEXT x
650 FOR x=numero+1 TO 40
660 LET c(x-numero,1)=m(x,1)
670 LET c(x-numero,2)=m(x,2)
675 NEXT x
680 PRINT AT 3,0;" PODEMOS E
MPEZAR A JUGAR"
690 PRINT AT 18,0;" !!! BUEN
A SUERTE !!!"
700 PRINT #0;" Pulsa una tecla
para continuar"
710 PAUSE 0
715 CLS
720 REM *****
* INICIAL JUGADOR *
*****

722 PRINT AT 0,0;" TU JUEGAS"

```

```

730 LET puntos jugador=0
740 LET carta=1
750 PRINT AT 3,0;"> ";
760 GO SUB 800
770 LET puntos jugador=valor ca
780 LET carta=2
790 GO TO 860
800 REM
*****
* PRESENTACION *
*****

810 LET numero=c(carta,1)
820 LET palo=c(carta,2)
830 PRINT t$(numero,palo)
840 LET valor carta=v(numero)
850 RETURN
860 REM
*****
* MAS JUGADAS? *
*****

870 PRINT #0;AT 1,3;"Quieres ot
ra carta (S/N)"
880 IF INKEY$="" THEN GO TO 880
890 LET z$=INKEY$
900 IF z$="S" OR z$="s" THEN GO
TO 930
910 IF z$="N" OR z$="n" THEN LE
T indice=carta: GO TO 1140
920 GO TO 860
930 INPUT 0
932 FOR x=1 TO 25: NEXT x
934 LET carta=carta+1
940 PRINT "> ";
950 GO SUB 800
960 LET puntos jugador=puntos j
ugador+valor carta
970 IF puntos jugador>7.5 THEN
GO TO 1000
980 IF puntos jugador=7.5 THEN
GO TO 1070
990 GO TO 870
1000 PRINT AT 0,0;" LO SIENTO,
ME HAS PASADO"
1010 LET p$(1)=STR$ puntos jugad
or
1020 PRINT AT 21,0; INVERSE 1;"
HAS HECHO UN TOTAL DE ";p$(1)
1030 LET ganador=0
1035 LET puntos ordenador=0
1040 PRINT #0;" Pulsa una tecla
para continuar"
1050 PAUSE 0
1060 GO TO 1630
1070 REM
*****
* 7 1/2 JUGADOR *
*****

1080 PRINT AT 0,0;" ARRIBA SIET
E Y MEDIA"
1090 PRINT AT 21,0;" ESPERA A C
ONOCER MIS TANTOS"
1100 LET indice=carta
1110 PRINT #0;" Pulsa una tecla
para continuar"
1120 PAUSE 0
1130 GO TO 1200
1140 REM
*****
* TANTOS JUGADOR *
*****

1150 LET p$(1)=STR$ puntos jugad
or
1160 PRINT AT 0,0; INVERSE 1;"
HAS HECHO UN TOTAL DE ";p$(1)
1170 PRINT AT 21,0;" AHORA JUEG
O YO"

```

```

1175 INPUT w
1180 PRINT #0;" Pulsa una tecla
para continuar"
1190 PAUSE 0
1200 REM
*****
* INICIAL ORDENADOR *
*****

1210 CLS
1220 LET puntos ordenador=0
1230 LET carta=2
1240 PRINT AT 0,0;" YO JUEGO"
1250 PRINT AT 3,0;"> ";
1252 LET y=5
1260 GO SUB 800
1270 LET puntos ordenador=valor
carta
1280 LET carta=indice
1290 REM
*****
* MAS JUGADAS? *
*****

1300 IF puntos ordenador>7.5 THE
N GO TO 1330
1310 IF puntos ordenador=7.5 THE
N GO TO 1400
1320 IF puntos ordenador<7.5 THE
N GO TO 1450
1330 LET ganador=1
1340 PRINT AT 0,0;" ME HE PASADO
O"
1350 LET p$(1)=STR$ puntos orden
ador
1360 PRINT AT 21,0; INVERSE 1;"
HE HECHO UN TOTAL DE ";p$(1)
1370 PRINT #0;" Pulsa una tecla
para continuar"
1380 PAUSE 0
1390 GO TO 1630
1400 REM
*****
* 7 1/2 ORDENADOR *
*****

1410 LET ganador=0
1420 PRINT AT 0,0;" ARRIBA SIET
E Y MEDIA"
1430 PRINT #0;" Pulsa una tecla
para continuar"
1432 PAUSE 0
1440 GO TO 1630
1450 REM
*****
* CONTINUA *
*****

1460 IF puntos jugador=7.5 THEN
GO TO 1490
1470 IF puntos ordenador<6 THEN
GO TO 1490
1480 GO TO 1560
1490 PRINT AT 21,0;" PIDO OTRA
CARTA"
1492 PAUSE 50
1494 PRINT AT 21,0;"

1500 FOR x=1 TO 50: NEXT x
1510 LET carta=carta+1
1520 PRINT AT y,0;"> ";
1522 LET y=y+2
1530 GO SUB 800
1540 LET puntos ordenador = punt

```

```

os ordenador+valor carta
1550 GO TO 1300
1560 REM

*****
* TANTOS ORDENADOR *
*
*****

1570 PRINT AT 0,0,"ME QUEDO"
1580 LET p$(1)=STR$ puntos orden
ador
1590 PRINT AT 21,0, INVERSE 1;"
HE HECHO UN TOTAL DE ",p$(1)
1592 PRINT #0;" Pulsa una tecla
para continuar"
1594 PAUSE 0
1600 REM

```

```

*****
* COMPARACION *
*
*****

```

```

1610 IF puntos ordenador>puntos
jugador THEN LET ganador=0
1620 IF puntos ordenador<puntos
jugador THEN LET ganador=1
1630 REM

```

```

*****
* GANADOR *
*
*****

```

```

1635 CLS
1640 IF ganador=0 THEN PRINT AT
3,0," !!! HE GANADO !!!"
1645 AT 18,0," LO SIENTO, O
TRA VEZ SERA"
1650 IF ganador=1 THEN PRINT AT
3,0," !!! HAS GANADO !!!"
1655 AT 18,0," LA PROXIMA G
ANARE YO"
1660 PRINT AT 8,4;"Puntos Jugado
r...";puntos jugador
1670 PRINT AT 13,4;"Puntos Orden
ador..."
1680 IF puntos ordenador=0 THEN
PRINT "?": GO TO 1700
1690 PRINT puntos ordenador
1700 PRINT #0;" Pulse una tecla
para continuar"
1710 PAUSE 0
1720 REM

```

```

*****
* CALCULOS *
*
*****

```

```

1730 IF ganador=0 THEN GO TO 179
0
1740 IF puntos jugador=7.5 THEN
LET doble=2: GO TO 1760
1750 LET doble=1
1760 LET cred ordenador=cred ord
enador-apuesta*doble
1770 LET cred jugador=cred jugad
or+apuesta*doble
1780 GO TO 1810
1790 LET cred ordenador=cred ord
enador+apuesta
1800 LET cred jugador=cred jugad
or-apuesta
1810 REM

```

```

*****
* RESULTADOS *
*
*****

```

```

1815 CLS
1820 PRINT AT 7,3;"Manos jugadas
...";Manos
1830 PRINT AT 12,3;"Credito Juga
dor...";cred jugador
1840 PRINT AT 17,3;"Credito Orde
nador...";cred ordenador
1850 IF cred ordenador=0 THEN PR
INT AT 3,0;" HAS ROTO LA BANCA
"; PAUSE 0: STOP
1860 IF cred jugador=0 THEN PRIN
T AT 3,0;" SE ACABO TU DINERO
"; PAUSE 0: STOP
1870 PRINT AT 3,0;" CREDITOS"
1875 REM

```

```

*****
* CONTINUAR? *
*
*****

```

```

1880 PRINT #0;" Quieres jugar ot
ra mano (S/N)"
1890 PAUSE 0
1900 LET z$=INKEY$
1910 IF z$="S" OR z$="s" THEN GO
TO 1940
1920 IF z$="N" OR z$="n" THEN IN
PUT 0: PAUSE 0: STOP
1930 GO TO 1890
1940 INPUT 0
1945 LET manos=manos+1
1950 FOR a=1 TO 10
1955 FOR b=1 TO 4
1970 LET b(a,b)=0

```

```

1980 NEXT b
1990 NEXT a
2000 GO TO 380
2100 REM

```

```

*****
* INSTRUCCIONES *
*
*****

```

```

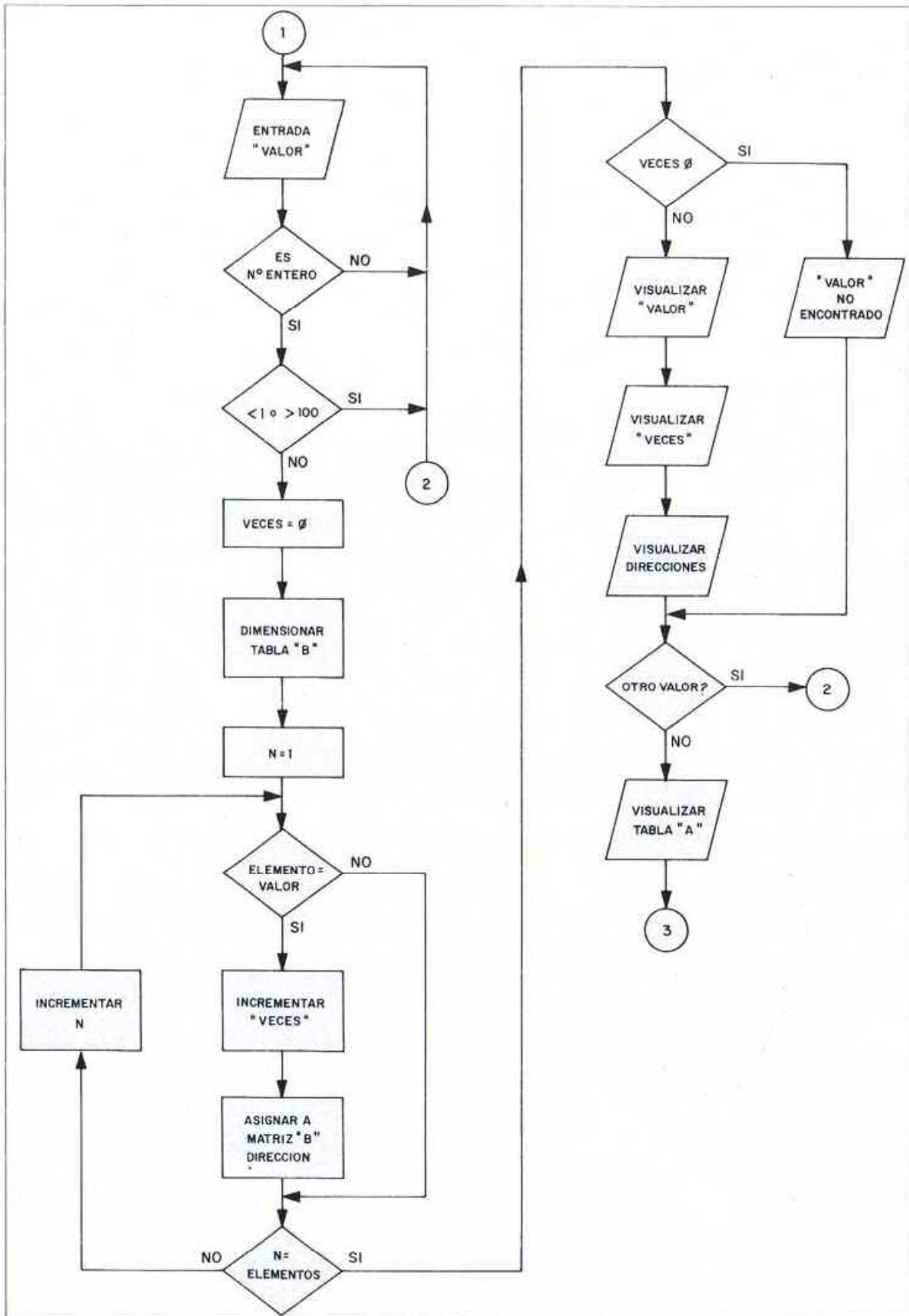
2110 PRINT #0,AT 1,6;"Instruccio
nes (S/N)"
2120 PAUSE 0
2130 LET z$=INKEY$
2140 IF z$="S" OR z$="s" THEN GO
TO 2165
2150 IF z$="N" OR z$="n" THEN RE
TURN
2160 GO TO 2120
2165 CLS
2170 PRINT AT 0,0;" INST
RUCCIONES"

```

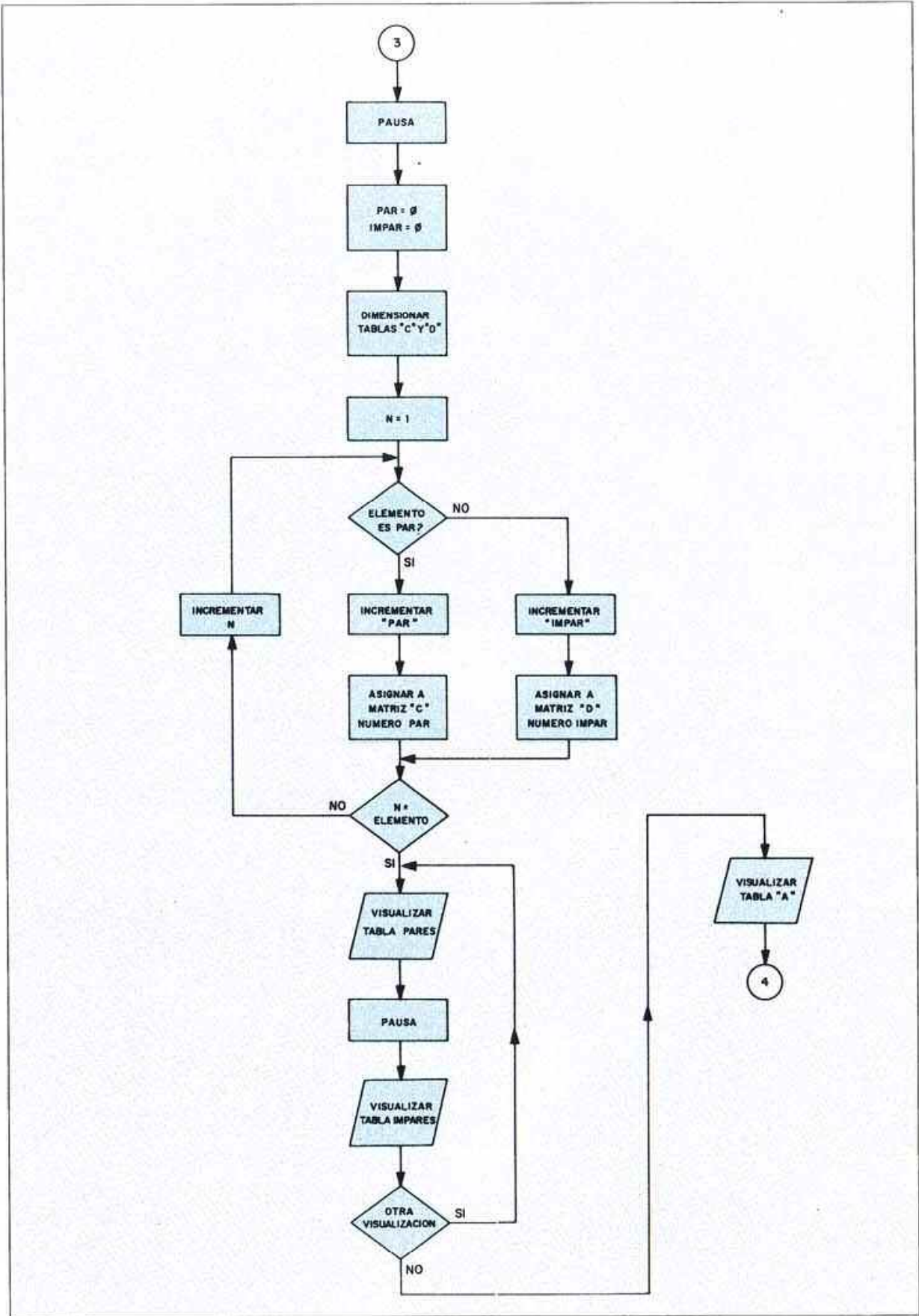
```

2180 PRINT " Este programa est
a basado en" " el conocido jueg
o de naipes:" " "Las
7 1/2"
2190 PRINT AT 10,1;"Tienes un cr
edito inicial de:";AT 12,9;"10.0
00 pesetas"
2200 PRINT AT 15,1;"Las apuestas
son de 100 ptas."
2210 PRINT AT 18,1;"Si tienes 7
1/2 y ganas, obtie" " un benefi
cio de 200 ptas."
2220 PRINT #0;" Pulsa una tecla
para continuar"
2230 PAUSE 0
2240 RETURN

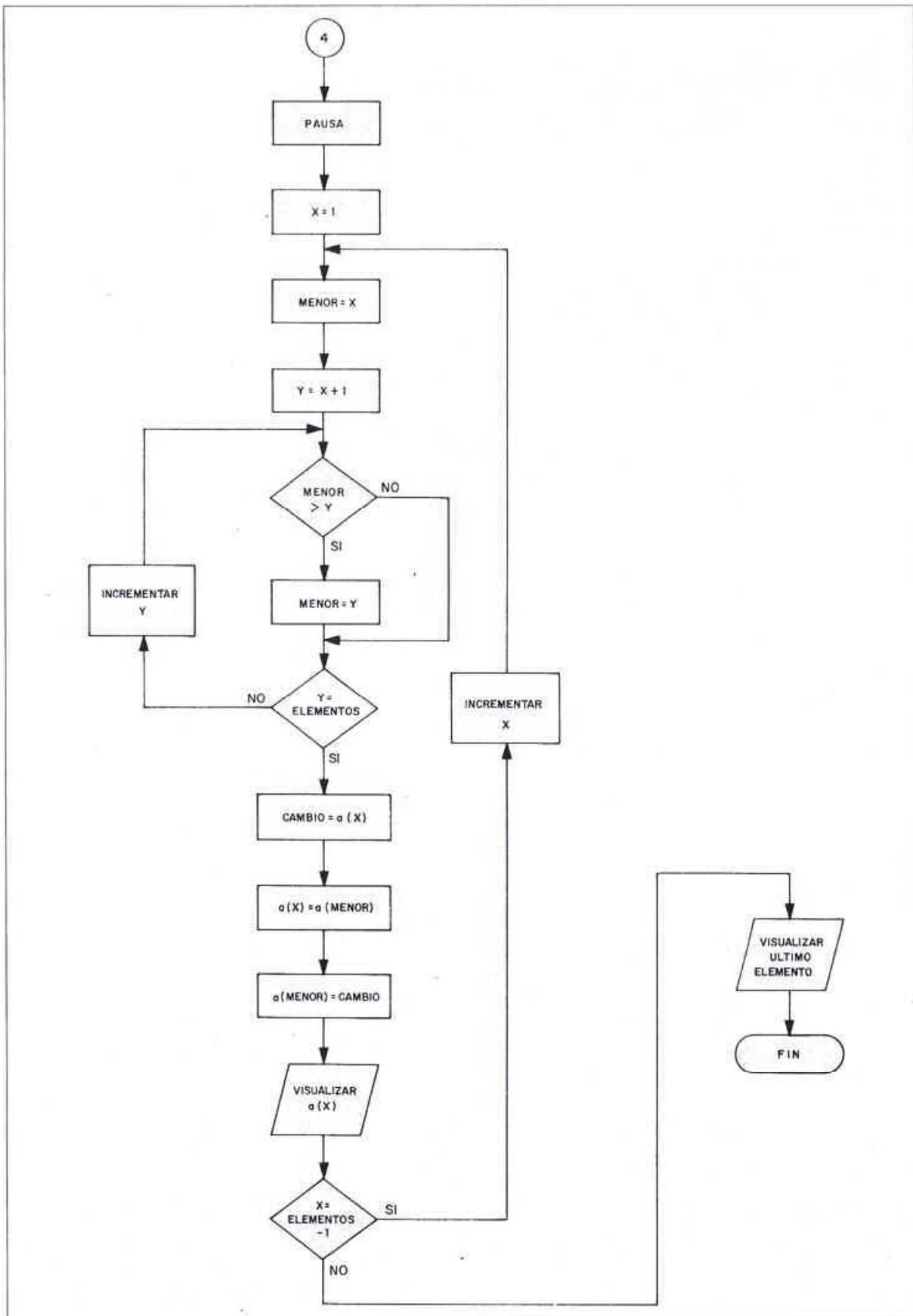
```

Manejo Tablas. Búsqueda.



Manejo tablas. PAR/IMPAR.



Manejo tablas. Ordenación.

Matrices de cadena

La estructura del dimensionado de matrices de cadena es el siguiente:

| SENTENCIA | ARGUMENTO |
|-----------|-----------------------------|
| DIM | letra \$ (lista de valores) |

Ejemplos:

- DIM a\$ (30)
- DIM x\$ (2, 20)
- DIM n\$ (3, 7, 5)
- DIM z\$ (8, 4, 5, 10)

El nombre de una matriz de cadena está formado por una sola letra seguida del símbolo «dolar» (\$). A diferencia de las numéricas, no pueden compartir el mismo nombre una matriz de cadena y una variable del mismo tipo.

Al dimensionar una matriz de cadena, el contenido de sus elementos queda inicializado con espacios en blanco.

Ejemplo:

```
10 DIM a$ (10, 7)
20 FOR n = 1 TO 10
30 PRINT INVERSE 1, a$ (n)
40 NEXT n
```

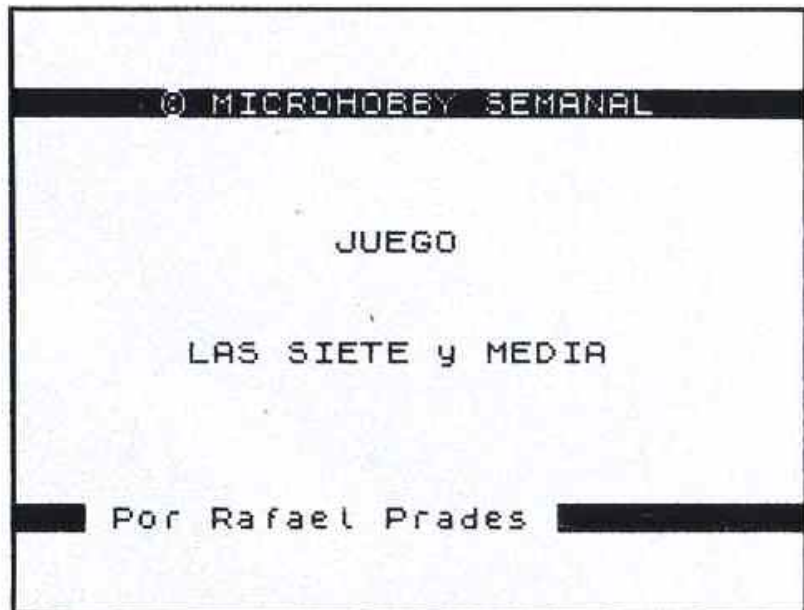
El argumento «INVERSE 1» permite visualizar los espacios, ya que intercambia el color de «tinta» por el de «papel».

Existen ciertas diferencias entre el dimensionado de matrices numéricas y el de cadenas, debido a que es preciso indicar de cuantos caracteres va a constar cada elemento.

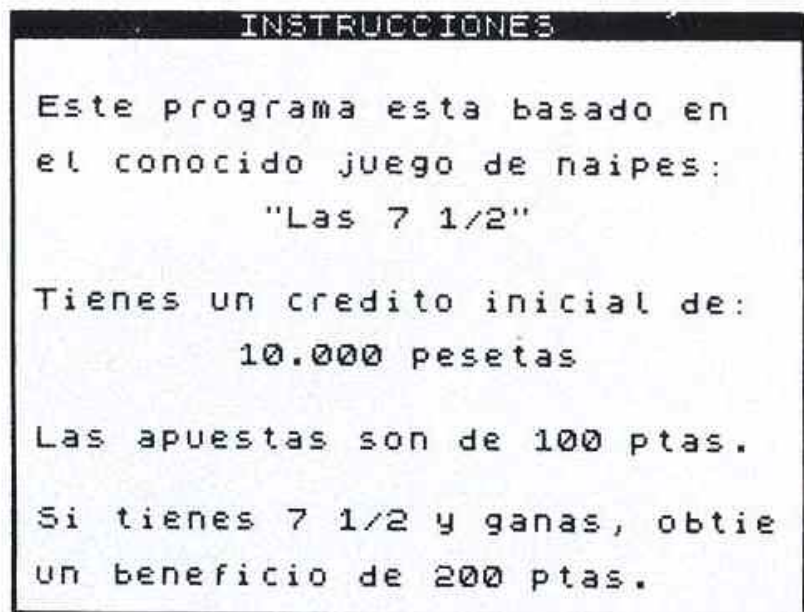
Ejemplos:

a) Dimensionar una matriz de un elemento de 20 caracteres.

```
DIM n$ (20)
```



Programa 2. Carátula.



Programa 2. Ilustraciones.

Las características de esta matriz son similares a las de una variable de cadena.

b) Matriz «unidimensional» de 100 por 20 elementos, con 7 caracteres cada uno.

```
DIM Z$ (7, 15)
```

c) Matriz «bidimensional»

de 100 por 20 elementos, con 7 caracteres cada uno.

```
DIM J$ (100, 20, 7)
```

d) Matriz «tridimensional» formada por tres planos de 10 por 4; el máximo de caracteres por elemento es cinco.

```
DIM T$ (3, 10, 4, 5)
```

```

LO SIENTO, TE HAS PASADO

> AS de BASTOS
> SOTA de ESPADAS
> TRES de COPAS
> CABALLO de BASTOS
> CUATRO de BASTOS

HAS HECHO UN TOTAL DE 9

```

Programa 2. Juegos.

Asignación

En la asignación de valores a los elementos de una matriz de cadena, no es necesario hacer referencia al último subíndice dimensionado, es decir, el que indica el número de caracteres.

Ejemplo:

```

10 DIM W$(10, 5)
20 LET W$(3) = "PEPE"
30 PRINT W$(3)

```

Cuando se indica dicho subíndice, es que se hace referencia a un determinado carácter del elemento seleccionado. Siguiendo con el mismo ejemplo:

```

40 LET W$(3, 4) = "A"
50 PRINT W$(3, 4)
60 PRINT W$(3)

```

La asignación de la línea 40 quiere expresar: Asignar al carácter cuatro del elemento tres, perteneciente a la matriz «W\$», el valor de cadena «A»; los demás caracteres

quedan con el mismo valor.

La asignación de valores es del tipo *procusteano*, es decir, que cuando la longitud de la cadena es menor que el número de caracteres reservados, el elemento de la matriz se rellena con espacios y por el contrario, cuando es más larga se recorta. En la página 45 (Asignación de subcadenas) viene explicado este tipo de asignación.

Ejemplos:

a) Rellenado con espacios.

```

10 REM *****
      RELENADO *****
20 DIM M$(12, 10)
30 RESTORE
40 FOR n=1 TO 12
50 READ JS
60 LET M$(n) = JS
70 PRINT INVERSE 1; JS, INVERSE
80
90 PRINT INVERSE 1; M$(n)
80 NEXT n
90 REM *****
      DATOS *****
100 DATA "ENERO", "FEBRERO", "MARZO", "ABRIL", "MAYO", "JUNIO", "JULIO", "AGOSTO", "SEPTIEMBRE", "OCTUBRE", "NOVIEMBRE", "DICIEMBRE"

```

Los elementos de la matriz «M\$» están dimensionados de 10 caracteres, todos los

datos (meses) que tengan *menor* longitud quedan rellenos con espacios, de esta manera se mantiene la longitud total (10).

b) Recortado.

```

10 REM *****
      RECORTADO *****
20 DIM d$(7, 5)
30 RESTORE
40 FOR n=1 TO 7
50 READ a$
60 LET d$(n) = a$
70 PRINT INVERSE 1; a$, INVERSE
80
90 PRINT INVERSE 1; d$(n)
80 NEXT n
90 REM *****
      DATOS *****
100 DATA "LUNES", "MARTES", "MIÉRCOLES", "JUEVES", "VIERNES", "SÁBADO", "DOMINGO"

```

En este otro ejemplo, los elementos de la matriz «d\$» tienen una longitud fija de cinco caracteres, cualquier dato (días) que tenga una *mayor* longitud queda recortado a este valor.

Fragmentación

De la misma manera que las variables de cadena, las matrices pueden fragmentarse (VER pag. 43).

Asignemos primero un valor a un elemento de la matriz «A\$», y veamos posteriormente unos ejemplos:

```

10 DIM A$(10, 10)
20 LET A$(5) = "ORDENADOR"
30 PRINT A$(5)

```

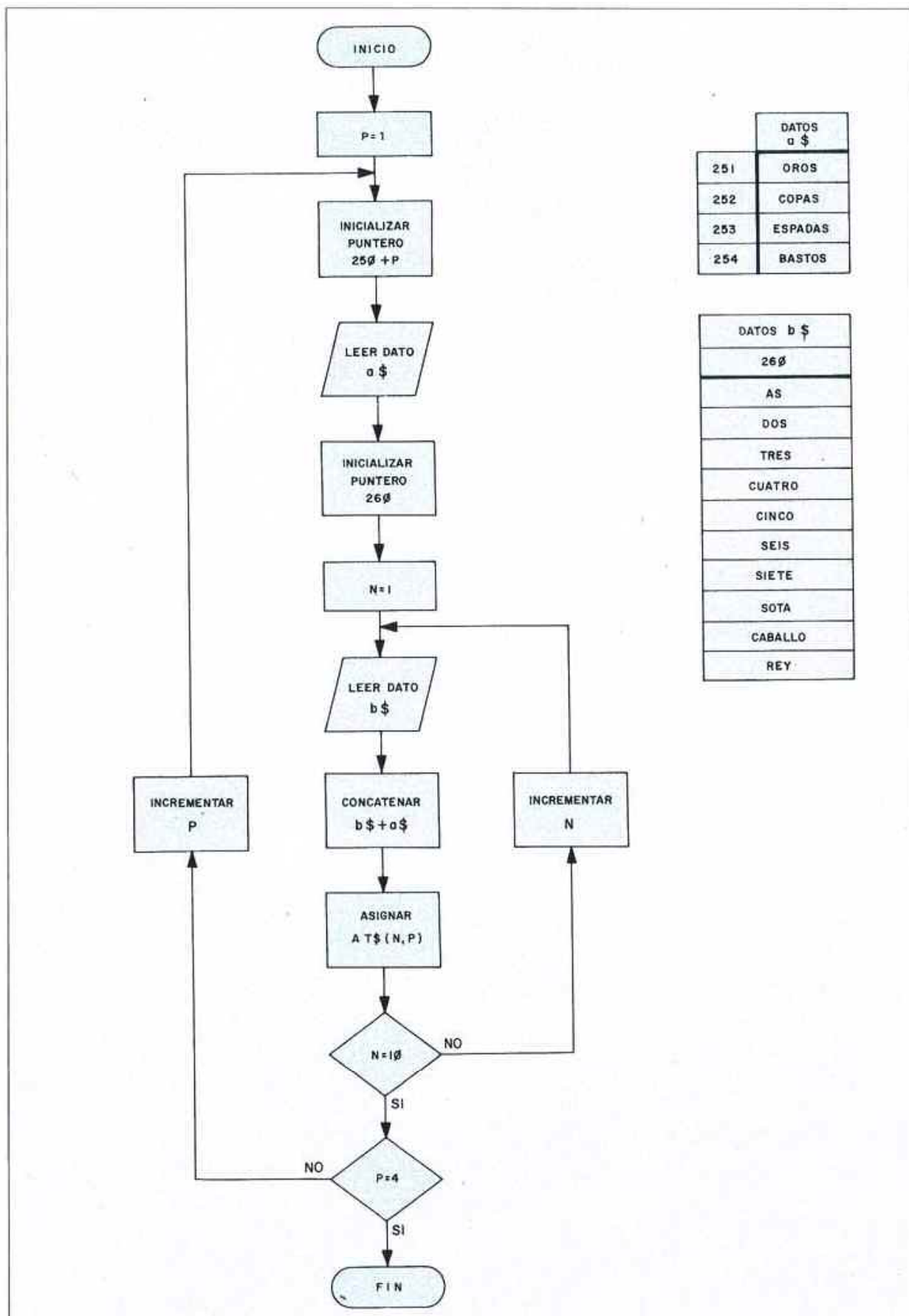
a) Asignar a los caracteres 3 a 5 el valor «POS».

```

40 LET A$(5, 3 TO 5) = "POS"
5 PRINT A$(5)

```

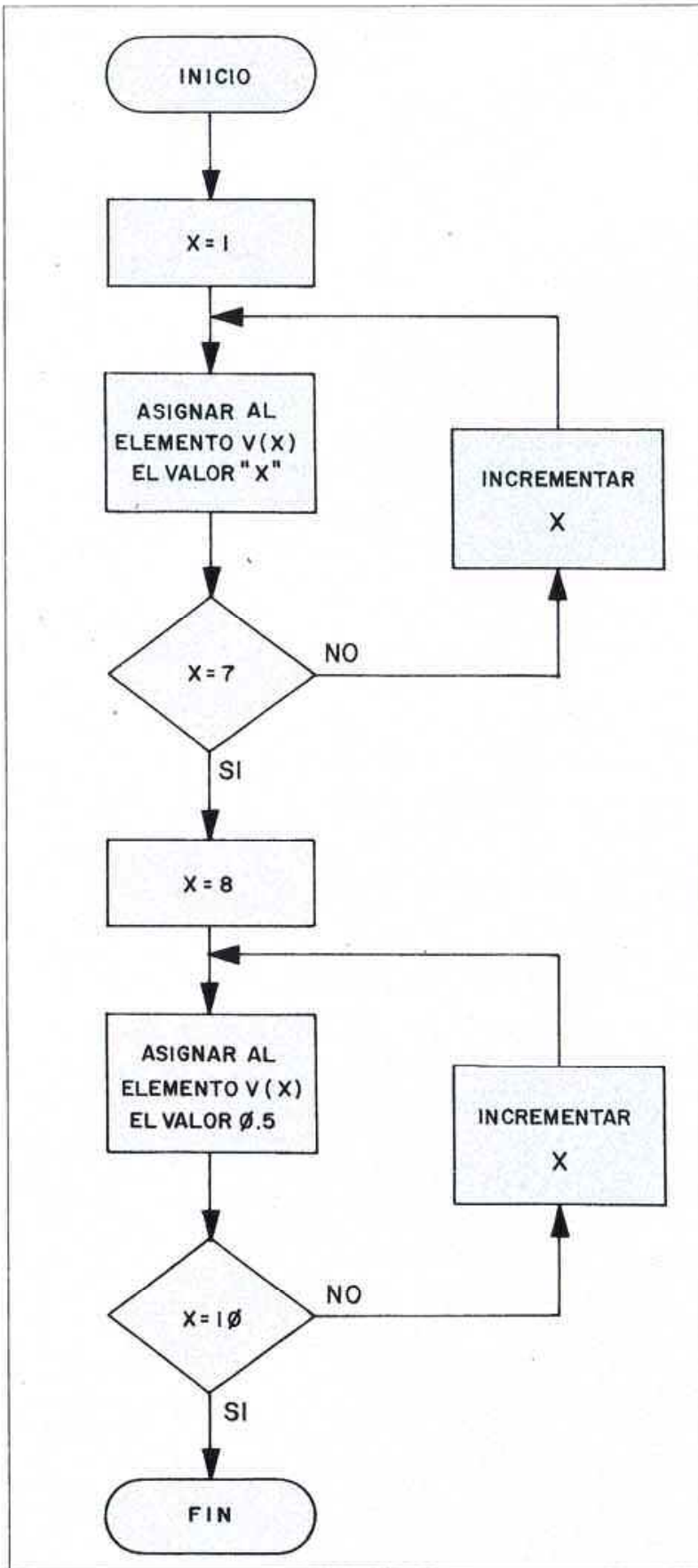
b) Asignar a los cuatro primeros caracteres la cadena «TRIN».



| DATOS a \$ | |
|------------|---------|
| 251 | OROS |
| 252 | COPAS |
| 253 | ESPADAS |
| 254 | BASTOS |

| DATOS b \$ | |
|------------|---------|
| 260 | AS |
| | DOS |
| | TRES |
| | CUATRO |
| | CINCO |
| | SEIS |
| | SIETE |
| | SOTA |
| | CABALLO |
| | REY |

Rutina asignación de naipes.



Rutina asignación de naipes.

```

60 LET A$(5, TO 4) = "TRIN"
70 PRINT A$(5)
  
```

c) Asignar a partir del quinto carácter la cadena "CHAR".

```

80 LET A$(5, 5 TO) = "CHAR"
90 PRINT A$(5)
  
```

d) Asignar al carácter número 9 el valor "E".

```

100 LET A$(5, 9) = "E"
110 PRINT A$(5)
  
```

La fragmentación permite utilizar tablas bidimensionales con diversos campos. Supongamos que deseamos crear una tabla para almacenar los datos referentes a nuestros programas, ésta podría tener cuatro campos:

- PROGRAMA
- CINTA
- CARA
- TIPO

En el primer campo almacenaríamos el nombre del programa; en el segundo, el nombre de la cinta donde se encuentra; en el tercero en qué cara de la cinta, y en el último, el tipo de programa, es decir, si es un juego, utilidad, etc.

La cantidad de caracteres de cada campo va a ser la siguiente:

| CAMPO | CARACT. |
|----------|---------|
| PROGRAMA | 10 |
| CINTA | 10 |
| CARA | 1 |
| TIPO | 1 |

El campo «CARA» es de un solo carácter ya que va a con-

2 Variable not found

b) Cuando uno de los subíndices es negativo o mayor que «65535» aparece:

8 Integer out of range

Ejemplo:

- PRINT a\$(12,-1)
- PRINT a\$(67000,3)

c) Cuando un subíndice está fuera de los límites del dimensionado de una matriz, se visualiza

3 Subscript wrong

Por ejemplo, en la matriz «b\$(20,10)»

- LET b\$(0,5)
- PRINT b\$(7,11)

d) Cuando se dimensiona una tabla demasiado grande que ocupa toda la memoria destinada a los programas «BASIC», aparece el error

4 Out of memory

Grabación de datos

Con el comando «SAVE» es posible almacenar o grabar en cinta las tablas generadas en alguno de nuestros programas; también pueden ser cargadas, posteriormente en el ordenador, con el comando «LOAD».

La sintaxis es la siguiente:

SAVE "nombre" DATA letra ()

en el caso de matrices numéricas, y

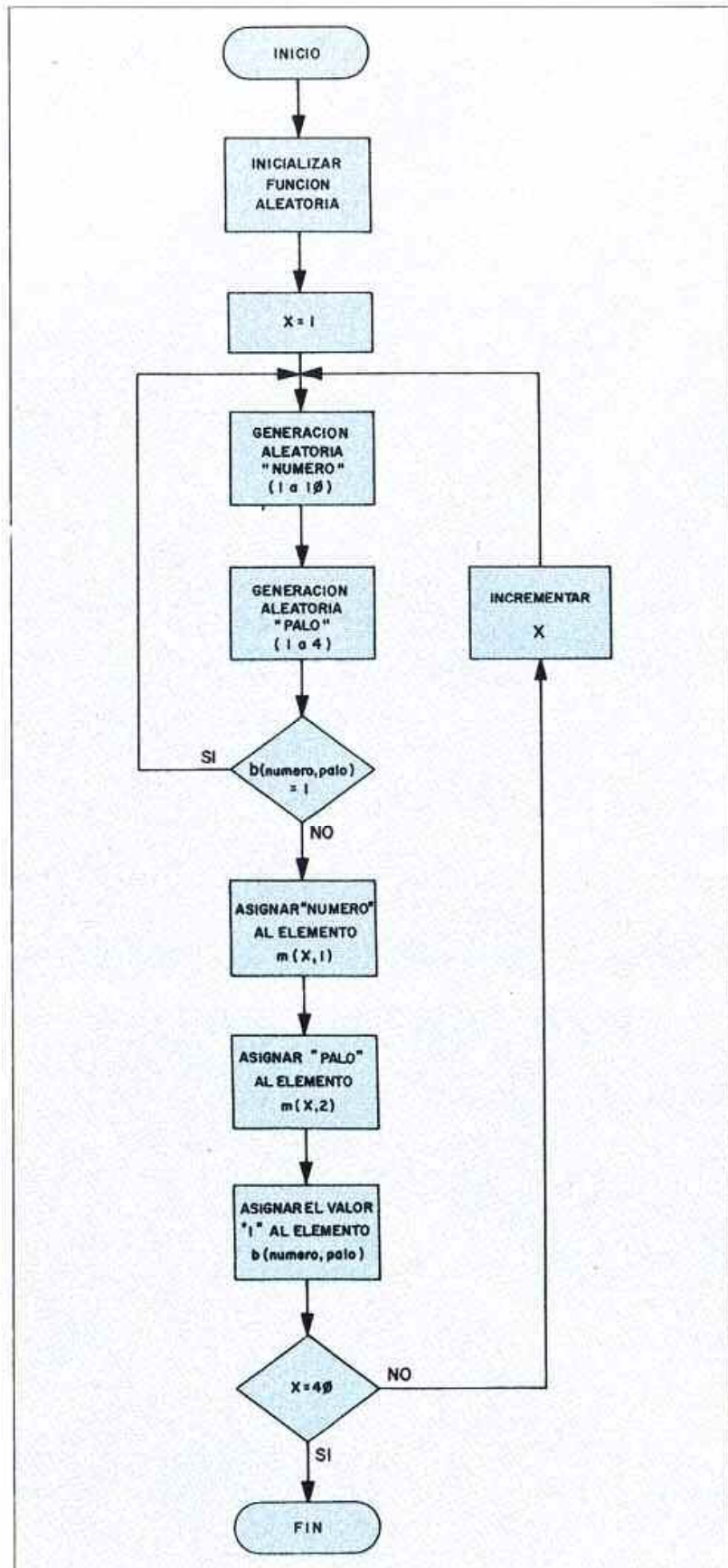
SAVE "nombre" DATA letra \$ ()

en el de cadena.

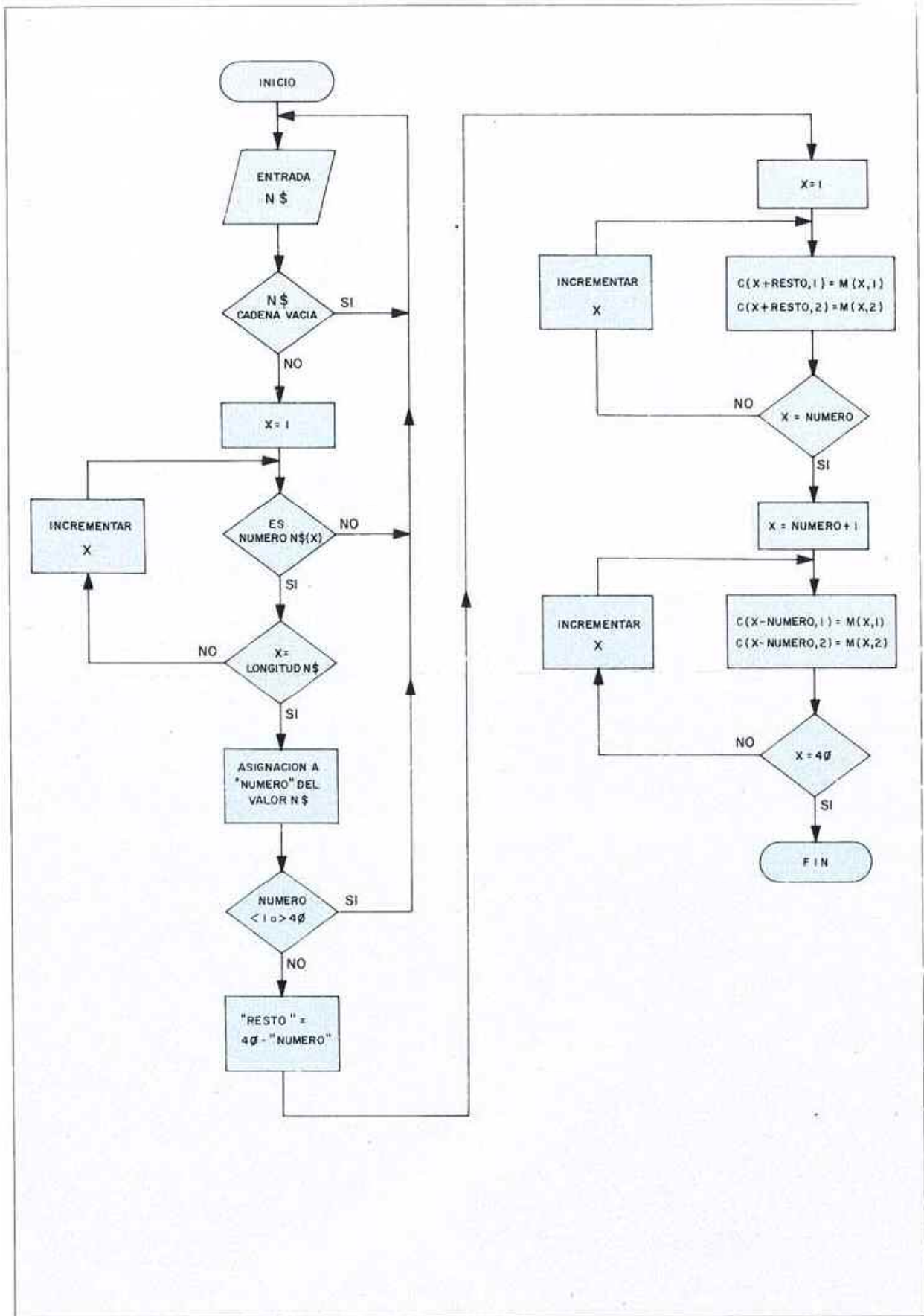
El *nombre*, entre comillas, es el que se asigna a la tabla de datos en la cinta, y la *letra* corresponde al asignado al dimensionar la matriz.

Si desea verificar o cargar una tabla, previamente salva-da, sustituya el comando «SAVE» por «VERIFY» o «LOAD».

MICROBASIC 200



Rutina «barajar» los naipes.



Rutina «cortar» el mazo.

Programa

El programa número «2» permite jugar con el ordenador, al conocido juego de cartas: «Las Siete y Media».

El ordenador hace el papel de «banca» y por tanto, baraja y reparte las cartas; nosotros tenemos la posibilidad de indicar por qué número deseamos «cortar» la baraja.

Después de repartir las dos primeras cartas, una para él y otra para nosotros, nos irá preguntando sucesivamente si deseamos otra carta o no.

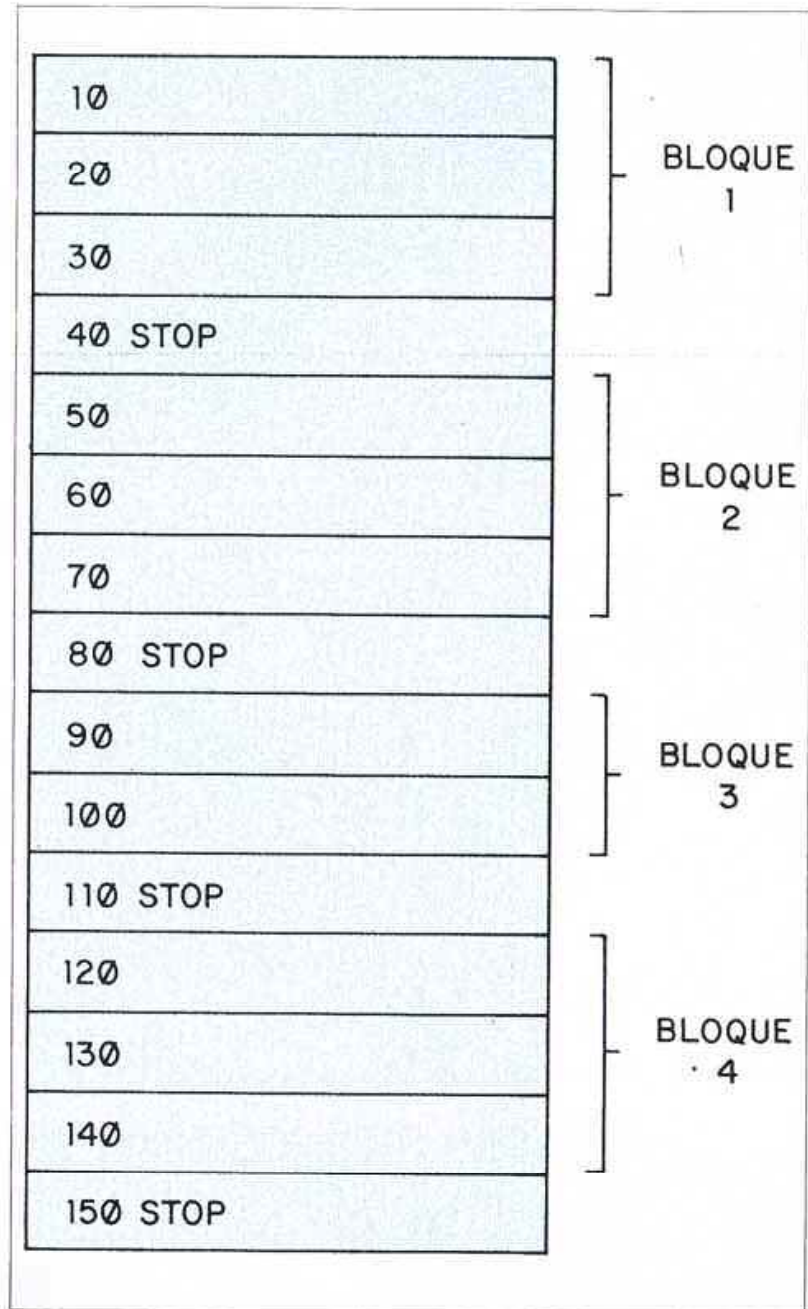
Al final de la partida se presentan los tantos obtenidos por cada uno de los dos jugadores y si deseamos jugar otra «mano».

Para dar mayor emoción a la partida, al principio de ésta, tenemos un crédito de «10.000» ptas. Las apuestas son de «100» ptas., ganamos el doble si obtenemos «siete y media» y el ordenador no, ya que en igualdad de puntos gana la banca.

¡BUENA SUERTE!

Hay una serie de matrices utilizadas en la confección del programa, que se encuentran dimensionadas en las líneas 90 a 125.

La matriz «T\$» almacena el nombre de cada carta (as de oros, dos de ..., etc.). Es una tabla bidimensional de «10» por «4», ya que la baraja consta de 4 «palos» de diez cartas. La tercera dimensión (20) indica el número máximo de caracteres de cada elemento, y se encuentra ligeramente sobredimensionada ya que con dieciocho hubiera bastado, por que el contenido de mayor longitud va a ser «caballo de espadas» (16 letras + 2 espacios).



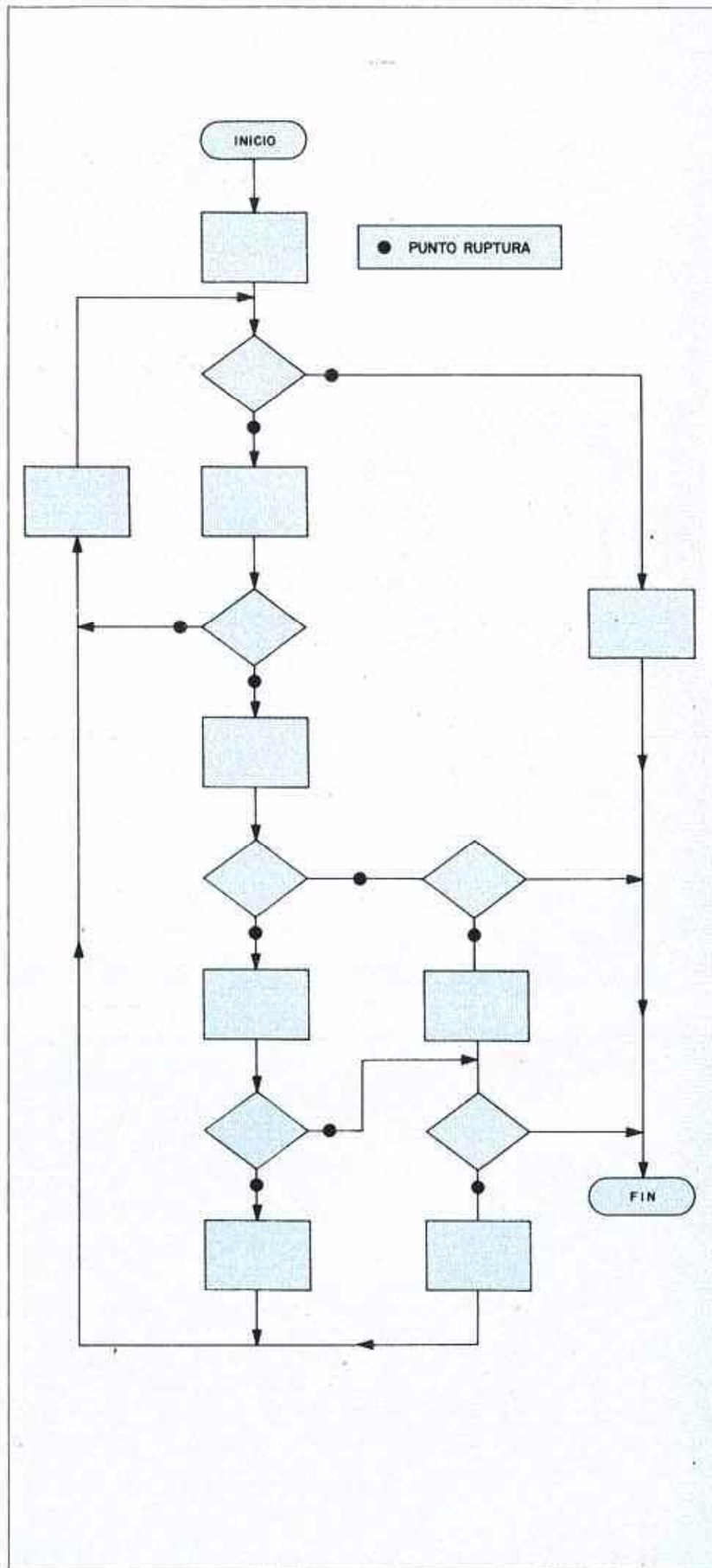
«Depurado por partes».

Los bucles anidados «p» y «n» (líneas 140-200 y 160-190) son los encargados de asignar a la matriz «T\$» los valores de cadena especificados en las sentencias «DATA» (líneas 251 a 260).

En los juegos de naipes, cada carta tiene un valor relativo distinto; en el caso de las siete y media, las cartas del «as» al «siete» tienen un valor

idéntico al que figura en una de sus esquinas. Por el contrario, las «figuras» (sota, caballo y rey) tienen un valor de medio punto. Con estos requisitos, los bucles 310-330 y 340-360 asignan a los elementos de la matriz «V» el valor correspondiente.

«V» es una matriz unidimensional de diez elementos, uno por cada valor de carta. La re-



Puntos estratégicos de ruptura.

lación entre el subíndice y la carta es el siguiente:

- 1 - AS
- 2 - DOS
- 3 - TRES
- 4 - CUATRO
- 5 - CINCO
- 6 - SEIS
- 7 - SIETE
- 8 - OCHO
- 9 - CABALLO
- 10 - REY

Para barajar las cartas se utiliza la función pseudoaleatoria «RND». La subrutina localizada en la línea 490, retorna dos valores aleatorios asignados a las variables «número» y «palo». La variable «número» tiene un valor comprendido entre «1» y «10»; la correspondencia, con los números de carta, es idéntica a la utilizada por la matriz «V». La variable «palo» puede tener valores entre «1» y «4»; la relación entre dicho valor y el palo de la carta es:

- 1 - OROS
- 2 - COPAS
- 3 - ESPADAS
- 4 - BASTOS

Las cartas generadas aleatoriamente son asignadas sucesivamente a los elementos de la matriz «M». La primera dimensión indica la posición de la carta dentro de la baraja, y la segunda, el número y el palo.

Al utilizar una función aleatoria pueden repetirse varios valores; para que la matriz «M» no contenga varias cartas idénticas, se ha utilizado la matriz «B». Esta matriz bidimensional de 10 por 4 está inicializada en un principio a cero.

Los valores proporciona-

dos por la subrutina de generación de cartas aleatorias son utilizados como subíndices de la matriz «B»; si el elemento direccionado tiene valor «0», es que la carta no se ha generado anteriormente; si, por el contrario, tiene valor «1», indica que la carta se repite. En el primer caso, los valores de las variables «número» y «palo» se asignan al elemento correspondiente de la matriz «M» y el valor «1» al elemento direccionado de la matriz «B». En el segundo caso, no se asigna ningún valor a la matriz «M» y el programa genera dos nuevos valores que vuelven a ser comprobados.

La rutina encargada de «cortar» la baraja se encuentra localizada en las líneas 610 a 675. En la matriz «C» quedan las cartas ordenadas y listas para poder comenzar la partida.

La transferencia de cartas entre la matriz «M» y «C» se realiza en dos fases. En la primera, se transfieren los elementos situados en las primeras posiciones, es decir, desde el número uno hasta el direccionado por la variable «número»; en la segunda, se realiza la transferencia de las restantes. En la matriz «C» estos dos bloques quedan intercambiados.

La estructura general del programa es:

- 10 : Comentario con el nombre del programa.
- 15 : Asignación del color azul para el borde, verde para el fondo y negro para los caracteres.
- 20-26 : Carátula del programa.
- 40-75 : Asignación de valores a las variables (crédito) y (manos). Esta última contabiliza el número de partidas jugadas.

- 90-125 : Dimensionado de las matrices explicadas anteriormente.
- 140-200 : Bloque de asignación de los nombres de las cartas a la matriz (T\$).
- 251-260 : Datos relativos a los nombres de las cartas.
- 310-360 : Bucles para la asignación de valores.
- 365 : Llamada a la subrutina de visualización de instrucciones.
- 372 : Inicialización de la función aleatoria.
- 380-400 : Mensaje de espera.
- 410-470 : Asignación aleatoria de valores, de carta, a la matriz (M).
- 490-520 : Generación aleatoria de (número) y (palo).
- 540-570 : Introducción del número de carta por donde se desea cortar la baraja.
- 572 : Comprobación de que el valor introducido no es una cadena vacía.
- 574-578 : Bucle para detectar si hay algún valor no numérico dentro de la variable (N\$).
- 580 : Evaluación de la cadena (N\$).
- 590 : Comprobación de que el valor está dentro de los márgenes (1) a (40).
- 592-594 : Mensaje de espera.
- 600 : Cálculo del número de cartas que hay desde la (cortada) hasta el final.
- 610-675 : Transferencia de valores entre las matrices (M) y (C) (cortar la baraja).
- 680-690 : Pantalla de (ánimo).
- 700-715 : Temporización hasta que se pulsa una tecla y borrado de la pantalla.
- 722 : Invitación a jugar.
- 730 : Inicialización de los tantos del jugador.
- 740 : Asignación de la primera carta del jugador.
- 760 : Llamada a la subrutina de presentación de cartas.
- 770 : Asignación de los tantos

- 780 : del jugador.
- 780 : Asignación de la segunda carta al ordenador.
- 800-850 : Presentación de la carta y asignación de su valor.
- 870 : Invitación a pedir otra carta.
- 880-920 : Comprobación de la respuesta.
- 930 : Borrar el mensaje de la zona inferior.
- 932 : Temporización entre dos jugadas.
- 934 : Asignación de la siguiente carta al jugador.
- 950 : Llamada a la subrutina de presentación de cartas.
- 960 : Incrementar los tantos del jugador.
- 970 : Comprueba si el jugador se ha (pasado).
- 980 : Comprueba si el jugador tiene ($7^{1/2}$).
- 990 : Salto al mensaje de invitación.
- 1000-1020 : Mensaje de presentación de los tantos obtenidos por el jugador, en el caso de (pasarse).
- 1030 : Asignación a la variable (ganador) del valor (0), este indica que el ordenador gana.
- 1035 : Inicialización de los tantos del ordenador, para poder visualizar (?).
- 1040-1050 : Temporización hasta que se pulsa una tecla.
- 1080-1090 : Mensaje de obtención, por parte del jugador, de ($7^{1/2}$).
- 1100 : Indica cuál es la siguiente carta que debe (pedir) el ordenador.
- 1110-1120 : Temporiza hasta que se pulsa una tecla.
- 1150-1170 : Mensaje de visualización de los puntos obtenidos por el jugador en el caso de (quedarse).
- 1175 : Borra el mensaje de la zona inferior.
- 1180-1190 : Temporización hasta que se pulsa una tecla.

| | | | | | |
|-----------|---|-----------|--|-----------|---|
| 1220 | : Inicialización de los tantos del ordenador. | 1470 | : Si el jugador se ha «quedado», pide cartas mientras que él tenga menos de seis puntos. | 1740 | : Si el jugador gana y tiene $(7^1/2)$, la apuesta se duplica. |
| 1230 | : Asigna la segunda carta al ordenador. | 1490 | : Mensaje pidiendo otra carta. | 1760-1770 | : Cálculo de los nuevos créditos si gana el ordenador. |
| 1240 | : Indicación de que juega el ordenador. | 1492 | : Duración del mensaje. | 1790-1800 | : Idem. si gana el ordenador. |
| 1252 | : Coordenada (y) de la siguiente carta a visualizar. | 1494 | : Borrado del mensaje. | 1820 | : Visualización de las «manos» jugadas hasta ese momento. |
| 1260 | : Llamada a la subrutina de presentación de cartas. | 1500 | : Temporización entre dos jugadas. | 1830 | : Visualización del nuevo crédito del jugador. |
| 1270 | : Asignación de los tantos del ordenador. | 1510 | : Asignación de la siguiente carta. | 1840 | : Idem. del ordenador. |
| 1280 | : Asignación de la siguiente carta a escoger. | 1522 | : Incrementar, en dos, la siguiente posición. | 1850 | : Comprueba si el ordenador se ha quedado sin crédito. |
| 1300 | : Comprueba si el ordenador se ha «pasado». | 1530 | : Llamada a la subrutina de presentación de cartas. | 1860 | : Idem. del ordenador. |
| 1310 | : Comprueba si el ordenador ha obtenido $(7^1/2)$. | 1540 | : Incrementar los tantos del ordenador. | 1880 | : Mensaje de invitación a jugar otra mano. |
| 1320 | : Comprueba si no se ha pasado. | 1570-1590 | : Mensaje de presentación de los tantos del ordenador, en el caso de «quedarse». | 1890-1930 | : Comprobación de la respuesta elegida. |
| 1330 | : Asignación del valor (1) a la variable «ganador». El ordenador pierde. | 1592-1594 | : Temporiza hasta que se pulsa una tecla. | 1940 | : Borrado del mensaje de la zona inferior. |
| 1340-1360 | : Mensaje de presentación de los tantos del ordenador, en el caso de «pasarse». | 1610 | : Si el ordenador tiene más puntos o igual que el jugador, el ordenador gana. | 1945 | : Incremento de las «manos» jugadas. |
| 1370-1380 | : Temporiza hasta que se pulsa una tecla. | 1620 | : Si tiene menos, el ordenador pierde. | 1950-1990 | : Inicialización a cero de los elementos de la matriz (B); permite volver a barajar las cartas. |
| 1410 | : Asignación del valor (0) a la variable «ganador». El ordenador gana. | 1640 | : Mensaje de que el ordenador gana. | 2000 | : Salto a la rutina de barajar. |
| 1420 | : Mensaje de visualización $(7^1/2)$, por parte del ordenador. | 1650 | : Mensaje de que el jugador gana. | 2100 | : Comienzo de la subrutina «INSTRUCCIONES». |
| 1430-1432 | : Temporiza hasta que se pulsa una tecla. | 1660 | : Visualización de los tantos del ordenador. | 2110 | : Invitación a visualizar las instrucciones. |
| 1460 | : Comprueba si el jugador ha obtenido $(7^1/2)$, para seguir pidiendo cartas hasta que gane o se pase. | 1680 | : Si el jugador se «pasó», el ordenador no indica cuál era su carta. | 2120-2160 | : Comprobación de la operación elegida. |
| | | 1690 | : Visualización de los tantos del ordenador. | 2170-2210 | : Visualización de las instrucciones. |
| | | 1700-1710 | : Temporización hasta que se pulsa una tecla. | 2220-2230 | : Pausa hasta que se pulsa una tecla. |

DEPURACION DE PROGRAMAS

En capítulos anteriores se ha ido explicando la mayor parte del juego de sentencias del Spectrum, así como una serie de conceptos: Bucles, Subrutinas, Funciones, Matrices, etc. Con todo este *material* el iniciado en programación puede ya confeccionar un volumen importante de programas; por este motivo es necesario que conozca una serie de técnicas destinadas a la *depuración* o puesta a punto de programas; también son conocidas por el término inglés *debugging*.

Errores

Es muy difícil que un programa, medianamente complicado, funcione a la primera, ya que hay muchos factores que deben tenerse en cuenta.

Los tipos de error que surgen al elaborar un programa, pueden clasificarse, principalmente, en tres apartados:

- SINTACTICOS
- DE DEFINICION
- DE ESTRUCTURA

Los errores *sintácticos* son aquellos que se producen al teclear un argumento no compatible con una sentencia determinada, por ejemplo:

- NEW 32
- LET "A\$" = "abc"
- CLS fin
- PRINT \$



Zonas de la pantalla.

Este tipo de errores son detectados inmediatamente por el intérprete BASIC, al pulsar «ENTER». Una interrogación parpadeante se posiciona en las proximidades del error, indicando que no entiende dicha sentencia.

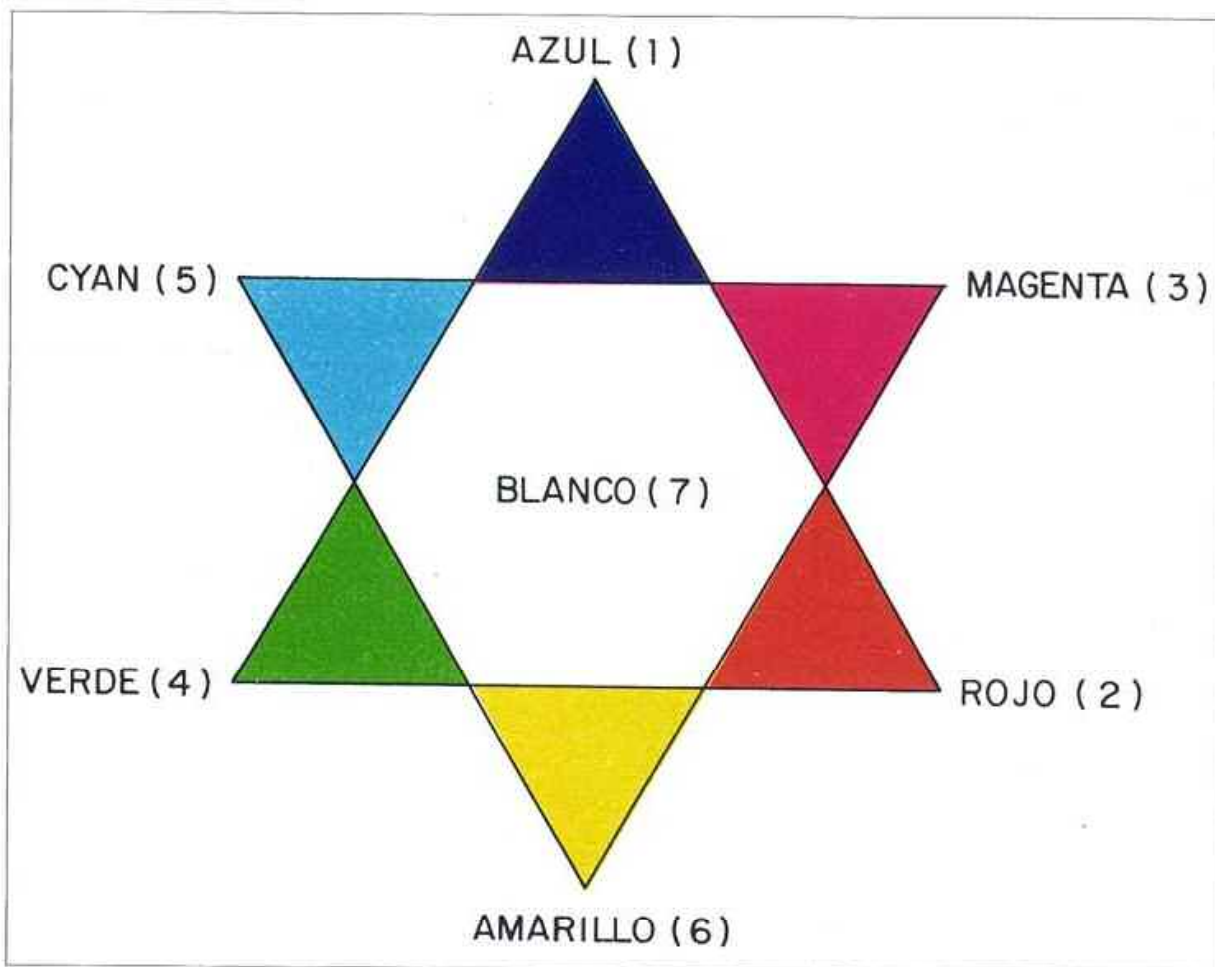
Revise la línea introducida; si no encuentra el fallo, consulte, en el capítulo correspondiente del libro, las posibles estructuras que puede adoptar dicha sentencia.

Como puede observar, los errores sintácticos son detectados y corregidos en la

fase de edición de programas, a diferencia de otros ordenadores en los que se realiza posteriormente.

Otro tipo de errores se manifiestan al ejecutarse el programa, presentando el correspondiente mensaje de error. Estos errores son originados principalmente por no haber definido previamente una variable, por utilizar un subíndice fuera de rango, por utilizar un parámetro erróneo en una función, etc.

El mensaje presentado sirve únicamente de referencia



Estrella de colores.

para encontrar el error, ya que indica el tipo y la línea en que lo ha detectado (ver pag. 96). El número de línea no siempre significa que el error se encuentre en la misma, sino que está relacionado con ella, por ejemplo, un error con mensaje:

```
2 Variable not found, 70:1
```

suponiendo que la línea 70 fuera:

```
70 PRINT mes
```

significa que el argumento de «PRINT» al no ir entre comillas, lo interpreta como variable y no encuentra su asignación inicial. Si efectivamente

«mes» era el nombre de una variable, solucionaremos el problema asignando a esta un valor en la zona del programa que corresponda, por ejemplo:

```
25 LET mes = 8
```

Si «mes» no fuera una variable, sino que por el contrario fuera una cadena a visualizar, el error se encontraría en la misma línea 70, ya que debería ser:

```
70 PRINT (mes)
```

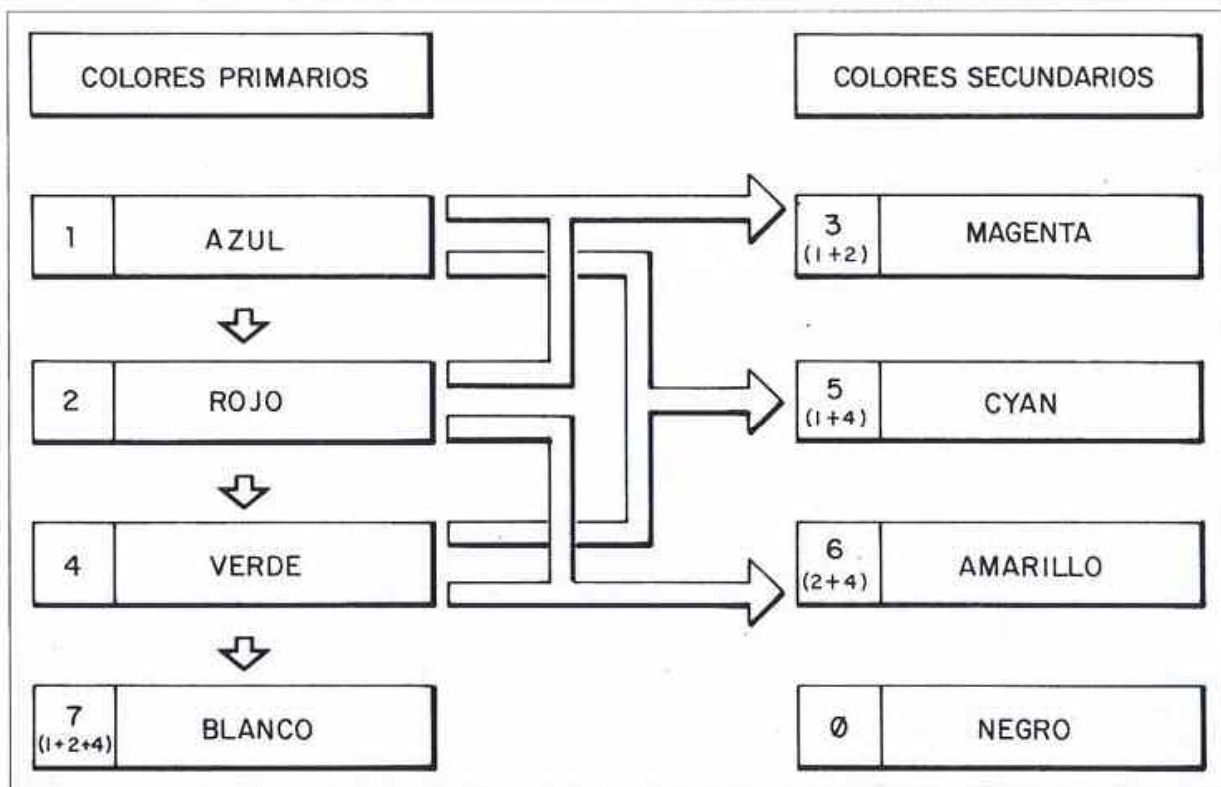
Los errores más difíciles de localizar son aquellos que no generan ningún tipo de mensaje de error, pero que hacen

que nuestro programa no realice las funciones que en un principio teníamos previstas. Al corregir este tipo de errores, de *estructura*, están concebidas las técnicas de depuración.

Depuración

La tarea de depurar no es muy complicada, pero sí laboriosa, aunque a veces basta con echar una ojeada al *listado*, para descubrir el error.

La complejidad de la depuración depende del número de bifurcaciones que tenga el programa, ya que puede ser bastante elevado el número de caminos distintos que siga



Síntesis aditiva de colores.

al ejecutarse.

Tres cosas que facilitan bastante la depuración son:

- DIAGRAMAS DE FLUJO
- LISTADOS
- USO DE SUBROUTINAS

El tener confeccionado el diagrama de flujos nos permite ver las funciones que realiza el programa paso a paso y los diversos caminos que debe seguir según se cumplan ciertas condiciones. Podemos localizar, con su ayuda,

la zona o el bloque de instrucciones donde falla el programa.

Si se dispone de una impresora, podemos obtener un listado y así localizar con mayor facilidad las sentencias que consideremos oportunas.

El uso de subrutinas facilita la depuración, ya que éstas pueden ser comprobadas individualmente, de esta manera se pueden tener ciertas garantías de que funcionan cuando se depura el programa final.

Es importante tener localizada la zona a partir de la cual se desea comenzar la depuración, por ejemplo, en el siguiente programa, que tiene dos errores, empezariamos a depurar a partir de la línea 40, ya que si lo ejecuta observará que se visualizan las constantes de cadena de las líneas 20 y 30.

Para corregir el ejemplo, deberá modificar la línea 60

```
60 PRINT AT n + 2,0; a$
```

PROGRAMA 1

```
10 REM *****
   * ERRORES *
   * *****
20 PRINT "ARTICULO","PRECIO"
30 PRINT "_____";"_____"
```

```
40 FOR n=1 TO 5
50 INPUT "Artículo > ";a$
60 PRINT AT 3,0;a$
70 INPUT "Precio > ";p
80 NEXT n
90 PRINT AT n+2,16;p
```

e introducir la línea 90 dentro del bucle, asignándola un nuevo número de línea comprendido entre 71 y 79.

Lamentablemente el Spectrum no dispone de dos sentencias muy potentes utilizadas en otros ordenadores para depurar. Estas son:

- TRON
- TROFF

que permiten habilitar o deshabilitar la facilidad de *traza*, con ella se puede visualizar en pantalla la secuencia de instrucciones *ejecutadas* por el ordenador, paso a paso y, comprobar si es correcta.

El Spectrum utiliza para depurar las sentencias «STOP» y «CONTINUE», utilizando la primera como *breakpoint* o punto de ruptura.

STOP y CONTINUE

La utilización de estas sentencias es bastante sencilla, pero si tiene alguna duda consulte las páginas 91 a 96.

Básicamente la depuración

con estas sentencias consiste en colocar en lugares estratégicos del programa, diversos puntos de ruptura con la sentencia «STOP».

Al ejecutarse deberá pararse en el primer punto, visualizando el manejo correspondiente, si no lo encuentras es que el fallo está localizado entre la línea de arranque y dicho «STOP». Si por el contrario se para, podremos mediante comandos directos conocer el valor de las variables utilizadas; si su contenido es correcto introduciremos también como comando directo la sentencia «CONTINUE» y el programa continuará su ejecución hasta el próximo punto de ruptura, donde haremos las mismas operaciones.

De esta manera iremos ejecutando por partes el programa hasta que localicemos el fallo.

Algunos de los puntos estratégicos para la colocación de los «STOP» son las bifurcaciones, es decir, donde el programa pregunta si se cumple una condición im-

puesta en un «IF ... THEN ...».

Una vez parado un programa si se desea continuar con su ejecución, en una línea determinada, debe utilizar «GO TO» o «GO SUB», si se trata de una subrutina, ya que si utiliza «RUN n» todas las variables que tenía definidas se borran y posiblemente aparezcan el error:

2 Variable not found

por que la definición se encontraba en las líneas anteriores.

Cuando el programa esté corregido deben suprimirse las líneas con las sentencias «STOP», utilizadas como puntos de ruptura.

Programa «Depurador»

Como alternativa al uso «STOP» y «CONTINUE» el programa «1» realiza las mismas funciones, pero tiene la ventaja de una mayor facilidad de uso.

El programa que usted quiera depurar deberá estar

PROGRAMA 2

```

10 REM *****
   *          *
   *  CURSO/BASIC *
   *          *
   * ***** *
   *  11  ERRORES *
   *          *
   * ***** *

20 BORDER 4: PAPER 4: INK 0: C
L5 30 REM VARIABLES: LET logo=1
000
35 LET igual=1100
40 GO TO logo
50 INPUT "Cadena 1 >>> ";a$
60 IF a$="" THEN GO TO 1000
70 IF LEN a$>20 THEN GO TO 50
80 PRINT AT 3,0;"Cadena 1: ";a
$
90 INPUT "Cadena 2 >>> ";b$
100 IF b$="" THEN GO TO 90
110 IF LEN b$>20 THEN GO TO 90
120 PRINT AT 10,0;"Cadena 2: ";
a$
130 IF a$=b$ THEN GO SUB igual
140 GO SUB distinto

```

```

150 REM CONTINUAR
160 PRINT #0;"Desea continuar (
S/N)"
170 PAUSE 0
180 LET z$=INKEY$
190 IF z$="S" OR z$="s" THEN GO
TO 190
200 IF z$="N" OR z$="n" THEN CL
S: PAUSE 0: STOP
210 GO TO 170
1000 REM LOGO
1005 RESTORE
1010 FOR n=1 TO 32
1020 READ dato
1030 PRINT INVERSE 1;AT 0,n;CHR$
dato
1040 NEXT n
1050 DATA 32,77,73,90,82,79,72,7
9,66,66,89,32,83,69,77,86,76,65,
76,32,32,32,32,32,32,32,32,32,
32
1060 RETURN
1100 PRINT AT 7,0;"Cadenas *1* y
*2* distintas"
1110 RETURN
1200 PRINT AT 7,0;"Cadenas *1* y
*2* iguales"
1210 RETURN

```

PROGRAMA 3

```

10 REM *****
   *          *
   *  CURSO/BASIC  *
   *          *
   * *****
   * SIN ERRORES  *
   *          *
   * *****
20 BORDER 4: PAPER 4: INK 0: C
LS
30 REM VARIABLES
32 LET logo=1000
35 LET distinto=1100
37 LET igual=1200
40 GO SUB logo
50 INPUT "Cadena 1 >>> ",a$
60 IF a$="" THEN GO TO 50
70 IF LEN a$>20 THEN GO TO 50
80 PRINT AT 3,0;"Cadena 1: ",a
$
90 INPUT "Cadena 2 >>> ",b$
100 IF b$="" THEN GO TO 90
110 IF LEN b$>20 THEN GO TO 90
120 PRINT AT 5,0;"Cadena 2: ",a
$
130 IF a$=b$ THEN GO SUB igual:
GO TO 150

```

```

140 GO SUB distinto
150 REM CONTINUAR
160 PRINT #0;"Desea continuar (
S/N)"
170 PAUSE 0
180 LET z$=INKEY$
190 IF z$="S" OR z$="s" THEN GO
TO 10
200 IF z$="N" OR z$="n" THEN CL
S: PAUSE 0: STOP
210 GO TO 170
1000 REM LOGO
1005 RESTORE
1010 FOR n=0 TO 31
1020 READ dato
1030 PRINT INVERSE 1,AT 0,n;CHR$
dato
1040 NEXT n
1050 DATA 32,77,73,67,82,79,72,7
9,68,68,89,32,83,69,77,65,78,65,
76,32,32,32,32,32,32,32,32,32
,32,32
1060 RETURN
1100 PRINT AT 7,0;"Cadenas *1* y
*2* distintas"
1110 RETURN
1200 PRINT AT 7,0;"Cadenas *1* y
*2* iguales"
1210 RETURN

```

ubicado en las líneas de «1» a la «9978», para no crear conflicto. Suponiendo que al salvarlo le haya llamado «debug», podrá *combinar* ambos programas de la siguiente manera:

- Deberá cargar primero el que quiera depurar, si no lo tiene en memoria.
- Para cargar el programa «debug» utilice:

```
MERGE debug
```

Cuando los dos estén en memoria, inserte el siguiente comando directo para facilitar su manejo:

```
LET debug = 9980
```

Para colocar los puntos de ruptura utilice, en lugar de «STOP», la sentencia:

```
GO SUB debug
```

Una vez ejecutado el programa a depurar, cuando encuentra un punto de ruptura se ejecuta la subrutina «debug» que presenta el mensaje:

```
Parada en...
```

indicando la línea y la sentencia dentro de la línea.

Si se pulsa cualquier tecla menos la «V» el programa continúa hasta el próximo punto de ruptura. Por el contrario, si se pulsa la «V», la subrutina nos pregunta que variable deseamos visualizar:

```
Variable >
```

introduciendo el nombre y pulsando «ENTER» nos aparece su contenido.

Pulsando cualquier tecla nos vuelve a preguntar el nombre de otra variable, si no deseamos conocer el conte-

nido de ninguna más, basta con pulsar «ENTER» para que el programa principal continúe hasta el próximo punto de ruptura.

Cuando el programa esté depurado deberá eliminar todos los puntos de ruptura así como las líneas «9970» a «9999».

Ejercicio

Como programa-ejercicio de este capítulo, edite el número 2. Se encuentra, lógicamente libre de errores *sintácticos*, pero se han introducido «11» errores de *definición* y de *estructura*. Intente, con la ayuda de las técnicas explicadas, localizar y corregir dichos errores.

Si se encuentra «desesperado» o desea comparar sus resultados, el programa número 3 ofrece una de las posibles soluciones.

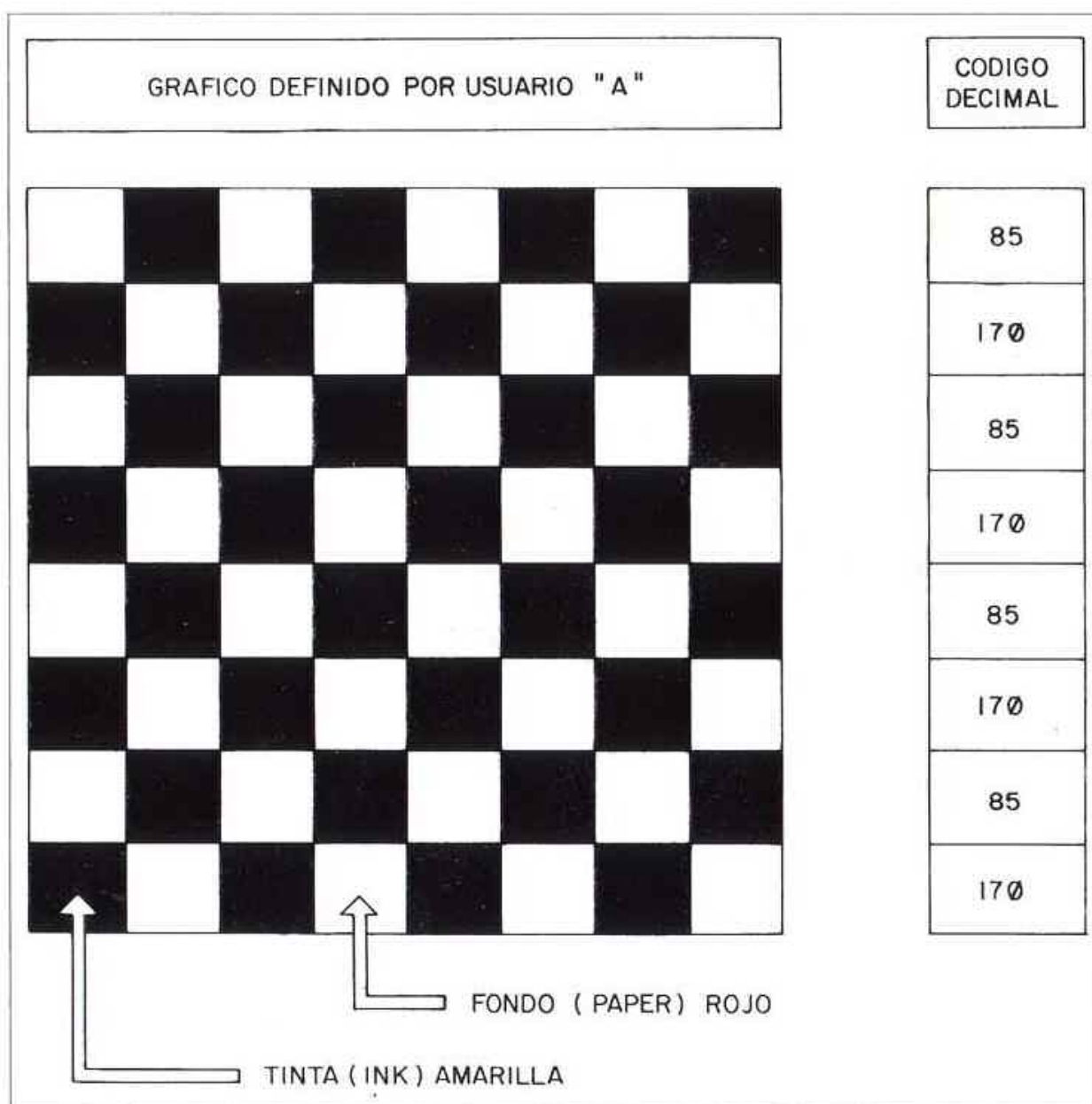
COLOR

El *color* es actualmente una de las características a tener en cuenta cuando se adquiere un ordenador personal, principalmente si lo piensa

destinar a programas de juegos, aunque en otro tipo de aplicaciones tampoco está de más añadir algo de color, bien sea por estética o para

destacar algún mensaje o zona de pantalla.

El «ZX Spectrum», hermano mayor del conocido «ZX-81», debe precisamente su nombre a la capacidad que tiene



Rejilla utilizada en la simulación de colores (Ej. naranja)



Carta de ajuste

para generar los colores del espectro luminoso.

El Spectrum dispone de ocho colores que pueden conseguirse en dos gamas de brillo; cada uno tiene asignado un número que lo identifica a la hora de programar.

| CODIGO | COLOR |
|--------|----------|
| 0 | NEGRO |
| 1 | AZUL |
| 2 | ROJO |
| 3 | MAGENTA |
| 4 | VERDE |
| 5 | CYAN |
| 6 | AMARILLO |
| 7 | BLANCO |

Este código está en función de la luminosidad del color, así el negro o ausencia de luz, tiene el código cero y según va progresando la luminosidad aumenta el valor hasta llegar al blanco que tiene el

código siete. La luminosidad de los colores es difícil de apreciar ya que el ojo humano es más sensible a ciertos colores, los cuales, nos parecen más claros.

El programa n.º «1» muestra unas barras verticales con diversos colores, éstos están ordenados según su código. Si dispone de un televisor en blanco y negro podrá comprobar la luminosidad de cada color ya que cada uno toma un valor distinto de gris; el resultado es una escala degradada de grises. Obtendrá el mismo efecto en un televisor de color, si éste es anulado; el programa también sirve para poder sintonizar correctamente el televisor.

Teoría del color

Antes de explicar la utiliza-

ción de los comandos que afectan sobre el color de las diversas zonas de la pantalla, es conveniente tener algunos conocimientos sobre la teoría del color.

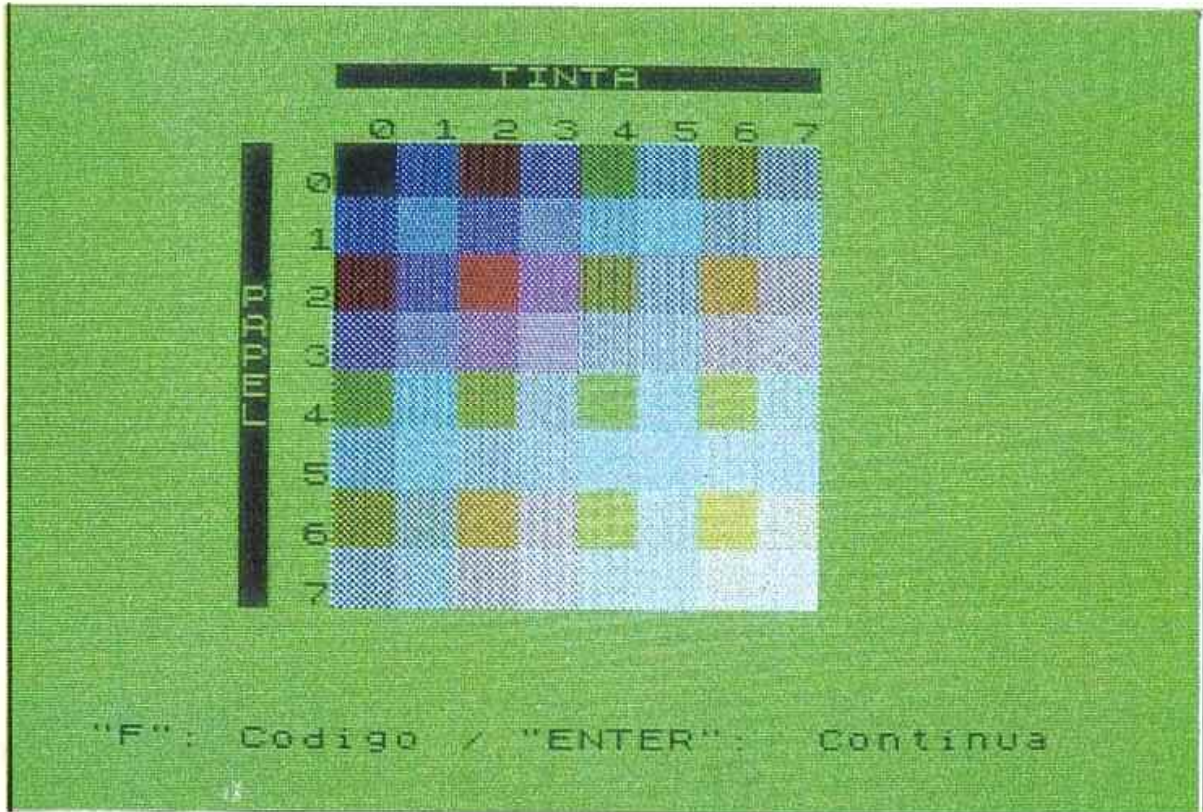
La descomposición de la luz blanca, por un prisma, se reduce básicamente a tres colores:

- AZUL
- ROJO
- VERDE

éstos son llamados «primarios». A partir de éstos, y por *síntesis aditiva*, pueden obtenerse los llamados «secundarios».

- MAGENTA
- CYAN
- AMARILLO

Se denominan colores



«64» Colores

complementarios aquéllos que al mezclarse, por síntesis aditiva, dan origen, de nuevo, a la luz blanca. En la estrella de los colores, representada en la figura, pueden observarse los colores complementarios, ya que éstos se encuentran enfrentados:

| |
|-----------------|
| AZUL - AMARILLO |
| ROJO - CYAN |
| VERDE - MAGENTA |

Lógicamente, al recomponer los tres colores primarios se obtiene también la luz blanca.

Síntesis aditiva

Esta se demuestra con la utilización de tres proyectos de luz blanca, en el primero insertamos un filtro de color

rojo, en el segundo uno verde y en el tercero uno azul.

Los proyectores se colocan de manera que sus haces coincidan según la figura adjunta.

Se comprueba que en el área iluminada por los tres proyectores se obtiene luz blanca; en el área de inserción de la luz roja con la verde, el amarillo; en el caso de la luz roja y del azul, resulta el magenta y por último, de la mezcla de la luz verde y azul, de cyan.

| | |
|---------------------|----------|
| ROJO + VERDE + AZUL | BLANCO |
| ROJO + VERDE | AMARILLO |
| ROJO + AZUL | MAGENTA |
| VERDE + AZUL | CYAN |

Comprobemos que en el Spectrum se cumple la síntesis aditiva, sumando los respectivos códigos de color.

a) Blanco:

| | |
|--------|---|
| Azul | 1 |
| Rojo | 2 |
| Verde | 4 |
| Blanco | 7 |

b) Magenta:

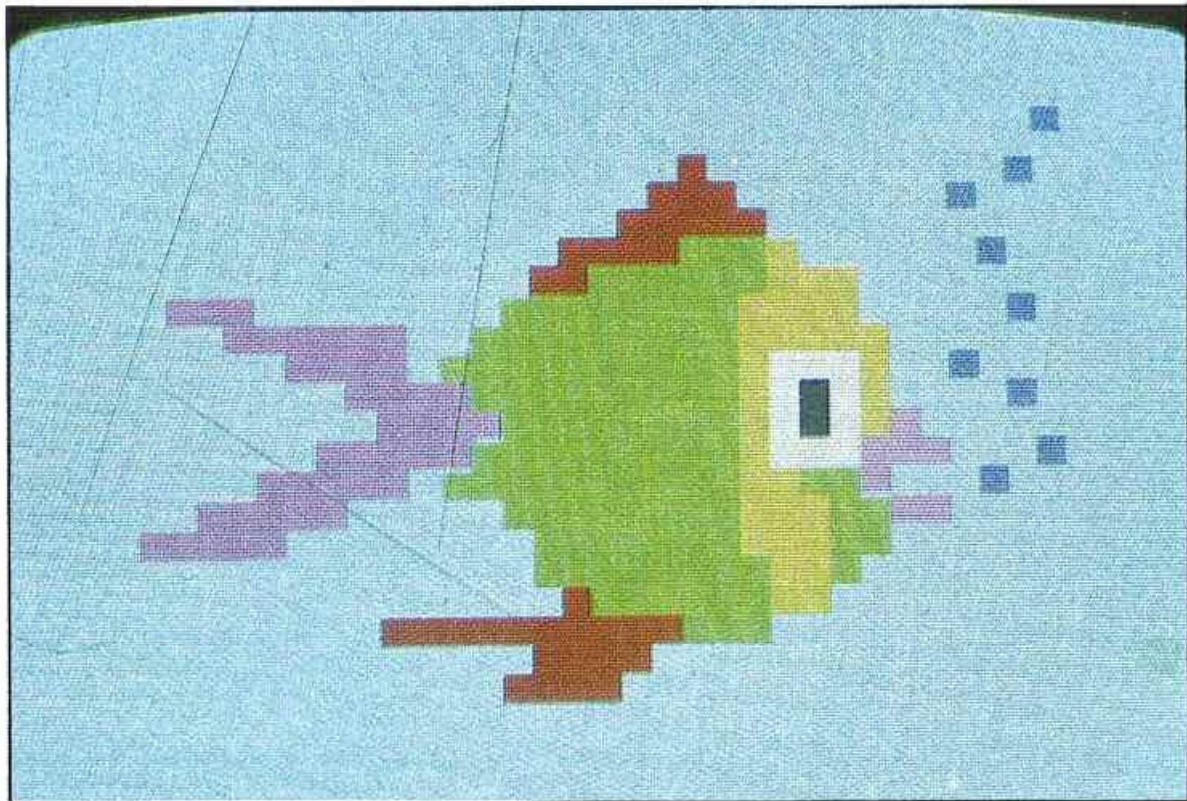
| | |
|---------|---|
| Azul | 1 |
| Rojo | 2 |
| Magenta | 3 |

c) Cyan:

| | |
|-------|---|
| Azul | 1 |
| Verde | 4 |
| Cyan | 3 |

d) Amarillo:

| | |
|----------|---|
| Rojo | 2 |
| Verde | 4 |
| Amarillo | 6 |



Las posibilidades del color

Zonas de pantalla

Existen dos zonas que pueden ser modificadas por los comandos de control de color:

- BORDE
- PAPEL

El *borde*, como su propio nombre indica, es la zona formada por el margen exterior de la pantalla del televisor, en éste no se pueden imprimir caracteres.

El *papel* o fondo es el rectángulo central, rodeado por el «borde», donde se imprimen los caracteres o gráficos.

Las siguientes instrucciones muestran estas zonas:

```

10 REM *****
   BORDE/PAPEL
*****

20 BORDER 2
30 PAPER 5
40 CLS
50 PRINT @@; AT 1,10; "BORDE rojo"
60 PRINT AT 10,5; "PAPEL azul"
70 PAUSE 0
  
```

El mensaje «BORDE rojo» está visualizado en la línea 23 (zona reservada), ésta al igual que la zona 22, toman el color del «borde».

El color en que visualizamos los caracteres se denomina *tinta*.

Comandos

```
BORDER
```

Acceso al teclado

BIN



MODO E

BRIGHT

Definición

Con este comando puede alterarse el color del borde. La estructura general es:

| SENTENCIA | ARGUMENTO |
|-----------|--------------|
| BORDER | código color |

Ejemplos:

- BORDER 2 (rojo)
- BORDER 3 (magenta)
- BORDER 0 (negro)
- BORDER 7 (blanco)

Observará que nada más introducir el comando, el borde cambia de color.

El siguiente programa cambia el color del borde cuando se pulsa una tecla; si la mantiene constantemente pulsada, éste cambia de color a toda velocidad.

```

10 REM *****
   BORDER
*****

20 RESTORE
30 READ a$,color
35 IF a$="FIN" THEN GO TO 20
40 BORDER color
45 CLS
50 PRINT @B,AT 1,10,"BORDE ",a
$

55 PAUSE 10
60 IF INKEYS="" THEN GO TO 60
65 GO TO 30
70 DATA "negro",0,"azul",1,"rojo",2,"magenta",3,"verde",4,"cyan",5,"amarillo",6,"blanco",7,"FIN",0

```

PAPER

Acceso al teclado

L PRINT



MODO **E**

PAPER

Definición

Permite modificar el color del fondo o papel.

Su estructura general es la siguiente:

| SENTENCIA | ARGUMENTO |
|-----------|--------------|
| PAPER | código color |

Ejemplos:

- PAPER 2
- PAPER «MICROHOBBY»
- PAPER 5
- PRINT «SEMANAL»

Cuando se introduce una sentencia de este tipo, observará, que el fondo no cambia de color; para que los atributos de éste cambien, debe pulsarse dos veces consecutivas la tecla «ENTER», cuando el comando es directo, o introducir a continuación el comando «CLS»; ya que de lo contrario, el fondo cambia de color únicamente en las zonas donde se imprime algo.

El siguiente programa altera, de forma parecida al anterior, el color del papel:

```

10 REM *****
   PAPEL
*****

20 BORDER 7
30 RESTORE
40 FOR n=0 TO 7
50 READ a$
50 PAPER n
70 CLS
80 PRINT AT 10,8,"PAPEL ",a$
85 PAUSE 10
90 IF INKEYS="" THEN GO TO 90
100 NEXT n
110 GO TO 30
120 DATA "negro","azul","rojo","magenta","verde","cyan","amarillo","blanco"

```

Observará que, cuando el fondo es negro, no se visualiza ningún mensaje, ya que coinciden el color del papel y tinta.

INK

Acceso al teclado

EXP



MODO **E**

INK



Definición

El color de la tinta o del carácter es modificado con esta sentencia.

Su estructura general es:

| ESTRUCTURA | ARGUMENTO |
|------------|--------------|
| INK | código color |

Ejemplos:

- INK 0
- PRINT «MICRO»;
- INK 2
- PRINT «HOBBY»

Para que los atributos cambien debe operarse de la misma manera que en el caso de «PAPER».

Las siguientes instrucciones cambian secuencialmente el color de la tinta:

```

10 REM *****
   TINTRA
*****

20 BORDER 7 PAPER 7: CLS
30 RESTORE
40 FOR n=0 TO 7
50 READ a$
50 INK n
70 CLS
80 PRINT AT 10,8,"TINTRA ",a$
85 PAUSE 0
90 IF INKEYS="" THEN GO TO 100
100 NEXT n
110 GO TO 30
120 DATA "negro","azul","rojo","magenta","verde","cyan","amarillo","blanca"

```

Por la misma razón que en el programa anterior, el mensaje, en tinta blanca, no se visualiza.

Ejecuta el siguiente programa que combina todas las posibilidades de «borde», «papel» y «tinta».


```

10 REM *****
   : BORDE/PAPEL/TINTA
   : *****
20 FOR x=0 TO 7
30 FOR y=0 TO 7
40 FOR z=0 TO 7
50 BORDER x
50 PAPER y
70 INK z
80 CLS
90 PRINT AT 10,7;"MICROHOBBY S
EMANAL"
100 PRINT 80;" Pulse una tecla
para continuar"
110 PAUSE 0
120 NEXT z
130 NEXT y
140 NEXT x
150 INK 0: CLS

```

Los colores, por defecto, que presenta el Spectrum al conectarlo, son:

```

Borde blanco
Papel blanco
Tinta negra

```

Atributos permanentes y temporales

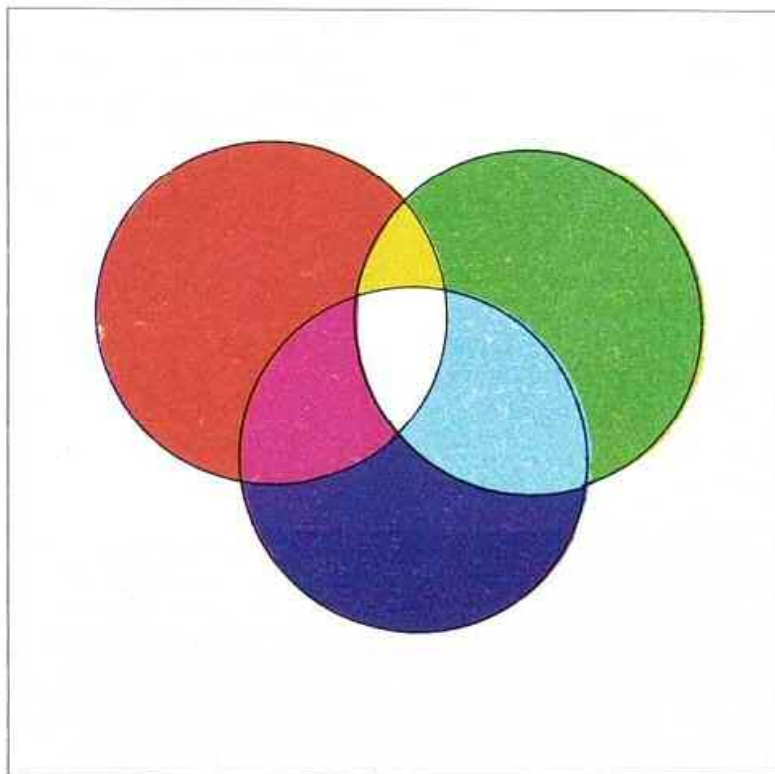
Como se comentó en la página 71, la zona de representación, destinada al usuario, está constituida por 22 líneas y 32, esto hace un total de «704» caracteres. Cada posición, de estos caracteres, posee unos *atributos* que determinan, entre otras cosas, el color del «papel» y el de «tinta»; estos atributos son fijados mediante ciertos parámetros que pueden ser:

- PERMANENTES
- TEMPORALES

Los atributos *permanentes* son fijados con las sentencias «BORDER», «PAPER», «INK», etc. éstos permanecen en la memoria de *presentación visual* hasta que se vuelven a modificar, de manera que todas las sentencias que tengan que ver con la visualización de caracteres o gráficos lo harán en los colores a tinta y papel especificados.

Ejemplo:

216 MICROBASIC



Demostración síntesis aditiva

```

10 REM *****
   : PERMANENTES
   : *****
12 LET azul=1: LET rojo=2: LET
verde=4
14 LET magenta=3: LET cyan=5:
LET amarillo=6
16 LET negro=0: LET blanco=7
20 INPUT "Color borde > ";bord
30 INPUT "Color fondo > ";fond
40 INPUT "Color tinta > ";tint
50 BORDER borde
50 PAPER fondo
70 INK tinta
80 CLS
90 INPUT "Mensaje > ";a$

100 PRINT a$
110 GO TO 90

```

Los atributos *temporales* se utilizan para visualizar uno o varios caracteres con unos atributos distintos de los especificados.

Las sentencias «PRINT», «INPUT», etc. deben ir acompañadas en este caso, de las conocidas «PAPER» e «INK».

Ejemplo:

```
PRINT PAPER 2, "HOLA"
```

que visualiza la cadena alfanumérica "HOLA" sobre fondo verde, independientemente de los atributos permanentes que tuviera.

Veamos otro ejemplo aclaratorio:

```

10 REM *****
   : TEMPORALES
   : *****
12 LET azul=1: LET rojo=2: LET
verde=4
14 LET magenta=3: LET cyan=5:
LET amarillo=6
16 LET negro=0: LET blanco=7
20 INPUT "Color borde > ";bord
30 INPUT "Color fondo > ";fond
40 INPUT "Color tinta > ";tint
50 BORDER borde
50 PAPER fondo
70 INK tinta
80 CLS
90 INPUT "Mensaje 1 > ";a$

100 PRINT a$
110 INPUT "Color de mensaje 2 > ";temporal
120 INPUT "Mensaje 2 > ";b$
130 PRINT INK temporal;b$
140 GO TO 90

```

Resolución del color

Cada carácter está formado por una matriz de ocho por ocho puntos, también conocidos por el término inglés «pixel»; por tanto hay 64. A pesar de que podemos activar individualmente cada punto, con las sentencias utilizadas en la realización de gráficos en alta resolución, no se puede, sin embargo, asignar un color de tinta distinto para cada uno de ellos, ya que cada carácter está controlado por unos atributos, bien permanentes o temporales. Estos afectan a cada matriz de 64 pixels, por tanto sólo puede haber dos colores distintos en cada posición de carácter.

Esta configuración se denomina «color en baja resolución».

Transparencia y contraste

Las sentencias «PAPER» e «INK» pueden tener como argumento los códigos de color 8 y 9, teniendo un significado de *transparencia* y *contraste* respectivamente.

La transparencia consiste en conservar los atributos temporales de la pantalla al imprimir un nuevo carácter.

Ejemplos:

a) Papel transparente:

```
10 REM *****
   PAPER "8"
   *****
12 BORDER 2 : PAPER 2 : INK 0 : CLS
L5
20 FOR I=1 TO 22
30 FOR E=3 TO 6
40 PRINT PAPER €;" "
50 NEXT E
60 NEXT I
70 PRINT AT 7,7;"MICROHOBBY SE
MANUAL"
80 PRINT AT 14,7:"PAPER 8:"MIC
ROHOBBY SE"MANUAL"
```

El mensaje de la línea 70 «machaca» los atributos temporales, imprimiéndose sobre fondo verde, ya que este es el atributo permanente especificado en la línea 12.

Sin embargo, el mensaje de la línea 80 conserva estos atributos.

b) Tinta transparente.

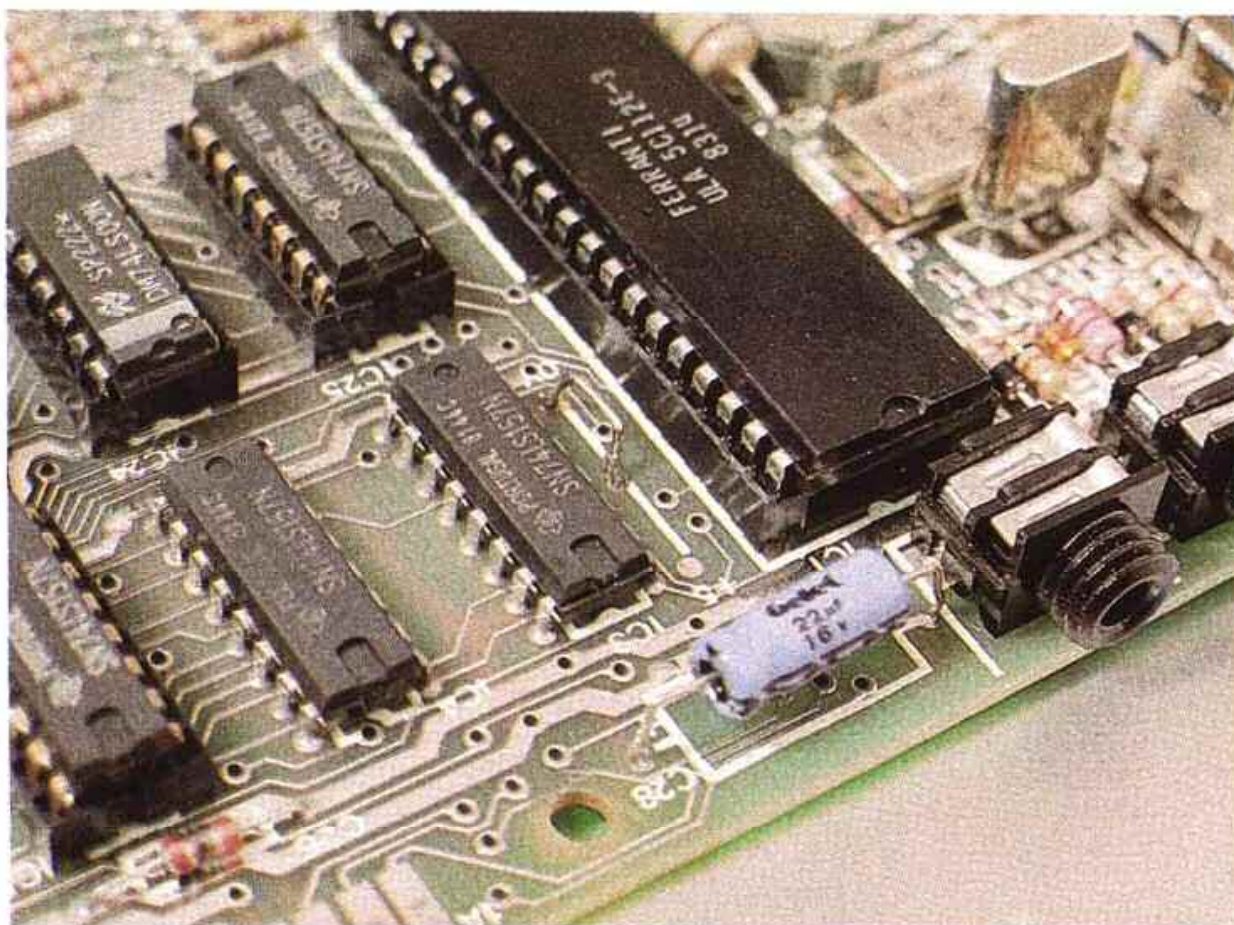
Introduzca el siguiente comando directo que selecciona los colores «rojo» para borde y fondo, y «negro» para tinta.

```
BORDER 2 : PAPER 2 : INK 0 : CLS
```

Teclee el siguiente bucle:

```
FOR n = 0 TO 7 :
PRINT AT n,0; INK n; (MICROHOBBY);
NEXT n
```

que visualiza la cadena «Mi-



Circuito generador de colores.

CROHOBBY» con diversos colores temporales de tinta.

Introduzca ahora este otro bucle:

```
FOR n = 0 TO 10:
PRINT AT n,0: INK 8:
(MICROHOBBY SEMANAL):
NEXT n
```

que visualiza la cadena «MICROHOBBY» con diversos colores temporales de tinta.

Introduzca ahora este otro bucle:

```
FOR n = 0 TO 10:
PRINT AT n, 0: INK 8:
(MICROHOBBY SEMANAL):
NEXT n
```

Observe como son respetados los atributos temporales de tinta.

El contraste es una característica que indica que el color correspondiente a papel o tinta va a ser en blanco o negro para destacar sobre el otro.

El color de contraste es «negro» cuando el otro es de tonalidad clara, y «blanco» cuando es oscura.

| CONTRASTE | COLOR |
|-----------|----------|
| BLANCO | NEGRO |
| | AZUL |
| | ROJO |
| | MAGENTA |
| NEGRO | VERDE |
| | CYAN |
| | AMARILLO |
| | BLANCO |

El siguiente programa cambia el color de borde y papel; observe el color de la tinta.

```
10 REM *****
: CONTRASTE
: *****
```

```
20 FOR n=0 TO 7
30 BORDER 0: PAPER 0: CLS
40 PRINT AT 10,10: INK 9: "MICROHOBBY"
50 PRINT 80: " PULSE UNA TECLA PARA CONTINUAR " PAUSE 0
60 NEXT n
70 GO TO 20
```

Tanto los mensajes que envía el ordenador como los que usted visualiza a través de los canales de comunicación cero y uno (0 y 1). tienen la característica de tinta de contraste, en relación con el color del borde.

Edite el siguiente programa que genera con colores temporales, una serie de barras verticales.

```
10 REM *****
: TINTA "9"
: *****
20 FOR x=0 TO 21
30 FOR z=0 TO 7
40 FOR y=0 TO 7
50 PRINT PAPER z: " "
60 NEXT z
70 NEXT y
80 NEXT x
90 PRINT AT 7,7: PAPER 8: "MICROHOBBY SEMANAL"
100 PRINT AT 14,7: PAPER 8: INK 9: "MICROHOBBY SEMANAL"
```

Los mensajes de las líneas 90 y 100 están visualizados sobre papel transparente. El primer mensaje no se puede leer correctamente debido al poco contraste que existe en algunas zonas, entre el papel y la tinta; sin embargo, el segundo aprovecha la característica de contraste de tinta para ser legible.

En este otro ejemplo se demuestra la característica de papel de contraste:

```
10 REM *****
: PAPEL "9"
: *****
20 BORDER 2: PAPER 2: CLS
30 FOR n=7 TO 14 STEP 7:
40 PRINT AT n,7: INK 0: " MICROHOBBY SEMANAL "
50 PRINT INK 2: "HOBBY "
60 PRINT INK 6: " SEMANAL "
70 PAPER 9
80 NEXT n
```

Con el papel de contraste, aparte de quedar el mensaje destacado, se visualiza la cadena «HOBBY».

Simulación de colores

Aparte de los ocho colores de que dispone el Spectrum, pueden simularse hasta «28» más, disponibles también en dos gamas de brillo, por tanto pueden conseguirse en total hasta «72» tonalidades distintas [(28 + 8) * 2].

La forma de simularlos es bastante simple. Utilizando una rejilla pequeña, similar a un tablero de ajedrez, en que los cuadros blancos corresponden al fondo y los negros a la tinta, se observa que asignando diversos colores al papel y a la tinta, y situada la rejilla a cierta distancia, el ojo integra ambos colores dando como resultado uno distinto que es la mezcla; por ejemplo, con el rojo y el amarillo simularíamos el naranja.

Con el programa número «2», que utiliza los gráficos definidos, asignamos a la letra «A» una rejilla similar a la explicada.

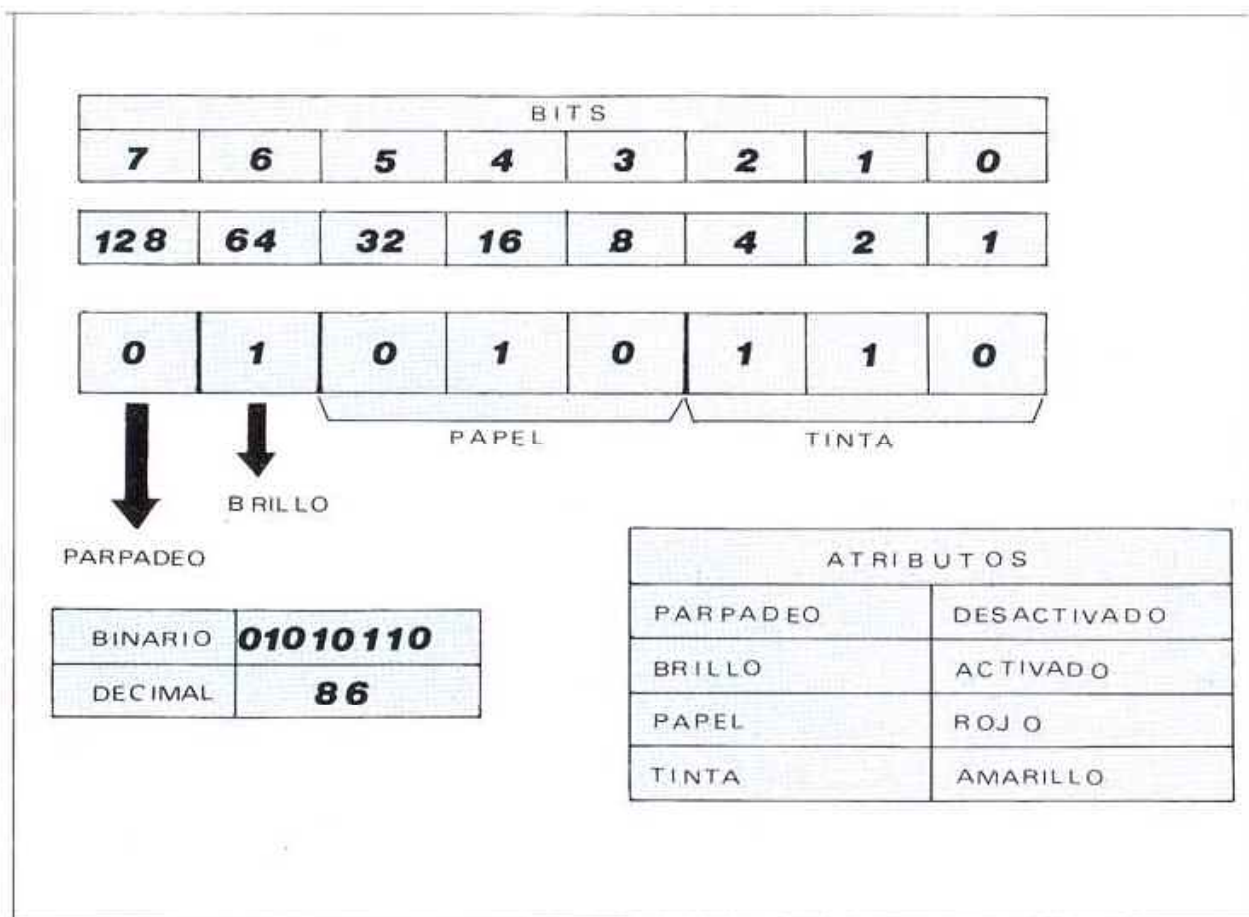
No se preocupe si al ejecutarse la línea 60 el programa desaparece, visualizándose el conocido mensaje:

© 1982 Sinclair Research Ltd

ya que el gráfico permanece en memoria, podrá comprobarlo, seleccionando el modo G (gráficos) y pulsando la letra «A».

Retorne al modo anterior y ejecute el programa n.º «3».

Dicho programa nos presenta, utilizando la rejilla, todas las combinaciones de tin-



Ejemplo función «ATTR».

ta y papel. Observará que algunos colores se encuentran repetidos ya que la combinación de papel «3» y tinta «5», es la misma que papel «5» y tinta «3».

Si pulsamos cualquier tecla, los colores se visualizan en una gama de brillo distinta. Si por el contrario pulsamos «CAPS SHIFT» simultáneamente con la tecla «F», accedemos a la opción «CODIGO».

El código debe ser introducido de la siguiente manera:

- Teclar el código del papel (0 - 7)
- Teclar el código de la tinta (0 - 7)
- Pulsar "ENTER".

Este código selecciona el color simulado por la mezcla de ambos, el cual será visualizado con su brillo correspondiente, en la zona de pantalla

conocida como fondo.

Pulsando cualquier tecla se accede otra vez al «muestreo» de colores.

Control de impresión

Con el uso de ciertos comandos se pueden alterar las siguientes características de impresión:

- BRILLO
- INVERSION DE VIDEO
- PARPADEO

Estas, al igual que los colores, pueden ser permanentes o temporales.

El argumento de estas sentencias indica si se desea activar o desactivar cierta característica, utilizando para ello un código de control.

| CODIGO | SIGNIFICADO |
|--------|-------------|
| 0 | DESACTIVADO |
| 1 | ACTIVADO |

BRIGTH

Acceso al teclado

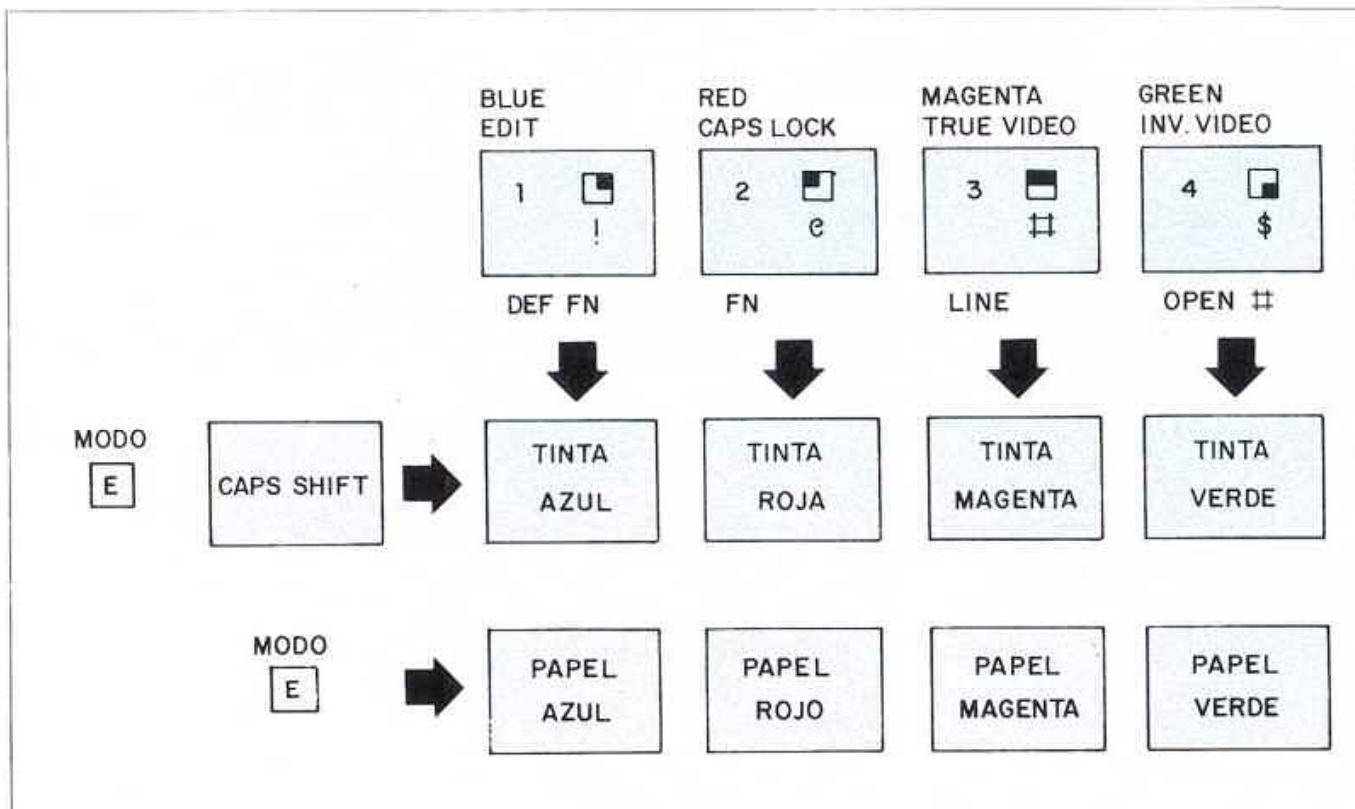
MODO **E**

SYMBOL SHIFT

Definición

El comando «BRIGTH» permite modificar el brillo de los colores.

La estructura general de esta sentencia es:



Acceso directo al color

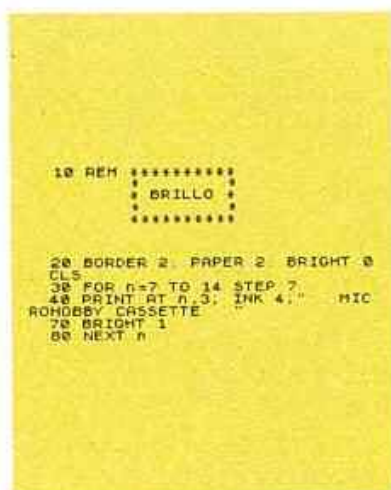
| SENTENCIA | ARGUMENTO |
|-----------|-------------------|
| BRIGHT | código de control |

Ejemplos:

- BRIGHT 0
- PRINT BRIGHT 1; "HOLA"
- BRIGHT 1
- INPUT BRIGHT 1; ">"; a

La opción por defecto es brillo desactivado. El brillo afecta tanto al color del papel como al de la tinta.

Ejemplo:



«BRIGH» también puede tener como código de control el número «8», con el significado de transparente.

Ejemplo:



Acceso al teclado



Definición

Básicamente esta sentencia intercambia los colores de la tinta y papel. Véase en la página 70 la forma de acceder directamente a esta función y en la 81 una figura alusiva.

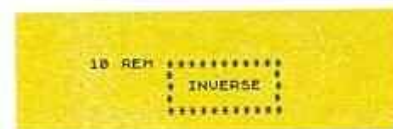
Su estructura general es:

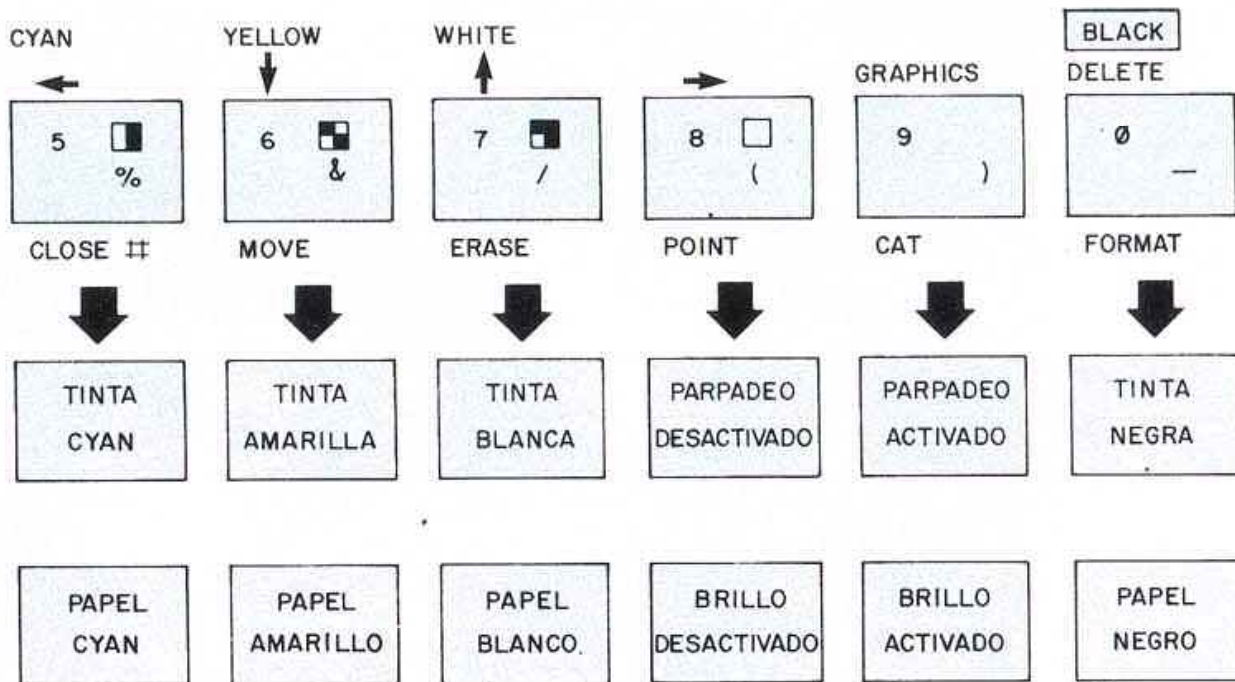
| SENTENCIA | ARGUMENTO |
|-----------|-------------------|
| INVERSE | código de control |

Ejemplos:

- INVERSE 1
- PRINT INVERSE 1; "JUAN"
- INPUT INVERSE 1; "edad?"; a
- INVERSE 0

También en este caso la opción por defecto es "VIDEO TRUE". Veamos un ejemplo:





```
20 BORDER 2: PAPER 2: INK 0: B
RIGHT 0: CLS
30 PRINT AT 7,10: " ZX SPECTRUM "
40 PRINT AT 14,10: INVERSE 1: "
ZX SPECTRUM "
```

FLASH

Acceso al teclado

L LIST

MODOS E



FLASH

Definición

Con la sentencia «FLASH» se controla la característica

de parpadeo de los caracteres. Secuencialmente se van intercambiando los colores de tinta y papel.

Su estructura general es la siguiente:

| SENTENCIA | ARGUMENTO |
|-----------|-------------------|
| FLASH | código de control |

Ejemplos:

- FLASH 1
- PRINT FLASH 1; "HOP"
- FLASH Ø
- INPUT FLASH 1; "? "; a

El atributo por defecto es: parpadeo desactivado. Las sentencias «FLASH» y «BRIGHT» al igual que «PAPER» e «INK», controlan las características de impresión de caracteres completos; es decir, de bloques de 64 pixels (8 * 8).

Introduzca los siguientes comandos directos que harán parpadear a toda la pan-

talla, puesto que se modifican sus atributos.

```
FLASH 1: CLS
```

Veamos otro ejemplo:

```
10 REM *****
PARPADEO
*****
20 BORDER 2: PAPER 2: INK 0: C
L5
30 PRINT AT 7,10: " MICROMANIA "
40 PRINT AT 14,10: FLASH 1: " M
ICROMANIA "
```

«FLASH» también tiene la posibilidad de conservar los atributos de pantalla con el código de control «Ø» (transparencia).

```

10 REM *****
      * FLASH "6" *
*****
20 BORDER 2 PAPER 2 INK 0 F
LASH 0 CLS
30 FOR n=1 TO 10
40 PRINT AT n,0, FLASH 1;"MONI
TOP"
50 NEXT n
60 PRINT #0;" PULSA UNA TECLA
PARA CONTINUAR"
70 PAUSE 0
80 INPUT 0
90 FOR n=1 TO 15
100 PRINT AT n,0, FLASH 6;"IMPR
ESORA"
110 NEXT n

```

Atributos de pantalla

Como se explicó anteriormente, cada posición de carácter tiene unos atributos que determinan el color del papel, de la tinta, si tiene brillo o si parpadea. Para conocer estos atributos, el Spectrum dispone de una función denominada «ATTR».

ATTR

Acceso al teclado

USR

MODO E



ATTR

Definición

«ATTR» retorna los atributos de una posición determinada.

Su estructura general es la siguiente:

| SENTENCIA | ARGUMENTO |
|-----------|------------------|
| ATTR | (línea, columna) |

Ejemplos:

- LET a = ATTR (2, 3)

- PRINT ATTR (0, 5)
- IF ATTR (3, 7) = 10 THEN...
- FOR n = 1 TO ATTR (10, 9)

El valor retornado es un número decimal que debe codificarse en su forma binaria para poder interpretar los atributos.

El significado de cada *bit* es el siguiente:

| BIT | SIGNIFICADO |
|-------|-------------|
| 7 | PARPADEO |
| 6 | BRILLO |
| 3 a 5 | PAPEL |
| 0 a 2 | TINTA |

Con un ejemplo se comprenderá mejor. Supongamos que el valor retornado por la función «ATTR», correspondiente a una posición de la pantalla es «86». Este, codificado en binario es:

01010110

Si tiene alguna duda sobre la notación binaria y las correspondencias entre decimal-binario, consulte la página 30.

Analicemos cada bit:

- El bit «7» (más significativo), situado a la izquierda, el «0» por lo tanto corresponde a:

PARPADEO DESACTIVO

- El valor del bit «6» es «1» que corresponde a:

BRILLO ACTIVO

- Los bits «3» a «5» tienen un valor de «010» este valor codificado en decimal es «2» que corresponde al código de color:

PAPEL ROJO

- Y por último, los bits «0» a «2» son «110» que al codificarlo también en decimal es «6», que corresponde a

TINTA AMARILLA

Compruébelo con las siguientes instrucciones, que seleccionan estos atributos y posteriormente los leen:

```

BRIGH 1
PAPER 2
INK 6
PRINT AT 0, 0, "A"
PRINT ATTR (0, 0)

```

Se comprueba también que sumando las potencias, de base dos, correspondientes a las posiciones donde hay un bit a «1», se obtiene el valor retornado por «ATTR».

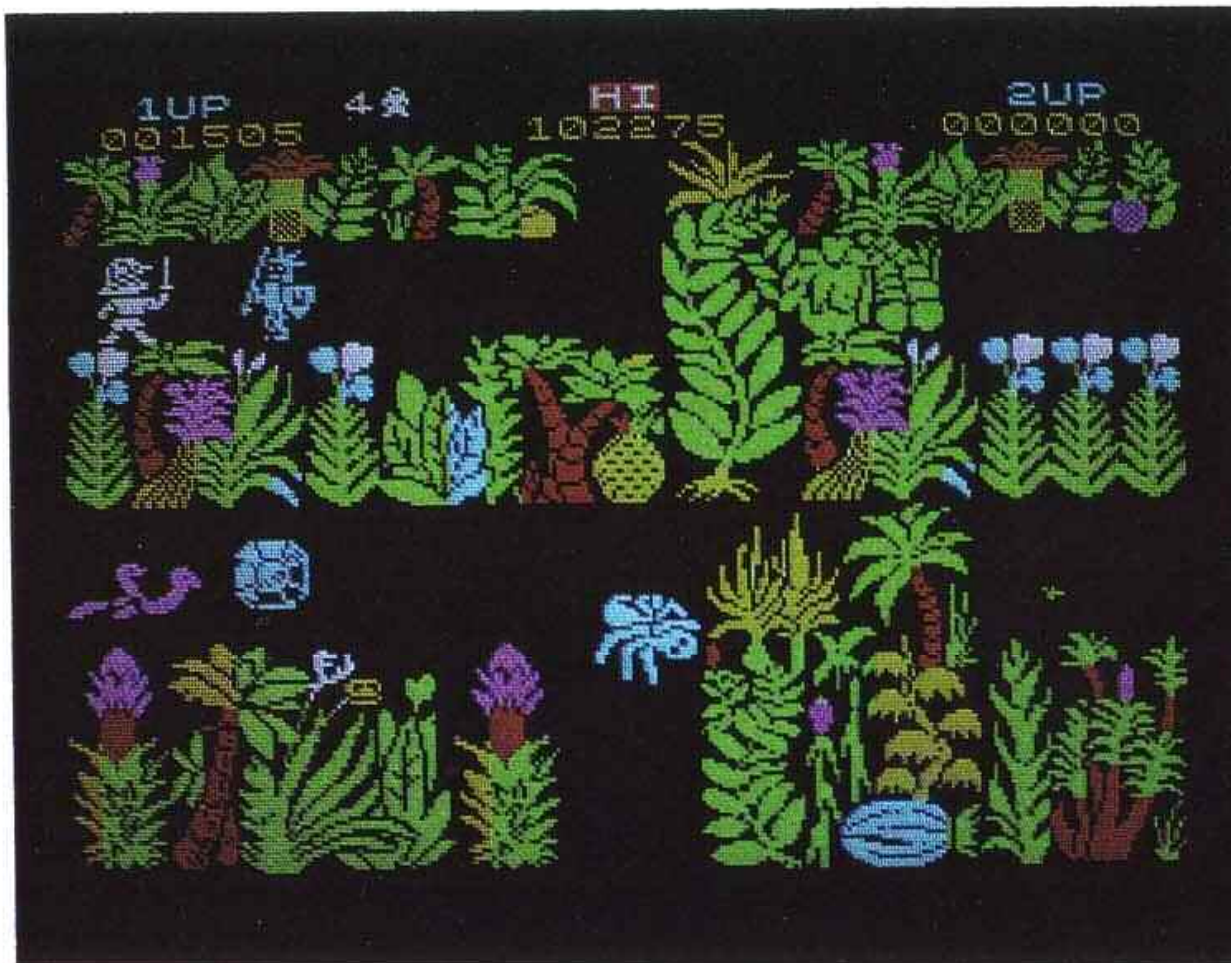
64 + 16 + 4 + 2 = 86

El programa n.º «4» permite visualizar los atributos correspondientes a un código decimal introducido por teclado.

Caracteres de control

En el capítulo dedicado al código ASCII (pág. 37) se comentó que la zona comprendida entre el código 0 y el 31 era la constituida por el código transparente. Dentro de este código hay una serie de caracteres de control que tienen relación con el color, que son:

| | |
|----|-----------|
| 16 | TINTA |
| 17 | PAPEL |
| 18 | PARPADEO |
| 19 | BRILLO |
| 20 | INVERTIDO |



Posibilidades de color en un juego.

Para introducir estos códigos es necesario utilizar la función «CHR\$».

Ejemplo:

```
PRINT CHR$ 16; CHR$ 5; "PEPE"
```

El primer carácter de control corresponde al de tinta y el segundo al código de color; la cadena «PEPE» será visualizada con tinta amarilla (5). Consiguiendo el mismo resultado que:

```
PRINT INK 5; "PEPE"
```

El uso de estos caracteres puede tener utilidad en la asignación de atributos a variables de cadena.

Ejemplo:

```
LET a$ = CHR$ 16 + CHR 0 +
"MICRO"
LET b$ = CHR$ 16 + CHR$ 2 +
"HOBBY"
LET c$ = a$ + b$
PRINT c$
```

Una parte de la variable «c\$» se imprime en tinta negra y la otra en roja.

Se podría haber asignado directamente la totalidad de los caracteres de control a la variable c\$.

Acceso directo

Sin hacer uso de las sentencias «PAPER», «INK»,

«FLASH» o «BRIGHT» se pueden utilizar de forma directa, los colores.

Pase a modo \boxed{E} (extendido) y pulse, por ejemplo, la tecla «1»; ¿qué ocurre?, simplemente que a partir de ese instante escribe con fondo azul; y si vuelve al modo \boxed{E} y pulsa simultáneamente «CAPS SHIFT» y la tecla «7», la tinta será blanca.

En el modo extendido se tiene un acceso directo, con las teclas de la fila superior, a todas las combinaciones de color de papel y tinta y a la activación o desactivación de las características de parpadeo y brillo.

En la figura adjunta se puede identificar la forma de ac-

ceder a cualquiera de estas combinaciones.

Entre las aplicaciones de acceso directo, puede destacar la utilidad que tiene el editar un programa con diversos colores, para resaltar u ocultar alguna de sus zonas.

El acceso directo a las funciones de VIDEO fue revisado en la página 70.

Errores

Si se especifica un argumento erróneo en alguna de estas sentencias, se producen los siguientes errores:

a) Color no válido.

K Invalid colour

Ejemplos:

```
BORDER 8
PAPER 10
INVERSE 2
FLASH 9
```

Ejemplo programa 4.

b) Entero fuera de rango.

B Integer out of range

Ocurre cuando el código de color es inferior a «0» o superior a «255».

Ejemplos:

INK -1
BRIGHT 280

Una instrucción del tipo «BORDER 3.6» no es errónea ya que el argumento queda redondeado a «4», obteniéndose por tanto un borde de color verde. ■

PROGRAMA 4

```
10 REM *****
  * ATRIBUTOS *
  * ***** *
20 BORDER 4: PAPER 4: INK 0: C
LS
25 REM ENTRADA ATRIBUTO
30 INPUT "Atributo > "; LINE a$
  *
  * 40 IF a$="" THEN GO TO 30
  * 50 FOR n=1 TO LEN a$
  * 60 IF a$(n) < "0" OR a$(n) > "9" THEN GO TO 30
  * 70 NEXT n
  * 80 LET atributo=VAL a$
  * 90 IF atributo < 0 OR atributo > 255 THEN GO TO 30
  * 100 PRINT AT 1,0;"CODIGO DECIMAL ... ";atributo
  *
  * 102 REM CALCULO BINARIO
  * 104 LET numero=atributo
  * 106 LET b$=""
  * 110 DIM i(8)
  * 120 FOR n=7 TO 0 STEP -1
  * 130 IF numero>=INT (2^n) THEN LET numero=numero-INT (2^n): LET i(n+1)=1
  * 140 LET b$=b$+STR$ i(n+1)
  * 150 NEXT n
  * 160 PRINT "CODIGO BINARIO ... ";b$
  *
  * 170 REM IDENTIFICACION
  * 180 IF b$(1)="1" THEN PRINT FLASH 1;"PARPADEO ACTIVADO"^^^: GO TO 200
```

```
190 PRINT "PARPADEO DESACTIVADO"^^^
200 IF b$(2)="1" THEN PRINT BRIGHT 1;"BRILLO ACTIVADO"^^^: GO TO 210
210 PRINT "BRILLO DESACTIVADO"^^^
212 LET papel=0
214 LET peso=0
220 FOR n=5 TO 3 STEP -1
230 LET valor=VAL b$(n)
240 IF valor=1 THEN LET papel=papel+2^n*peso
244 LET peso=peso+1
250 NEXT n
250 PRINT "PAPER ";papel; "INK 9;"PAPER ... ";papel
270 LET tinta=0
280 LET peso=0
290 FOR n=8 TO 6 STEP -1
300 LET valor=VAL b$(n)
310 IF valor=1 THEN LET tinta=tinta+2^n*peso
320 LET peso=peso+1
330 NEXT n
340 PRINT "PAPER 9; INK ";tinta;"TINTA ... ";tinta
350 REM CONTINUACION
360 PRINT #0;"Otro atributo (5/362 PAUSE 0
364 LET z$=INKEY$
370 IF z$="n" OR z$="N" THEN STOP
380 IF z$="s" OR z$="S" THEN GO TO 20
390 GO TO 362
```

PROGRAMA DEPURACION

```
9979 STOP
9980 REM DEPURADOR BASIC
9983 LET err sp=(PEEK 23614)*256
+(PEEK 23613)
9984 POKE 23692,1
9985 LET linea=(PEEK (err sp+3))
*256+(PEEK (err sp+2))
9986 LET posicion=PEEK (err sp+4)
)-2
9987 OPEN #2,"K"
9988 PRINT AT 0,0;"Parada en ";l
linea;" ";posicion
9989 PAUSE 0
9990 IF INKEY$="v" OR INKEY$="u"
THEN GO SUB 9993
```

```
9991 CLOSE #2
9992 INPUT 0: RETURN
9993 INPUT AT 0,0;"Variable > ";
LINE v$
9994 IF v$="" THEN INPUT 0: RETU
RN
9995 PRINT AT 0,0;"Contenido ";v
```

```
9996>IF LEN v$=2 THEN IF v$(2)="
$" THEN PRINT "*****";VAL$ v$;"*****";
GO TO 9998
9997 PRINT VAL v$
9998 PAUSE 0
9999 GO TO 9993
```

Capítulo Depuración programas

PROGRAMA 5

```
10 REM *****
* CURSO/BASIC *
* *****
* CARTA COLOR *
* *****
LS 20 BORDER 4: PAPER 4: INK 0: C
30 RESTORE
40 FOR x=0 TO 31 STEP 2
50 READ color
60 FOR y=0 TO 21
70 PRINT PAPER color;AT y,x;"
```

```
80 NEXT y
90 NEXT x
95 REM CONTORNO
100 PLOT 0,0
110 DRAW 0,175
120 DRAW 255,0
130 DRAW 0,-175
140 DRAW -255,0
145 PRINT #0;" Sintoniza correc
tamente la T U"
150 PAUSE 0
160 REM TABLA COLORES
170 DATA 0,1,2,3,4,5,6,7,0,1,2,
3,4,5,6,7
```

PROGRAMA 6

```
10 REM *****
* CURSO/BASIC *
* *****
* COLORES 1 *
*
```

```
* *****
*
20 FOR n=0 TO 6 STEP 2
30 POKE USR "a"+n,85
40 POKE USR "a"+n+1,170
50 NEXT n
60 NEU
```

PROGRAMA 7

```
100 REM *****
* CURSO/BASIC *
* *****
* COLORES 2 *
* *****
LS GO SUB 1000
104 LET a$=CHR$ 144
106 LET brillo=0
110 FOR y=3 TO 17 STEP 2
120 FOR x=8 TO 22 STEP 2
130 PRINT BRIGHT brillo; PAPER
(y-3)/2; INK (x-8)/2;AT y,x;a$;A
T y,x+1;a$;AT y+1,x;a$;AT y+1,x+
1;a$
140 NEXT x
150 NEXT y
160 PRINT #0;"*****F****:Codigo / "
"ENTER****: Continua"
170 PAUSE 0
180 INPUT 0
```

```
182 IF INKEY$="F" THEN GO SUB 1
200
190 IF brillo=0 THEN LET brillo
=1: PRINT AT 0,0;"BRILLO": GO TO
110
200 PRINT AT 0,0;" " " : GO T
O 106
1000 REM NUMEROS
1010 PRINT INVERSE 1;AT 0,8;"
TINTA
1030 FOR x=9 TO 23 STEP 2
1040 PRINT AT 2,x;(x-9)/2
1050 NEXT x
1054 LET b$=" PAPEL "
1056 FOR y=3 TO 18
1058 PRINT INVERSE 1;AT y,5;b$(y
-2)
1072 NEXT y
1090 FOR y=4 TO 18 STEP 2
1100 PRINT AT y,7;(y-4)/2
1120 NEXT y
1130 RETURN
1200 REM PANTALLA
1210 INPUT "Codigo >>> "; LINE c
$
1212 IF LEN c$<>2 THEN GO TO 121
```

```

0
1220 IF VAL c$(1) < 0 OR VAL c$(1)
> 7 THEN GO TO 1210
1230 LET papel=VAL c$(1)
1240 IF VAL c$(2) < 0 OR VAL c$(2)
> 7 THEN GO TO 1210
1250 LET tinta=VAL c$(2)
1252 LET d$=""
1254 FOR n=1 TO 32
1255 LET d$=d$+a$
1256 NEXT n

```

```

1260 FOR n=0 TO 21
1270 PRINT PAPER papel; INK tinta
a; BRIGHT brillo; AT n,0;d$
1280 NEXT n
1290 PRINT #0;" Pulsa una tecla
para continuar"
1300 PAUSE 0
1310 CLS
1312 GO SUB 1000
1320 RETURN

```

GRAFICOS

La posibilidad de realizar gráficos con un ordenador como el Spectrum, es muy valioso para el usuario ya que entre las aplicaciones de éstos, unas veces como parte importante y otras como complemento a los programas, merecen destacarse:

- Posibilidades gráficas en juegos.

- Presentación de gráficos de gestión (histogramas, gráficos de tarta, etc.).

- Utilización en el diseño industrial, esta aplicación es conocida por las siglas CAD-CAM, Computer Aided Design y Computer Aided Manufacturing, que traducido al español viene a significar: Diseño asistido por ordenador y fabricación asistida por ordenador.

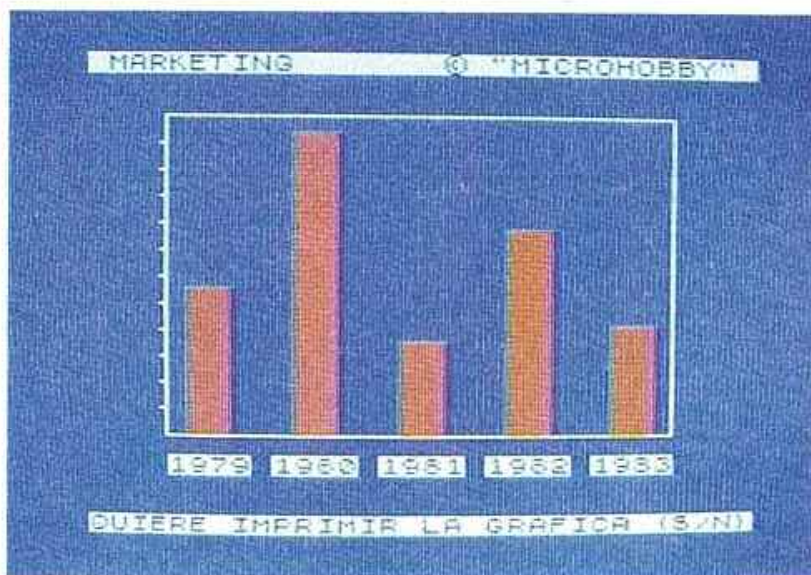
- Matemáticas (Dibujo de Funciones).

Tipos de Gráficos

Atendiendo a la *resolución gráfica* de los dibujos, éstos se pueden clasificar en:

- Gráficos de baja resolución.
- Gráficos de alta resolución.

Para la realización de dibujos en baja resolución. nue-



Gráficos de gestión.

den utilizarse bloques de color o los *gráficos predefinidos*; en cambio para los gráficos de alta resolución el Spectrum dispone de las sentencias:

- PLOT
- DRAW
- CIRCLE

Estas sentencias pueden ser combinadas con «PAPER», «INK», «FLASH», «INVERSE» y «OVER».

Bloques de color

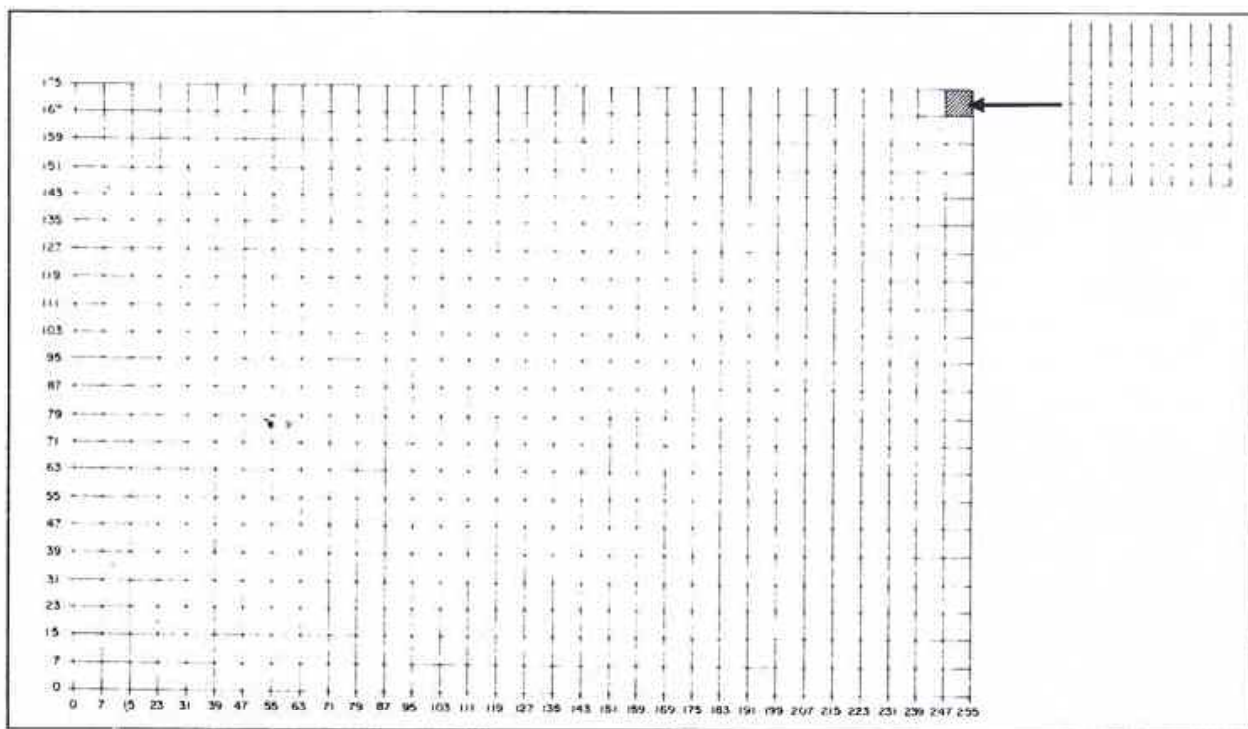
Utilizando el carácter «espacio», con diversos colo-

res de fondo (PAPER), se pueden conseguir gráficos a base de bloques coloreados de 8 por 8 pixel.

El programa número «1» dibuja en pantalla un simpático pez. El número «2» visualiza el nombre de la revista así como las bandas coloreadas características del Spectrum.

Gráficos predefinidos

Para realizar nuestros dibujos en baja resolución, también podemos utilizar los gráficos que se encuentran predefinidos en el juego de caracteres del Spectrum. Pa-



Pantalla en alta resolución.

PROGRAMA 1

```

10 REM *****
* CURSO/BASIC *
*****
* DIBUJO PEZ *
*****

20 BORDER 5: PAPER 5: INK 0: C
LS
30 REM *****
* PRINCIPAL *
*****

32 RESTORE
40 LET color=4
50 GO SUB 1000
60 LET color=6
70 GO SUB 1000
80 LET color=3
90 GO SUB 1000
100 LET color=7
110 GO SUB 1000
120 LET color=0
130 GO SUB 1000
140 LET color=2
150 GO SUB 1000
160 PAUSE 100
170 REM *****
* BURBUJAS *
*****

180 FOR n=1 TO 9
190 READ y,x
200 PRINT PAPER 1,AT y,x;" "
210 PAUSE 25
220 NEXT n
230 PAUSE 0
999 STOP
1000 REM

```

```

*****
* SUB. DIBUJO *
*****

1010 READ y,x
1020 IF NOT y THEN RETURN
1030 PRINT PAPER color,AT y,x;"
"
1040 GO TO 1010
2000 REM *****
* TABLA DE DATOS *
*****

2010 DATA 5,19,5,20,5,21,6,16,6,
17,6,18,6,19,6,20,6,21,7,13,7,14
,7,15,7,16,7,17,7,18,7,19,7,20
2020 DATA 8,12,8,13,8,14,8,15,8,
16,8,17,8,18,8,19,8,20,9,11,9,12
,9,13,9,14,9,15,9,16,9,17,9,18,9
,19,9,20
2030 DATA 10,12,10,13,10,14,10,1
5,10,16,10,17,10,18,10,19,10,20,
11,13,11,14,11,15,11,16,11,17,11
,18,11,19,11,20
2040 DATA 12,12,12,13,12,14,12,1
5,12,16,12,17,12,18,12,19,12,20,
13,11,13,12,13,13,13,14,13,15,13
,16,13,17,13,18,13,19,13,20
2050 DATA 14,13,14,14,14,15,14,1
6,14,17,14,18,14,19,14,20,15,14,
15,15,15,16,15,17,15,18,15,19,15
,20
2060 DATA 16,14,16,15,16,16,16,1
7,16,18,16,19,16,20,16,21,17,16,
17,17,17,18,17,19,17,20,17,21
2070 DATA 18,19,18,20,18,21,13,2
3,13,24,14,24,14,25,15,24,15,25,
15,24,0,0
2080 DATA 5,22,6,22,6,23,6,24,7,
21,7,22,7,23,7,24,8,21,8,22,8,23
,8,24,8,25,9,21,9,25,10,21,10,25
,11,21,11,25
2090 DATA 12,21,13,21,13,22,14,2

```

```


1, 14, 22, 14, 23, 15, 21, 15, 22, 15, 23,
16, 22, 16, 23, 17, 22, 17, 23, 0, 0
2100 DATA 11, 26, 12, 25, 12, 26, 12, 2
7, 13, 25, 14, 26, 14, 27, 11, 12, 11, 11,
11, 10, 11, 9, 10, 11, 10, 10, 9, 10, 8
2110 DATA 12, 11, 12, 10, 12, 9, 12, 8,
12, 7, 9, 9, 8, 9, 7, 9, 6, 13, 9, 13, 8, 1
3, 7, 13, 6, 13, 5, 8, 9, 8, 8, 8, 7, 8, 6, 8,
5, 8, 4
2120 DATA 14, 7, 14, 6, 14, 5, 14, 4, 14
3, 7, 4, 7, 3, 7, 2, 15, 5, 15, 4, 15, 3, 15
2, 15, 1, 0, 0
2130 DATA 9, 22, 9, 23, 9, 24, 10, 22, 1

```

```

0, 24, 11, 22, 11, 24, 12, 22, 12, 23, 12,
24, 0, 0
2140 DATA 10, 23, 11, 23, 0, 0
2150 DATA 2, 19, 3, 18, 3, 19, 3, 20, 4,
17, 4, 18, 4, 19, 4, 20, 4, 21, 5, 15, 5, 16
5, 17, 5, 18, 6, 14, 6, 15
2160 DATA 17, 15, 18, 9, 18, 10, 18, 11
18, 12, 18, 13, 18, 14, 18, 15, 18, 16, 1
8, 17, 18, 18
2170 DATA 19, 14, 19, 15, 19, 16, 19, 1
7, 20, 13, 20, 14, 20, 15, 20, 16, 0, 0
2180 DATA 13, 29, 12, 31, 10, 30, 9, 28
7, 30, 5, 29, 3, 28, 2, 30, 0, 31

```

ra acceder a ellos debe seleccionarse el modo «gráfico» (); en la página 7 de este manual se explica con detalle dicho procedimiento.

El programa número «3» genera una serie de dibujos aleatorios utilizando estos gráficos; el color del «papel» es también aleatorio y la «tinta» tiene atributo de contraste. La visualización de estos gráficos se realiza con la función «CHR\$».

Dentro del código ASCII del Spectrum, los gráficos predefinidos tienen un código comprendido entre el 128 y el 143 en decimal.

El programa número «4», utilizando los gráficos predefinidos, visualiza un dibujo infantil de nuestra redacción.

Pantalla en alta resolución

Cuando se utiliza la pantalla en la modalidad de alta resolución, el eje de *abscisas* (x) se divide en 256 pixel y el de *ordenadas* (y) en 176, esto nos da un total de 45056 pixel.

El origen del eje de *coordenadas* de la pantalla en alta resolución se encuentra en el ángulo inferior izquierdo, a diferencia del de baja resolución que se encuentra en el ángulo superior izquierdo.

PLOT

Acceso al teclado

SIN



ASN

Tipo de sentencia

Comando de dibujo.

Definición

La sentencia «PLOT» visualiza un pixel, determinado por sus coordenadas «x» e «y», del color especificado de «tinta».

La estructura general de la sentencia es:

| SENTENCIA | ARGUMENTO |
|-----------|--------------------|
| PLOT | coord. x, coord. y |

Ejemplos:

- PLOT 100, 100
- PLOT PAPER 4; 20, 30
- PLOT INK 5; 127, 30
- PLOT a, b

El siguiente programa genera una serie de puntos aleatorios, de distinto color, en la pantalla.

```

10 REM *****
   : PUNTOS :
   : *****
20 BORDER 5: PAPER 5: CLS
30 LET X=INT (RND*256)
40 LET Y=INT (RND*176)
50 LET COLOR=INT (RND*8)
60 PLOT INK:COLOR;X,Y
70 GO TO 30

```

Observará, cuando la pantalla tiene cierta cantidad de puntos, que al visualizar uno nuevo cambian de color los de alrededor, esto es debido a que la representación del color se hace en baja resolución, por tanto todos los pixel (64) de un bloque de caracteres deben tener el mismo color.

Con los siguientes programas se pueden dibujar punto a punto las gráficas de las funciones «SENO» y «COSENO»; para que pueda ser visualizado un ciclo completo (360°) ha sido necesario calcular los «puntos» cada dos grados.

a) Función «SENO»:

```

10 REM *****
   : FUNCION SENO :
   : *****
20 DEF FN a(x)=SIN (x*PI/180)
30 FOR n=0 TO 180
40 LET y=FN a(n*2)
50 PLOT n,88+60*y
60 NEXT n

```

b) Función «COSENO»:

```

10 REM *****
   : FUNCION COSENO :
   : *****
12 LET origen=88
14 LET amplitud=50
20 DEF FN a(x)=COS (x*PI/180)
30 FOR n=0 TO 100
40 LET y=FN a(n*2)
50 PLOT n,origen+amplitud*y
60 NEXT n

```

DRAW

Acceso al teclado

COS



ACS

Tipo de sentencia

Comando de dibujo

Definición

Con «DRAW» se pueden dibujar líneas rectas y curvas (arcos). Su estructura general es:

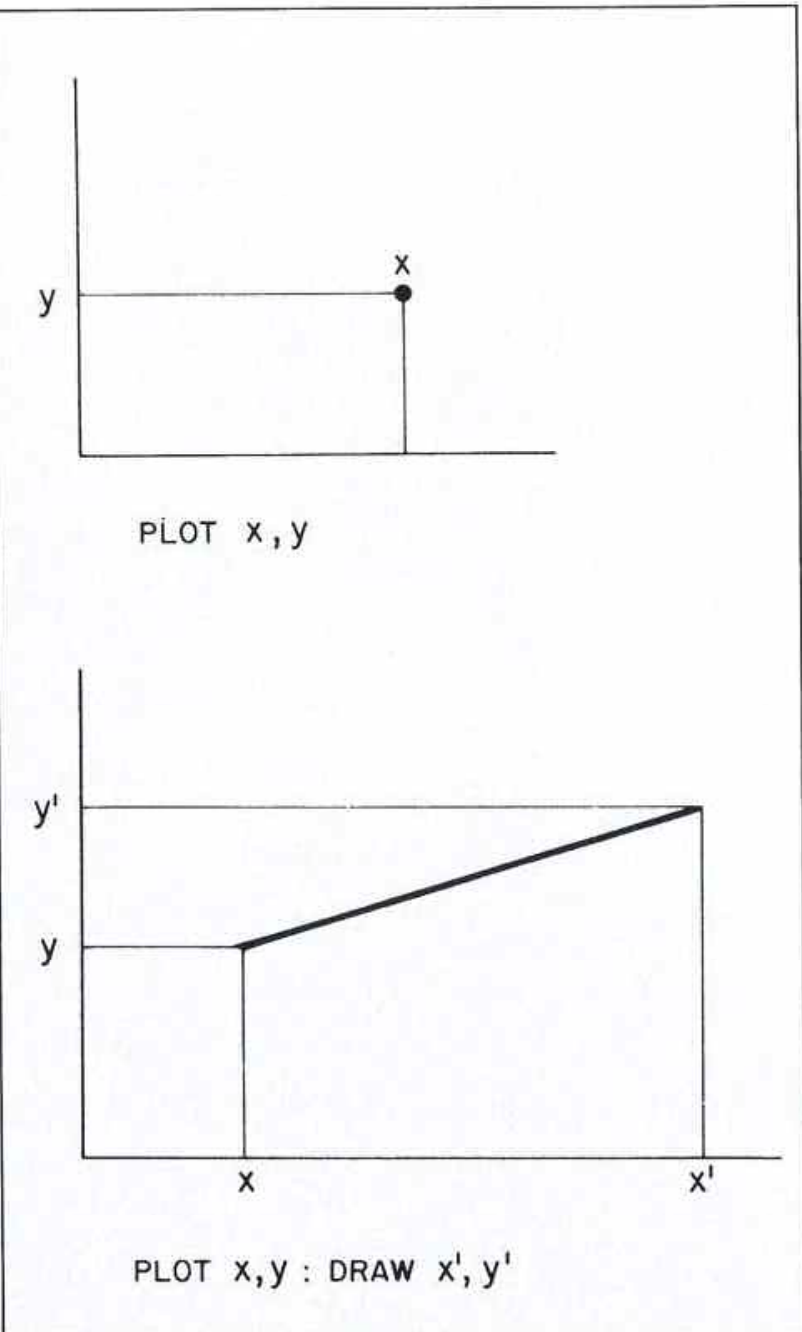
| SENTENCIA | ARGUMENTO |
|-----------|-----------------------|
| DRAW | coord. x, coord. y, z |

El parámetro «Z» es opcional y sirve para dibujar arcos.

Ejemplos:

- DRAW 40, 30
- DRAW INK 8; -20, 10
- DRAW 10, 50, 7
- DRAW 7, 10, -3

El punto de origen de una línea es el último pixel visualizado, bien sea por una sentencia «PLOT», «DRAW» o



Plot y Draw.

«CIRCLE» y el punto de destino es el especificado por las coordenadas (relativas al punto de origen) del argumento de «DRAW»; por ejemplo las sentencias:

```

PLOT 100
DRAW 60, -10

```

visualiza una recta entre los

puntos cuyas coordenadas son: a (100, 100) y b (160, 90).

Las sentencias «RUN», «CLS», «NEW» y «CLEAR» posicionan el origen, por defecto, en las coordenadas 0, 0.

El programa número «5» visualiza una serie de rectas aleatorias de diversos colores; pulsando la tecla «P» (pa-

PROGRAMA 2

```

10 REM *****
   *          *
   *  CURSO/BASIC *
   *          *
   * *****
   *          *
   * MICROHOBBY *
   *          *
   * *****

12 BORDER 1: PAPER 1: INK 0: C
LS
14 REM *****
   *          *
   *  BORDE *
   *          *
   * *****

20 FOR n=0 TO 31
30 PRINT AT 0,n; PAPER 0;" "
40 PRINT AT 21,n; PAPER 0;" "

50 NEXT n
60 FOR n=1 TO 20
70 PRINT AT n,0; PAPER 0;" "
80 PRINT AT n,31; PAPER 0;" "
90 NEXT n
100 REM *****
   *          *
   *  MICRO *
   *          *
   * *****

110 RESTORE
120 READ y,x
130 IF NOT y THEN GO TO 240
140 PRINT AT y,x; PAPER 0;" "
150 GO TO 120
160 REM *****
   *          *
   *  DATOS *
   *          *
   * *****

170 DATA 7,5,6,5,5,5,5,6,5,7,6,
7,7,7,5,8,5,9,6,9,7,9
180 DATA 5,11,6,11,7,11,7,12,7
13,3,11
190 DATA 5,17,5,16,5,15,6,15,7,
15,7,16,7,17
200 DATA 7,19,6,19,5,19,5,20,5,
21
210 DATA 7,23,6,23,5,23,5,24,5,
25,6,25,7,25,7,24
220 DATA 0,0
230 REM *****
   *          *
   *  HOBBY *
   *          *
   * *****

```

```

250 READ y,x
260 IF NOT y THEN GO TO 360
270 PRINT PAPER 2; AT y,x;" "
280 GO TO 250
290 REM *****
   *          *
   *  DATOS *
   *          *
   * *****

300 DATA 15,6,14,6,13,6,12,6,11
,6,13,7,13,8,14,8,15,8
310 DATA 15,10,14,10,13,10,13,1
1,13,12,14,12,15,12,15,11
320 DATA 11,14,12,14,13,14,14,1
4,15,14,15,15,15,16,14,16,13,16,
13,15
330 DATA 11,18,12,18,13,18,14,1
8,15,18,15,19,15,20,14,20,13,20,
13,19
340 DATA 13,22,14,22,15,22,15,2
3,15,24,14,24,13,24,16,24,17,24,
17,23,17,22

350 DATA 0,0
360 REM *****
   *          *
   *  ESPECTRO *
   *          *
   * *****

370 LET posicion=24
380 LET color=2
390 GO SUB 1000
400 LET color=6
410 GO SUB 1000
420 LET color=4
430 GO SUB 1000
440 LET color=5
450 GO SUB 1000
460 PAUSE 0
470 CLS
999 STOP
1000 REM *****
   *          *
   *  SUBROUTINA *
   *          *
   * *****

1010 LET y=y-1
1020 FOR x=posicion TO 30
1030 PRINT PAPER color; AT y,x;" "

1040 LET y=y-1
1050 NEXT x
1055 LET posicion=posicion+1
1060 RETURN

```

rar) dejan de generarse, y con la opción «C» continúa.

Para comprobar la resolución gráfica de las rectas, edite el siguiente programa que «rota» una recta sobre un punto, con un incremento de cinco grados.

```

10 REM *****
   *          *
   *  ROTACION *
   *          *
   * *****

20 BORDER 4: PAPER 4: INK 0: C
LS
30 LET x inicial=128

```

```

40 LET y inicial=68
50 LET longitud=80
60 FOR n=0 TO 360 STEP 5
70 LET radian=n*PI/180
80 LET x=COS radian*longitud
90 LET y=SIN radian*longitud
100 PLOT x inicial,y inicial
110 DRAW x,y
120 NEXT n

```

Observará que la resolución depende de la inclinación de la recta sobre la horizontal.

El siguiente programa genera una serie de cuadrados crecientes a partir de la esquina inferior izquierda.

```

10 REM *****
   *          *
   *  CUADRADOS *
   *          *
   * *****

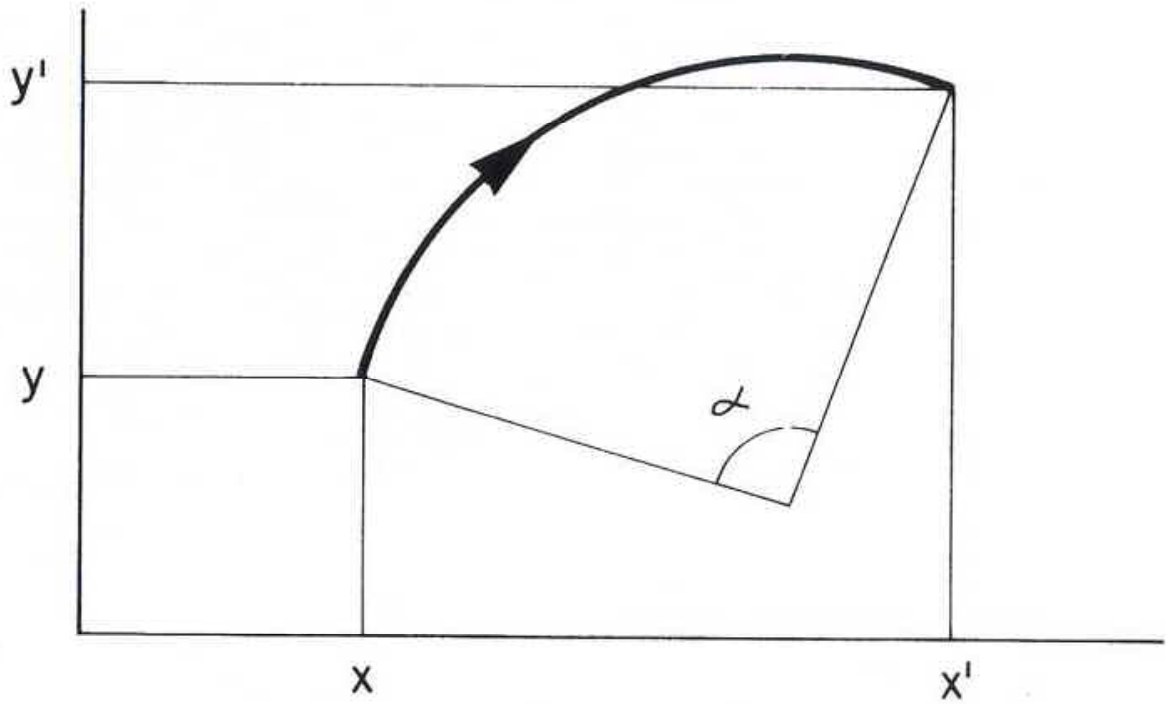
20 BORDER 5: PAPER 5: INK 0: C
LS
30 INPUT "Incremento >>> "; paso

40 FOR n=1 TO 80 STEP paso
50 PLOT n,n
60 DRAW 0,n
70 DRAW n,0
80 DRAW 0,-n
90 DRAW -n,0
100 NEXT n

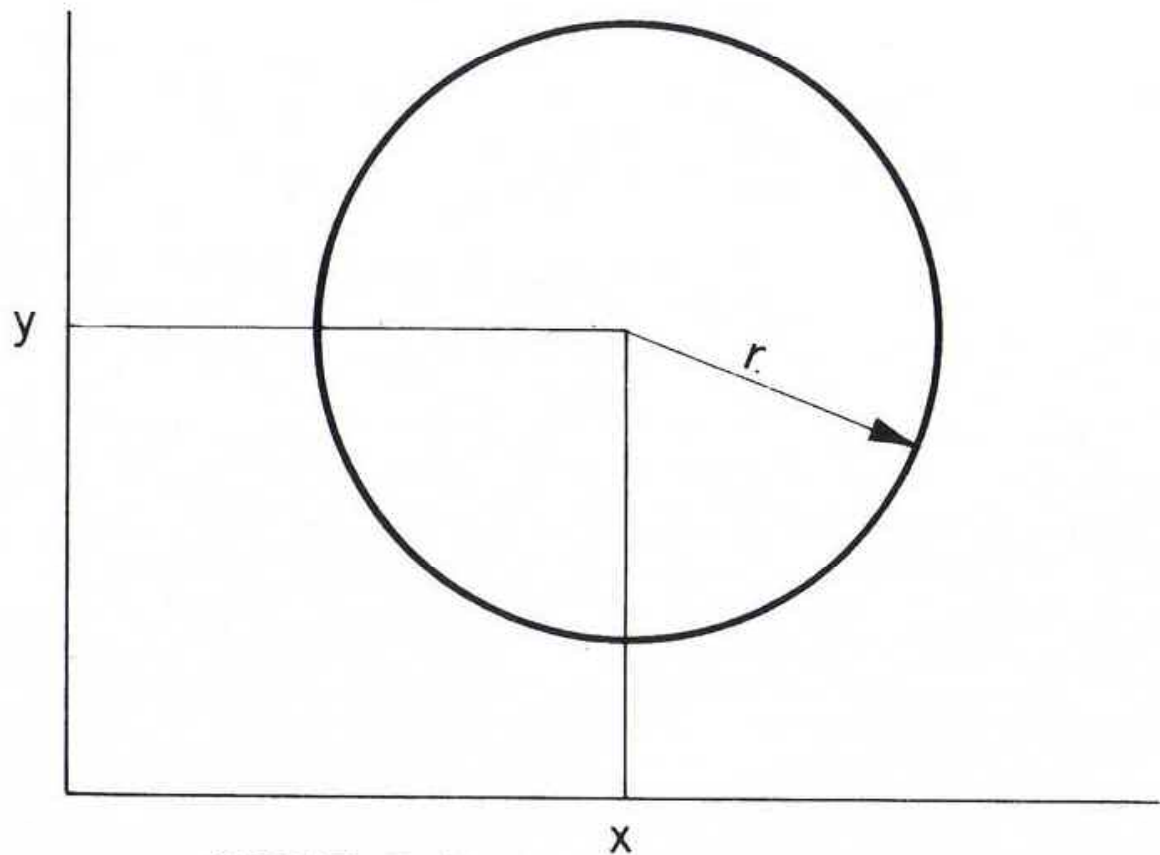
```

La variable «paso» indica la diferencia entre dos cuadrados consecutivos.

Este otro genera unos rectángulos concéntricos.



PLOT x, y : DRAW x', y', α



CIRCLE x, y, r

Draw y Circle.


```

10 REM *****
  * RECTANGULOS *
  *****
20 BORDER 5: PAPER 5: INK 0: C
LS 30 INPUT "Incremento >>> ";pas
0
40 LET base=235
50 LET altura=155
60 FOR n=10 TO 80 STEP paso
70 PLOT n,n
80 DRAW base,0
90 DRAW 0,altura
100 DRAW -base,0
110 DRAW 0,-altura
120 LET base=base-paso*2
130 LET altura=altura-paso*2
140 NEXT n

```

La variable «paso» tiene el mismo significado que en el programa anterior.

Arcos de circunferencia

Como se explicó anteriormente, el parámetro «Z» de una sentencia «DRAW» era opcional y permitía dibujar arcos de circunferencia (líneas curvas). Este parámetro indica el ángulo de giro expresado en radianes.

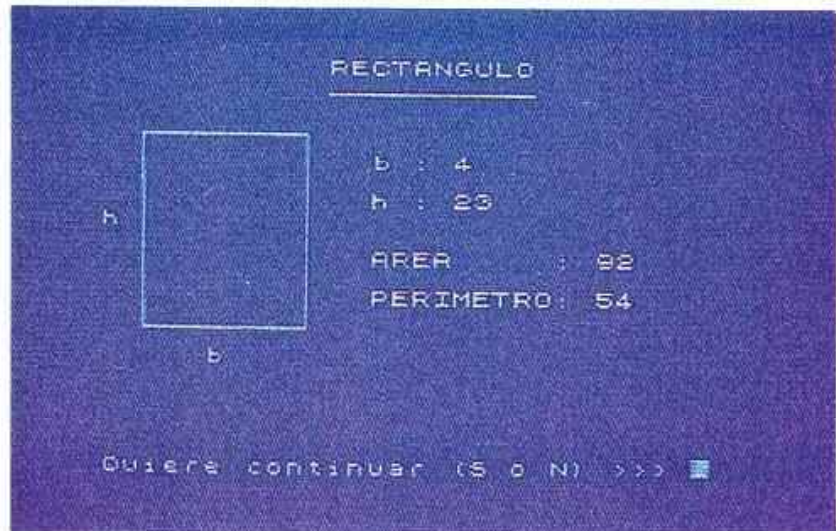
Cuando «Z» es positivo, el giro se realiza en sentido contrario a las agujas del reloj (hacia la izquierda).

Ejemplo:

```

PLOT 30, 30
DRAW 100, 100, 3

```



Aplicaciones en matemáticas.

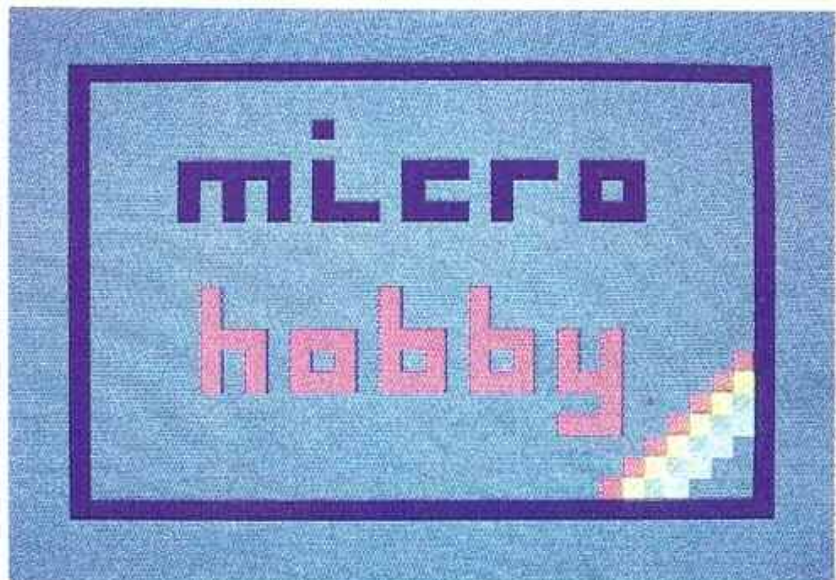


Gráfico de «baja resolución».

Cuando «Z» es negativo, el giro se realiza en el sentido de las agujas del reloj (hacia la derecha).

PROGRAMA 3

```

10 REM *****
  * CURSO/BASIC *
  *****
  * ABSTRACTO *
  *****
LS 12 BORDER 5: PAPER 5: INK 9: C
14 RANDOMIZE
20 FOR y=0 TO 21
30 FOR x=0 TO 31
40 LET caracter=INT (RND*2)

```

```

42 LET grafico=INT (RND*15)+12
9
44 LET color=INT (RND*8)
50 IF caracter=1 THEN PRINT AT
y,x: PAPER color;CHR$ grafico:
GO TO 70
60 PRINT AT y,x;" "
70 NEXT x
80 NEXT y
90 PRINT #0;" Pulsa una tecla
para continuar"
100 PAUSE 0
102 INPUT 0
110 GO TO 20

```

Ejemplo:

```
PLOT 50, 50
DRAW 100, 100, -1
```

El siguiente programa visualiza una esfera con sus husos.

```
10 REM *****
  : ARCOS :
  : ***** :
20 BORDER 3: PAPER 3: INK 0: C
LS
30 FOR n=0 TO 3.4 STEP 0.2
40 PLOT 120,0
50 DRAW 0,175,n
55 PLOT 120,0
60 DRAW 0,175,-n
70 NEXT n
```

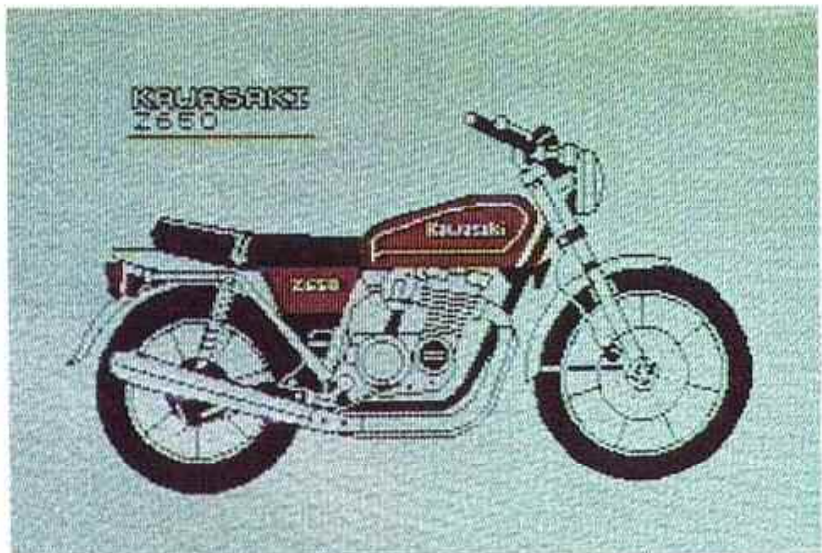


Gráfico de «alta resolución».

Utilizando como parámetro «Z» el valor PI que equivale a 180°, una semicircunferencia, el siguiente programa genera una serie de «ondas»:

```
10 REM *****
  : ONDAS :
  : ***** :
```

```
20 BORDER 4: PAPER 4: INK 0: C
LS
30 FOR x=-160 TO 20 STEP -20
40 PLOT 0,x
50 FOR n=1 TO 5
60 DRAW 25,0,PI
70 DRAW 25,0,-PI
80 NEXT n
90 NEXT x
```

```
10 REM *****
  : CURVAS???:
  : ***** :
12 BORDER 5: PAPER 5: INK 0: C
LS
20 INPUT "Angulo de giro >>>"
:angulo
24 CLS
30 PLOT 127,68
30 DRAW 50,50,angulo
40 GO TO 20
```

Como efecto curioso de la sentencia «DRAW» ejecute el siguiente programa:

Introduzca, por ejemplo, como datos de «ángulo de gi-

PROGRAMA 4

```
10 REM *****
  : CURSO/BASIC *
  : ***** *
  : GRAFICO *
  : ***** *
12 BORDER 5: PAPER 5: CLS
14 REM PAVIMENTO
20 FOR n=19 TO 21
30 PRINT PAPER 2; INK 6; AT n, 0
  : *****
40 NEXT n
50 REM WALLA
52 INK 4
60 PRINT AT 16,0; "TTTT";
70 PRINT "I";
80 PRINT "I";
90 PRINT "TTTT"
100 PRINT AT 17,4; "F";
110 PRINT "I";
120 PAPER 3: INK 1
130 REM EDIFICIO
140 FOR n=17 TO 18
150 PRINT AT n,10; "███ ███"
160 NEXT n
170 FOR n=15 TO 16
180 PRINT AT n,10; "██████"
190 NEXT n
200 FOR n=8 TO 14
210 PRINT AT n,10; "F";
220 PRINT "I";
230 PRINT "I";
```

```
240 PRINT "I";
250 PRINT "I";
260 PRINT "I";
270 PRINT "I";
280 PRINT "I";
290 NEXT n
300 PRINT AT 7,10; "F";
310 PRINT "██████";
320 PRINT "I";
330 INK 2
340 REM CARTEL
350 PRINT AT 6,11; PAPER 5; "I";
  : AT 6,16; "I";
352 PAPER 7
360 PRINT AT 5,8; "L";
370 PRINT "██████";
380 PRINT "I";
390 PRINT AT 4,8; "I"; AT 4,19; "I"
400 PRINT AT 3,8; "██████"
410 PRINT AT 4,9; INK 0; "MICRO"
420 PRINT INK 2; "HOBBY"
430 REM COCHE
440 PAPER 5: INK 3
450 PRINT AT 18,19; "I";
460 PRINT "I";
470 PRINT "I";
480 PRINT AT 17,20; "███"
490 REM AREOL
500 PRINT AT 18,28; INK 6; "I";
510 PRINT AT 17,28; INK 6; "I";
520 INK 4
530 PRINT AT 16,28; "I";
540 PRINT AT 15,28; "I";
550 PRINT "I";
560 PAUSE 0
```


Tipo de sentencia

Comando de dibujo

Definición

La sentencia «CIRCLE» dibuja un círculo. Su estructura es:

| SENTENCIA | ARGUMENTO |
|-----------|---------------------------|
| CIRCLE | coord. x, coord. y, radio |

Las coordenadas «x» e «y» corresponden a su centro.

Ejemplos:

- CIRCLE 128, 88, 30
- CIRCLE FLASH 1; 100, 100, 50
- CIRCLE a, b, c
- CIRCLE INK 3; 100, 100, 30

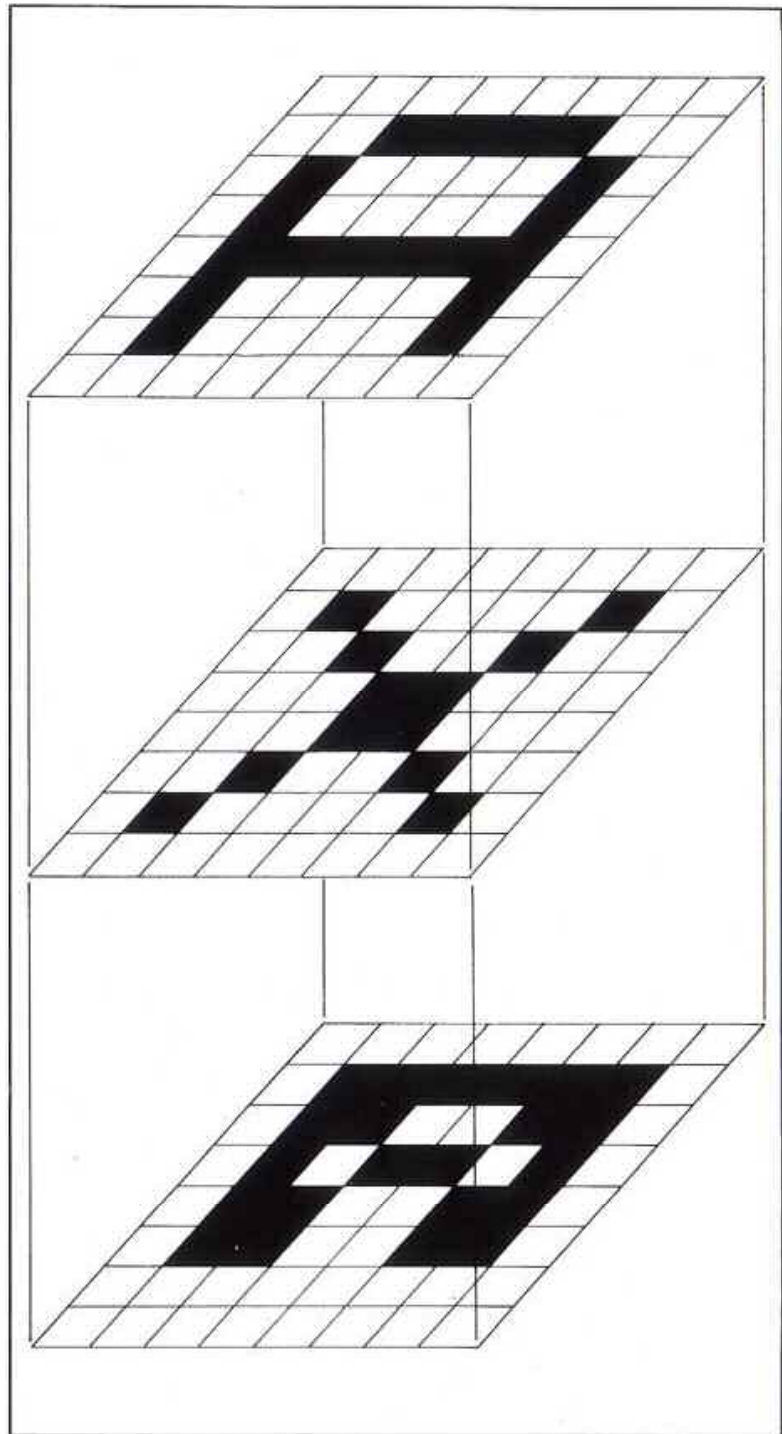
El siguiente programa presenta la diferencia entre un círculo dibujado a base de puntos y el visualizado con «CIRCLE».

```
10 REM *****
   * CIRCULO *
   * *****
12 BORDER 4: PAPER 4: INK 0: C
LS
20 LET x inicial=60
30 LET y inicial=88
40 LET radio=50
50 FOR n=1 TO 360
60 LET radian=n*PI/180
70 LET x=CO5 radian*radio
80 LET y=SIN radian*radio
90 PLOT (x inicial+x), (y inici
al+y)
100 NEXT n
110 CIRCLE 195,88,50
```

Observe las diferencias en precisión y velocidad.

Este otro genera una serie de circunferencias aleatorias; al superponerse cambian de color los bloques de caracteres afectados.

```
10 REM *****
   * REDONDELES *
   * *****
20 BORDER 1: PAPER 5: CLS
22 RANDOMIZE
```



Over 1

```
30 LET x=INT (RND*216)+20
40 LET y=INT (RND*136)+20
50 LET radio=INT (RND*16)+5
60 LET color=INT (RND*8)
62 IF color=5 THEN GO TO 60
70 CIRCLE INK color;x,y,radio
80 GO TO 30
```

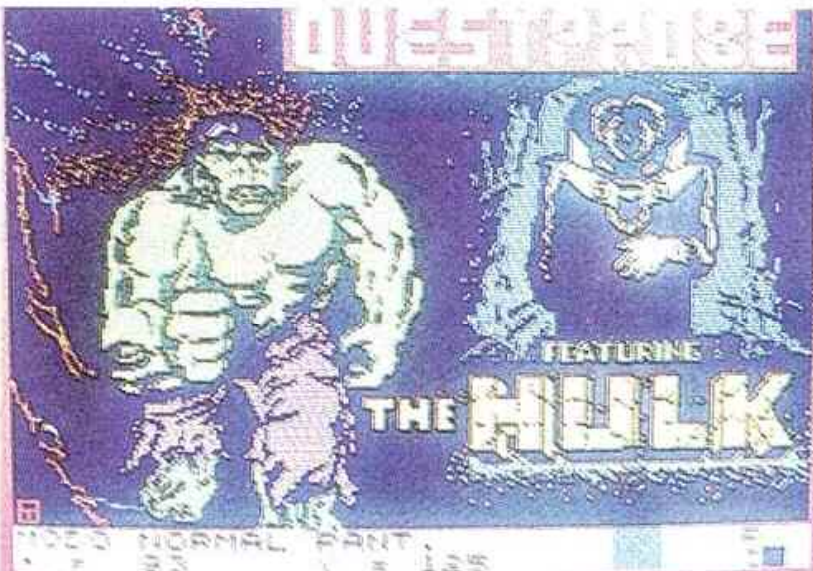
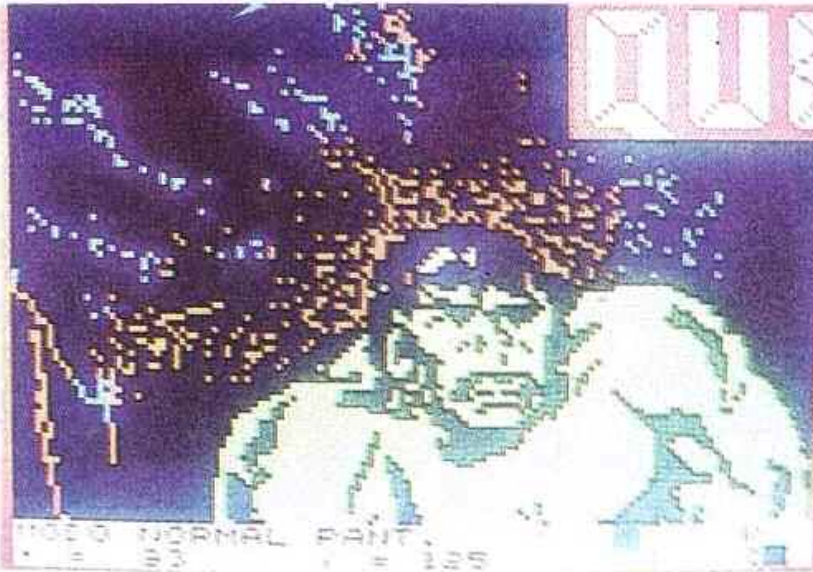
existen otros métodos para realizar gráficos; éstos se basan principalmente en la utilización de:

Técnicas avanzadas

Aparte de las sentencias «PLOT», «DRAW» y «CIRCLE»,

- Programas
- Periféricos

Los creadores de pantallas



Técnicas avanzadas en la realización de gráficos.

gráficas utilizan una serie de programas comerciales, que facilitan su labor a la hora de diseñarlas. Entre los más conocidos están:

- MELBOURNE DRAW
- PAINTBOX
- SCREEN MACHINE
- ARTIST

Lógicamente, las características de cada uno de estos programas son distintas de las del resto, pero, generalizando, se puede:

- Desplazar el cursor en las ocho direcciones, pixel a pixel, sin dibujar.
- Idem dibujando.
- Idem dibujando en inversa.
- Idem borrando.
- Cambiar los atributos de cada bloque de 64 pixel.
- Rellenar áreas de un color determinado.
- Desplazar un gráfico hasta situarlo en la posición adecuada.
- Invertir la pantalla de izquierda a derecha.

- Definir GDU (Gráfico Definido por el Usuario), almacenarlos en cinta y utilizarlos posteriormente.

- Incluir textos.
- Algunos permiten ampliar una zona de la pantalla (zoom), gracias a esta opción se puede dibujar con una mayor precisión.

Existen comercialmente ciertos *periféricos*, o dispositivos que se conectan al ordenador, con los que también se pueden realizar gráficos; normalmente necesitan de un software para ser utilizados.

Principalmente existen cuatro tipos:

- LAPIZ OPTICO
- MESA GRAFICA
- PANTALLA DE VIDEO TACTIL
- RATON

El *Lápiz Óptico* (Light Pen) es un dispositivo que al desplazarlo por la pantalla como si estuviera escribiendo, detecta la luz emitida por el televisor y la transforma en señales eléctricas, las cuales son enviadas al ordenador para ser procesadas. Una vez conocidas las coordenadas del punto donde está posicionado el lápiz, el programa se encarga de visualizar el correspondiente pixel.

La *Mesa Gráfica* es una especie de tablero donde se acopla un *brazo articulado*, conectado mecánicamente con unos componentes conocidos por el nombre de *potenciómetros*. Este tipo de *traductor* envía una información al ordenador, que depende de la posición del brazo.

Para conocer la posición de un punto se utiliza un sistema de *coordenadas polares*.

Existen también otro tipo de mesas más sofisticadas, don-

PROGRAMA 6

```

1 REM *****
  * CURSOR/BASIC *
  *
  * MAPA *
  *
  *****

2 BORDER 4: PAPER 4: INK 1: C
LS
4 RESTORE
10 READ X,Y
15 IF X=0 THEN GO TO 40
20 PLOT X,Y
30 GO TO 10
40 INK 0
45 PLOT 159,7
50 DRAW 96,0
60 DRAW 0,40
70 DRAW -96,0
80 DRAW 0,-40
90 PLOT 200,7
100 DRAW 24,40
110 READ X,Y
120 IF X=0 THEN GO TO 150
130 PLOT X,Y
140 GO TO 110
150 PAUSE 0: STOP
5000 REM
      *****
      * PENINSULA *
      *
      *****

5010 DATA 133,175,134,174,134,17
3,134,172,133,171,133,170,133,16
9,132,168
5020 DATA 131,167,130,166,129,16
5,128,164
5030 DATA 127,165,126,164,125,16
3,124,163,123,162,122,163,121,16
4,120,163
5040 DATA 119,163,118,164,117,16
4,116,165,115,164,114,165,113,16
6,112,165
5050 DATA 111,165,110,164,109,16
3,108,164,107,164,106,165,105,16
5,104,165
5060 DATA 103,165,102,166,101,16
6,100,166,99,165,98,166,97,167,9
6,167
5070 DATA 95,166,94,165,93,165,9
2,165,91,165,90,166,89,165,88,16
5
5080 DATA 87,165,86,164,85,165,8
4,165,83,165,82,166,81,166,80,16
7
5090 DATA 79,167,78,168,77,167,7
6,168,75,168,74,169,73,168,72,16
9
5100 DATA 71,169,70,169,69,169,6
8,170,67,171,66,170,65,171,64,17
0,64,169
5110 DATA 63,168,62,169,61,169,6
0,170,59,169,58,169,57,170,56,17
0
5120 DATA 55,169,54,169,53,168,5
3,167,52,166,51,167,51,168,51,16
9,50,170,49,170,48,169
5130 DATA 47,169,46,170,45,170,4
4,171,43,172,43,173,42,173,41,17
4,40,174,40,173
5140 DATA 39,172,38,173,38,174,3
8,175,37,174,36,173,35,173,34,17
4,35,175,33,173,32,172
5150 DATA 31,173,30,172,29,171,2
8,170,29,169,30,169,31,169,31,16
8
5160 DATA 31,167,30,166,29,165,2
8,166,28,167,27,166,26,166,26,16
7,25,166,24,166
5170 DATA 23,166,22,167,21,166,2
0,167,19,167,18,166,17,166,18,16
5,17,164,16,164
5180 DATA 15,164,14,165,14,163,1
3,163,12,162,13,161,13,160
5190 DATA 12,159,12,158,14,159,1
4,158,14,157,14,156,14,155,15,15
6,15,152

```

```

5200 DATA 16,156,17,155,16,154,1
6,153,16,152,17,152,18,153,19,15
3,20,152
5210 DATA 19,151,19,150,18,149,1
8,148,19,148,20,147,18,148,19,14
6,20,146,21,146,22,146,19,145,22
,145,21,144,15,149
5220 DATA 20,143,19,143,16,142,1
7,141,17,140,17,139,17,138,16,13
7,17,136
5230 DATA 1,135,17,134,17,133,1
8,132,17,131,17,130,18,129,17,12
8
5240 DATA 18,127,18,126,18,125,1
8,124,18,123,18,122,19,121,19,12
0
5250 DATA 19,119,19,118,18,117,1
8,116,18,115,18,114,17,113,19,11
3,19,112
5260 DATA 19,111,19,110,18,109,1
7,108,16,108,16,109,16,110
5270 DATA 15,107,15,106,15,105,1
4,104
5280 DATA 14,103,13,102,13,101,1
4,101,15,101,15,100,15,99,14,99,
13,98,12,97,12,96
5290 DATA 12,95,11,94,11,93,11,9
2,10,91,10,90,9,89,9,88
5300 DATA 8,87,8,86,8,85,7,84,6,
84,5,83,5,82,5,81,5,80
5310 DATA 4,79,4,78,3,77,3,76,3,
75,3,74,3,73,2,72
5320 DATA 2,71,2,70,3,70,4,70,5,
71,6,71,7,71,8,72,9,72,10,73,11,
72
5330 DATA 11,71,10,70,9,70,8,69,
7,68,6,69,5,68,5,67,6,66,6,65,6,
64,7,64
5340 DATA 8,64,9,64,10,65,11,66,
12,66,12,65,13,65,14,66,15,65,15
,64
5350 DATA 16,63,15,62,14,62,13,6
3,11,63,12,62,12,61,12,60,12,59,
12,58,12,57,11,56
5360 DATA 11,55,10,54,10,53,11,5
2,11,51,11,50,11,49,11,48
5370 DATA 11,47,11,46,11,45,11,4
4,10,43,10,42,10,41,9,40
5380 DATA 9,39,9,38,8,37,8,36,7,
35,7,34,8,34,9,34,10,35,11,35,12
,35,13,35,14,35,15,35
5390 DATA 16,34,17,34,18,34,19,3
4,20,34,21,34,22,33,23,33
5400 DATA 24,32,25,31,26,32,27,3
2,28,33,29,33,30,34,31,34
5410 DATA 32,35,33,35,34,35,35,3
5,36,36,37,36,38,36,39,36
5420 DATA 40,35,41,35,42,35,43,3
5,44,34,45,33,46,32
5430 DATA 47,31,48,30,49,29,50,2
9,51,28,51,27,51,26,50,25,50,24
5440 DATA 51,23,52,23,53,23,54,2
2,55,21,55,20,54,19,53,19,52,19,
52,20,53,18,54,17,54,16
5450 DATA 55,15,56,15,57,14,58,1
3,59,12,60,11,61,11,62,10,63,9
5460 DATA 64,9,65,9,66,10,66,11,
67,12,68,12,68,11,69,13,69,14,70
,15
5470 DATA 71,16,71,17,72,18,73,1
8,74,18,75,18,76,19,77,18,78,18,
79,18
5480 DATA 80,19,81,20,82,21,83,2
2,84,22,85,23,86,23,87,22
5490 DATA 88,22,89,23,90,23,91,2
3,92,23,93,24,94,24,95,23
5500 DATA 96,23,97,24,98,24,99,2
4,100,23,101,23,102,23,103,22
5510 DATA 104,23,105,23,106,23,1
07,23,108,24,109,23,110,23,111,2
2
5520 DATA 112,23,113,23,114,23,1
15,23,116,24,117,25,118,25,119,2
5
5530 DATA 120,25,121,24,122,24,1
23,25,124,25,125,25,126,24,126,2
5,126,26,127,27,127,26
5540 DATA 128,28,129,29,129,30,1
29,31,129,32,129,33,130,34,130,3
5,131,36,132,37,133,37,134,37,13
5,38,135,39
5550 DATA 136,40,137,41,138,40,1

```

```

39,40,140,41,141,41,142,40,143,4
0
5560 DATA 144,41,145,41,146,41,1
47,41,148,42,149,42,148,43,148,4
4,148,45,147,43,146,43,145,44,14
5,45,146,46,146,47
5570 DATA 147,46,147,49,148,50,1
48,51,148,52,149,53,149,54,149,5
5
5580 DATA 150,56,151,57,151,58,1
52,59,153,60,154,61,155,61,156,6
2,157,62,158,63
5590 DATA 159,64,160,65,161,65,1
62,66,162,67,161,68,160,68,159,6
9,158,69,157,70,156,70,155,71
5600 DATA 155,72,154,73,154,74,1
54,75,154,76,153,77,153,78,153,7
9
5610 DATA 153,80,153,81,153,82,1
53,83,154,84,154,85,154,86,155,8
7
5620 DATA 155,88,156,89,157,90,1
58,91,158,92,158,93,159,94
5630 DATA 160,95,160,96,161,97,1
62,98,162,99,163,100,163,101,164
,102,165,103
5640 DATA 165,104,166,105,166,10
6,167,107,167,108
5650 DATA 168,108,169,107,169,10
6,168,105,170,108,171,109,170,11
0,169,111
5660 DATA 170,112,171,113,171,11
4,172,115,173,116,174,116,175,11
7
5670 DATA 176,117,177,117,178,11
8,179,118,180,118,181,119,182,11
9,183,119
5680 DATA 184,119,185,120,186,12
0,187,120,188,121,189,121,190,12
1,191,122,191,123
5690 DATA 192,124,193,125,194,12
6,195,127,196,127
5700 DATA 197,128,198,129,199,12
9,200,129,201,130,202,131,203,13
1,204,132,205,133,206,134,207,13
5
5710 DATA 207,136,207,137,207,13
8,207,139,206,140,206,141,206,14
2,207,143
5720 DATA 208,144,208,145,208,14
6,207,146,206,147,205,148,205,14
9,204,150,204,151
5730 DATA 204,152,204,153,204,15
4,204,155,203,156,204,157,203,15
8,203,159
5740 DATA 203,160,204,161,205,16
2,205,163,206,164,207,165
5750 DATA 208,166,209,166,210,16
7,211,168,212,169,213,169,214,17
0,215,171
5760 DATA 216,172,217,173,218,17
3,219,173,220,172,221,171,222,17
1,223,170
5770 DATA 224,171,225,171,224,17
2,225,173,226,173,227,172,228,17
1,227,170,226,169
5780 DATA 225,168,226,167,227,16
7,228,168,229,168,230,167,231,16
7,232,168
5790 DATA 231,169,230,170,231,17
1,232,171,233,170,234,169,235,16
9,236,170,237,170,238,169,239,16
8
5800 DATA 240,167,241,167,242,16
8,243,167,244,166,244,165,245,16
5,246,164
5810 DATA 247,164,246,164,249,16
5,250,164,251,163,252,163,253,16
2,254,162,255,162
5820 REM
*****
*
* BALEARES *
*
*****
5830 DATA 215,88,214,89,213,90,2
12,89,211,88,210,89,209,90,210,9
1,209,92,210,93,210,94,209,94,20
8,93
5840 DATA 207,93,206,92,205,92,2

```

```

04,91,203,90,202,90,201,89,200,8
8
5850 DATA 199,87,199,86,198,85,1
97,84,199,84
5860 DATA 200,83,201,84,202,84,2
03,84,204,83,204,82,204,81,205,8
1,206,81,207,80
5870 DATA 206,79,209,79,209,77,2
07,76,210,80,211,81,212,82,213,8
3,213,84,213,85,214,86,215,87
5880 DATA 183,72,182,72,181,72,1
81,71,180,70,179,70,180,69,180,6
8,181,69,182,68,183,68,184,68,18
3,69
5890 DATA 184,70,185,71,186,72,1
85,73,184,73
5900 DATA 184,65,183,65,182,64,1
83,63,183,64,184,64,185,64,186,6
4
5910 DATA 224,96,223,97,222,96,2
21,96,220,96,219,95,220,94,220,9
3,221,94,222,94,223,94
5920 DATA 224,93,225,93,226,92,2
27,92,228,93,227,94,226,95,225,9
6
5930 REM
*****
*
* CANARIAS *
*
*****
5940 DATA 200,24,201,25,202,26,2
03,27,203,28,203,29,202,29,201,2
9,200,28
5950 DATA 199,27,198,27,197,26,1
96,26,195,26,194,26,193,26,192,2
5
5960 DATA 191,24,190,24,189,24,1
88,24,188,23,189,22,190,21,191,2
0
5970 DATA 192,19,193,18,193,17,1
94,16,195,16,196,17,197,18,197,1
9,198,20,198,21,199,22,199,23
5980 DATA 184,17,183,17,182,17,1
81,18,181,19,181,20,182,21,183,2
1,184,20,185,20,186,19,185,18
5990 DATA 169,32,170,31,170,30,1
71,29,171,28,172,27,173,28,174,2
9,174,30,174,31,174,32,174,33,17
3,34,172,34,171,35,170,34,169,34
,169,33
6000 DATA 169,10,168,10,168,11,1
69,12,170,12,171,12,172,13,173,1
2,173,11,172,10,171,9,170,10
6010 DATA 216,12,215,12,214,11,2
13,11,212,11,211,12,210,13,210,1
4,210,15
6020 DATA 211,16,211,17,212,16,2
12,19,213,20,214,20,215,19
6030 DATA 216,19,217,20,217,16,2
18,17,218,16,218,15,218,14,217,1
3
6040 DATA 240,24,240,25,240,26,2
41,27,241,28,241,29,242,30,243,3
0,244,30,246,31,245,29,245,28,24
5,27,245,26,244,25,244,24
6050 DATA 244,23,244,22,243,21,2
42,21,241,20,240,20
6060 DATA 239,19,238,18,237,17,2
36,16,235,16,234,17,234,18,235,1
8,236,19,237,20,238,21,239,22,23
9,23
6070 DATA 240,40,241,40,241,44,2
42,40,243,40,243,43,243,45,244,4
1
6080 DATA 244,39,244,38,243,37,2
42,37,241,35,240,36
6090 DATA 239,36,238,35,237,36,2
38,37,238,38,239,39,0,0
6095 REM
*****
*
* ESPANA *
*
*****
6100 DATA 1,8,2,8,3,8,4,8,5,8,6,

```

```

8,6,9,2,9,2,10,2,11,2,12,2,13,2,
14,1,14,3,14,4,14,5,14,6,14,6,13
3,11,4,11,4,12,4,10
6110 DATA 9,9,9,8,10,8,11,8,12,8
,13,8,13,9,13,10,13,11,12,11,11,
11,10,11,9,11,9,12,9,13,9,14,10,
14,11,14,12,14,13,14,13,13
6120 DATA 16,8,17,8,18,8,17,9,17
,10,17,11,17,12,17,13,17,14,16,1
4,16,14,19,14,20,14,21,14,21,13,
21,12,21,11,20,11,19,11,18,11
6130 DATA 23,8,24,8,25,8,24,9,24
,10,24,11,24,12,24,13,24,14,25,1

```

```

4,26,14,27,14,28,14,28,13,28,12,
26,11,28,10,26,9,28,8,27,8,29,8,
25,11,26,11,27,11
6140 DATA 31,8,32,8,33,8,32,9,32
,10,32,11,32,12,32,13,32,14,31,1
4,33,14,35,8,36,8,37,8,36,9,36,1
0,36,11,36,12,36,13,36,14,35,14,
37,14,33,12,34,11,35,10,33,16,34
,16,35,16
6150 DATA 39,8,40,8,41,8,40,9,40
,10,40,11,40,12,40,13,40,14,41,1
4,42,14,43,14,44,14,44,13,44,12,
44,11,44,10,44,9,44,8,43,8,45,8,
41,11,42,11,43,11,0,0

```

de el traductor ha sido sustituido por una especie de lápiz y la mesa incorpora un complejo sistema de detección, bien sea por campos eléctricos, magnéticos o ultrasónicos.

Otro tipo de periférico es la pantalla de video sensible al tacto; con ella se puede dibujar directamente con el dedo, posicionándolo en los lugares elegidos.

El sistema de detección, de la zona de la pantalla tocada con el dedo, está basado en la

interrupción de rayos infrarrojos.

Por último, el *ratón* es un pequeño mando que según se va desplazando sobre la mesa, genera un gráfico idéntico al movimiento de éste.

Es un periférico muy fácil de utilizar y además, existe una perfecta coordinación entre el movimiento de la mano sobre la mesa y la observación de la pantalla por parte del usuario.

OVER

Acceso al teclado

IN KEY \$



MODO E

SYMBOL SHIFT

OVER

Tipo de sentencia

Control de impresión.

PROGRAMA 7

```

1 REM *****
* CURSO/BASIC *
* *****
* OVER *
* *****
2 BORDER 1: PAPER 4: INK 0: C
LS
8 FOR z=1 TO 2
10 OVER 1
20 LET paso=5
30 RANDOMIZE
40 LET x=INT (RND*256)
50 LET y=INT (RND*176)
80 LET xi=-x
90 LET xf=255-x
92 LET yf=175-y
100 FOR n=xi TO xf STEP paso
102 PLOT x,y
110 DRAW n,yf
120 NEXT n
130 LET xf=255-x
140 LET yi=yf
150 LET yf=-y
160 FOR n=yi-paso TO yf STEP -p
aso
170 PLOT x,y

```

```

180 DRAW xf,n
190 NEXT n
200 LET xi=xf
210 LET xf=-x
220 FOR n=xi-paso TO xf STEP -p
aso
230 PLOT x,y
240 DRAW n,yf
250 NEXT n
260 LET yi=yf
270 LET yf=175-y
290 FOR n=yi+paso TO yf-paso ST
FP paso
300 PLOT x,y
310 DRAW xf,n
320 NEXT n
322 PRINT #0;" Pulsa una tecla
para continuar"
324 PAUSE 0: INPUT 0
330 NEXT z
340 FOR z=1 TO 7
350 FOR n=0 TO 21
360 PRINT PAPER z; INK 9;AT n,0
;"
370 NEXT n
380 PAUSE 200
390 NEXT z
400 PAUSE 0: OVER 0

```


Definición

La sentencia «OVER» controla la *sobreimpresión* de los caracteres. Su estructura general es:

| SENTENCIA | ARGUMENTO |
|-----------|-------------------|
| OVER | código de control |

Ejemplos:

- PRINT OVER 1; «BASIC»
- OVER 0
- OVER c
- PRINT OVER n; 1

Cuando el código de control es «cero», al visualizar un carácter en una posición de la pantalla, se borra el que anteriormente hubiese en esa misma posición. Al conectar el ordenador al código de control por defecto es el «0».

Ejemplo:

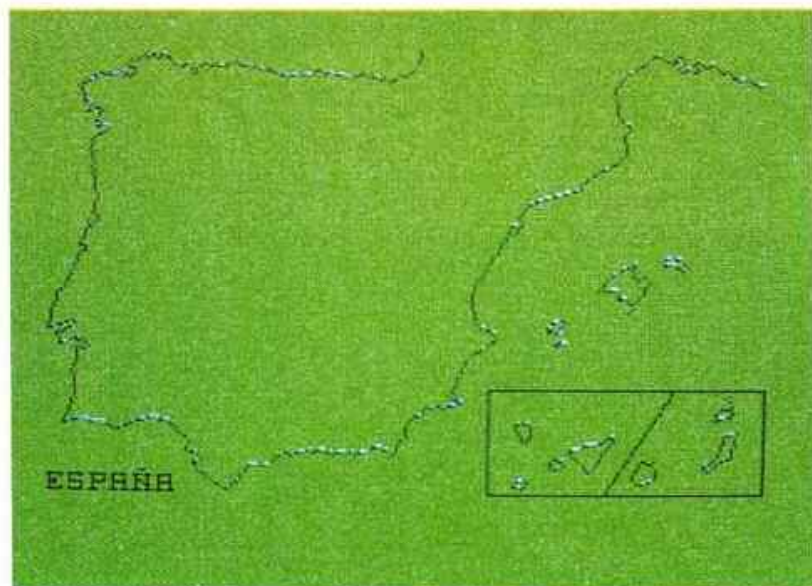
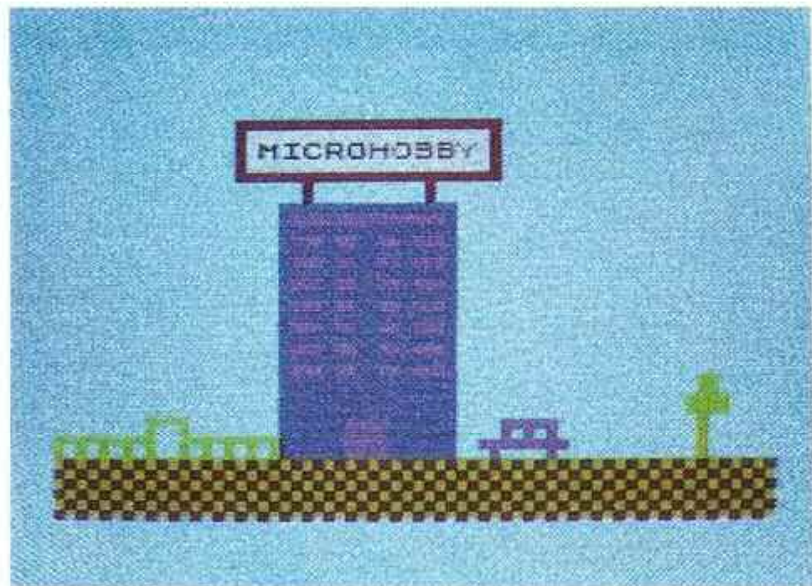
```

10 REM *****
      * OVER 0 *
      * *****
12 OVER 0
20 PRINT AT 10,10,"MICROHOBBY"
30 PRINT #0;" Pulsa una tecla
para continuar"
40 PAUSE 0
50 PRINT AT 10,11;"I"
60 PRINT AT 10,17;"Bb"
  
```

Con el código de control «uno», se combinan uno a uno, los pixel del carácter antiguo y del nuevo, de acuerdo con la función lógica «XOR» (exclusive OR), cuya *tabla de verdad* es la siguiente:

| a | b | XOR |
|---|---|-----|
| 0 | 0 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 0 |

Es decir, si los dos pixel que se combinan tienen color de «papel» o de «tinta» el pixel resultante tendrá color de «papel»; por el contrario, si son distintos, es decir, si uno tiene color de papel y otro de tinta,



el pixel resultante tendrá color de «tinta».

Ejemplo:

```

10 REM *****
      * A OVER X *
      * *****
20 PRINT AT 0,0;" A OVER X = "
30 PRINT AT 0,10;"A"
40 PRINT OVER 1;AT 0,12;"X"
  
```

Si el código de control es distinto de los mencionados anteriormente nos aparecerá

el error de «Invalid colour» o «Interger out of range».

El siguiente programa es una aplicación curiosa de la sentencia «OVER».

```

10 REM *****
      * "OVER" *
      * *****
20 REM © MICROHOBBY.
30 LIST : OVER 1
40 PRINT #0; INVERSE 1;" Pulsa
una tecla para continuar " : PAU
SE 0 : INPUT 0
50 FOR n=1 TO 3
60 PRINT BRIGHT 1;AT n,10;"
70 NEXT n
80 PRINT INVERSE 1;AT 6,10;"
90 OVER 0
  
```

Al realizar gráficos con la sentencia «OVER» activada, se obtienen unos resultados artísticos.

El siguiente programa realiza unos círculos crecientes, tomando como origen las cuatro esquinas; aproximadamente tarda dos minutos en completar el gráfico.



El programa número «7», con la función «OVER» activada, genera un punto aleatorio, a partir del cual se trazan líneas rectas hacia los bordes de la zona de visualización, posteriormente se genera otro y se repite el proceso, de esta manera se obtiene una combinación «espectacular»; el color del fondo va cambiando de color, con una temporización, mientras que la «tinta» tiene color de contraste.

SCREEN\$

Acceso al teclado

LEN

MODO **E**

SYMBOL SHIFT



SCREEN\$

Tipo de sentencia

Función auxiliar.

Definición

Esta función retorna una cadena con el carácter existente en una determinada posición. Su estructura general es la siguiente:

| SENTENCIA | ARGUMENTO |
|-----------|-----------------|
| SCREEN\$ | (fila, columna) |

Ejemplo:

- PRINT SCREEN\$ (5,7)
- IF SCREEN\$ (2, 3) = "9" THEN...
- LET a\$ = SCREEN\$ (10, 11)
- LET b = VAL (SCREEN\$ (7, 4))

«SCREEN\$» reconoce cualquier carácter comprendido entre los códigos 32 (espacio) y el 127 (©) en decimal; independientemente del atributo temporal que tengan.

Ejemplo:

```

10 PRINT FLASH 1; AT 10, 10;
  (HOP)
20 PRINT SCREEN$ (10, 11)
  
```

Esta función retorna una cadena vacía cuando la posición especificada contiene un carácter fuera del rango.

Ejemplo:

```

10 PRINT CHR$ 135
20 PRINT SCREEN$ (0, 0)
  
```

El programa número «8» genera un laberinto y utiliza la función «SCREEN\$» para detectar los muros y los objetos.

Almacenamiento de pantallas

«SCREEN\$» puede utilizarse conjuntamente con las sentencias «SAVE» y «LOAD»

para grabar en cinta y posteriormente cargar la imagen (gráficos + texto) que hay en pantalla.

La estructura general es:
a) Salvar pantallas.

SAVE nombre SCREEN\$

donde el nombre es una cadena entrecomillada de un máximo de diez caracteres.

Ejemplo:

SAVE (batalla) SCREEN\$

b) Cargar pantallas.

LOAD nombre SCREEN\$

Ejemplo:

LOAD (batalla) SCREEN\$

La sentencia «VERIFY» no opera conjuntamente con «SCREEN\$», de manera que no puede verificarse la grabación.

POINT

Acceso al teclado



MODO **E**

POINT SYMBOL SHIFT

Tipo de sentencia

Función auxiliar

Definición

La función «POINT» indica si un pixel es de color de «tin-

ta» o de «papel».

Su estructura general es:

| SENTENCIA | ARGUMENTO |
|-----------|----------------------|
| POINT | (coord. X, coord. Y) |

Ejemplos:

- PRINT POINT (12, 120)
- LET a = POINT (70, 40)
- IF POINT (5, 3) = 1 THEN...
- PRINT POINT (X, Y)

La función «POINT» retorna un «cero» si el pixel especificado tiene color de «papel» y un «uno» si es de color de «tinta».

Ejemplo:

a)

| |
|-----------------------|
| PLOT 10, 120 |
| PRINT POINT (10, 121) |

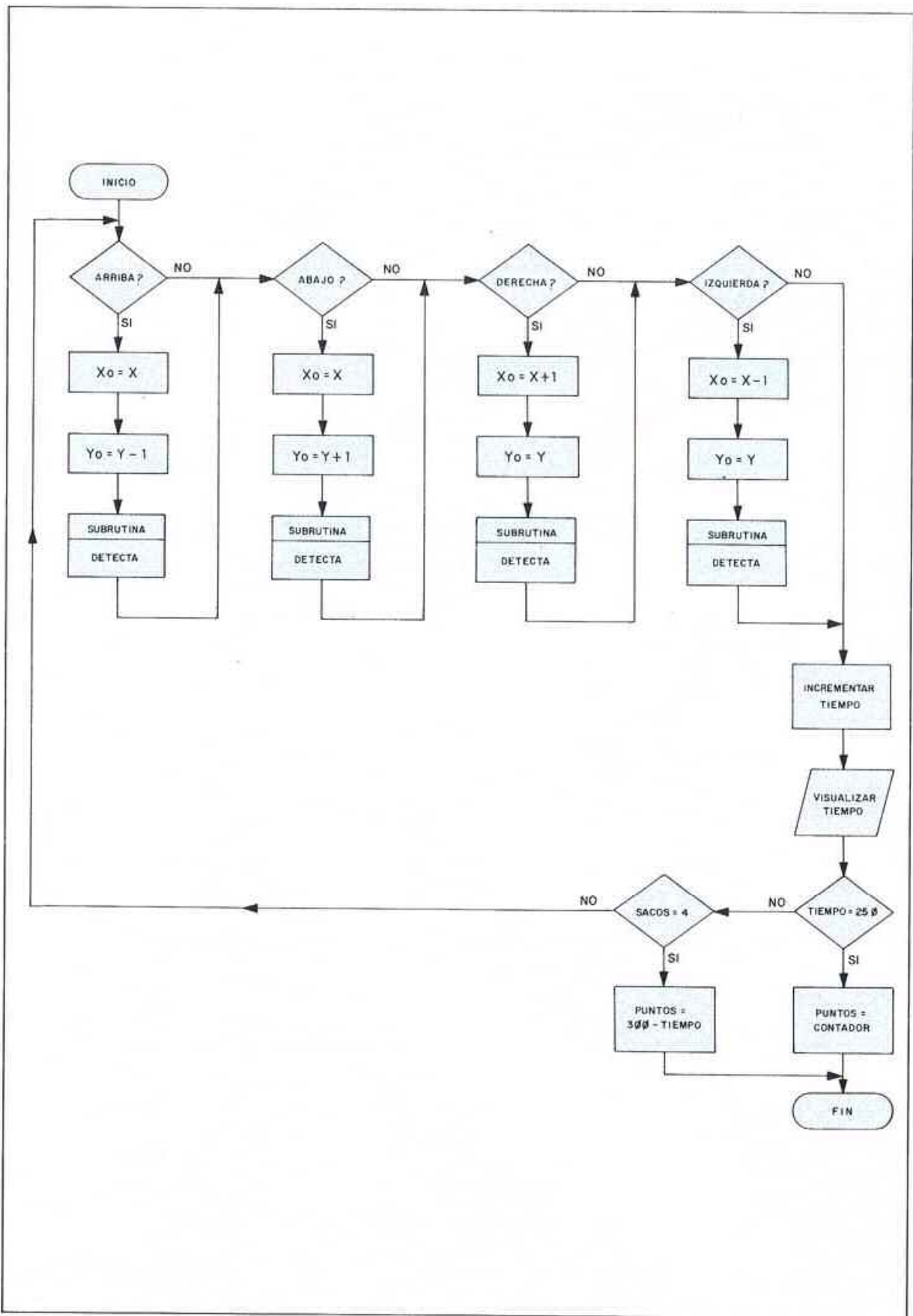
b)

| |
|-----------------------|
| PLOT 60, 127 |
| PRINT POINT (60, 127) |

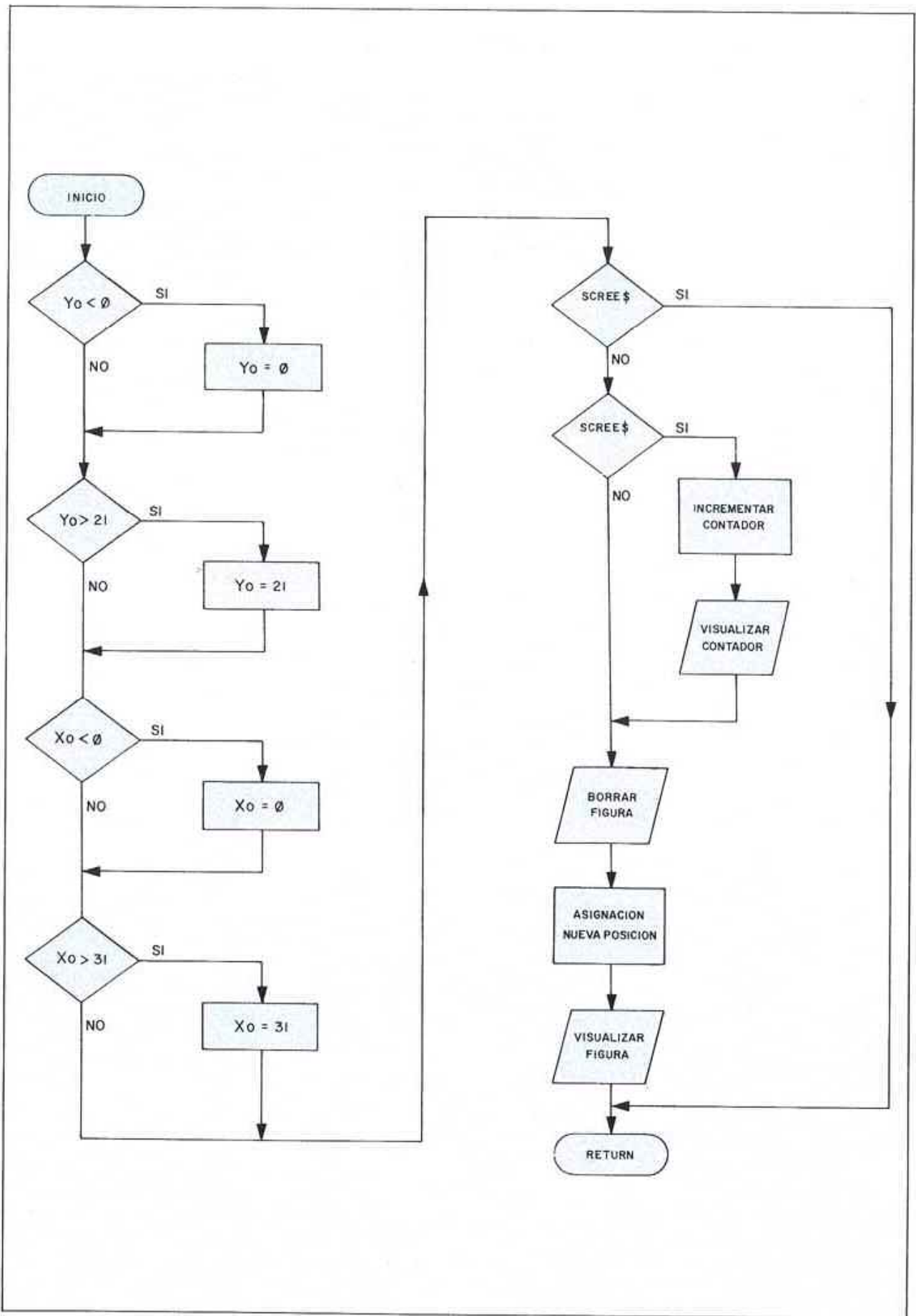
Programa

La estructura del programa «LABERINTO», el número 8, es la siguiente:

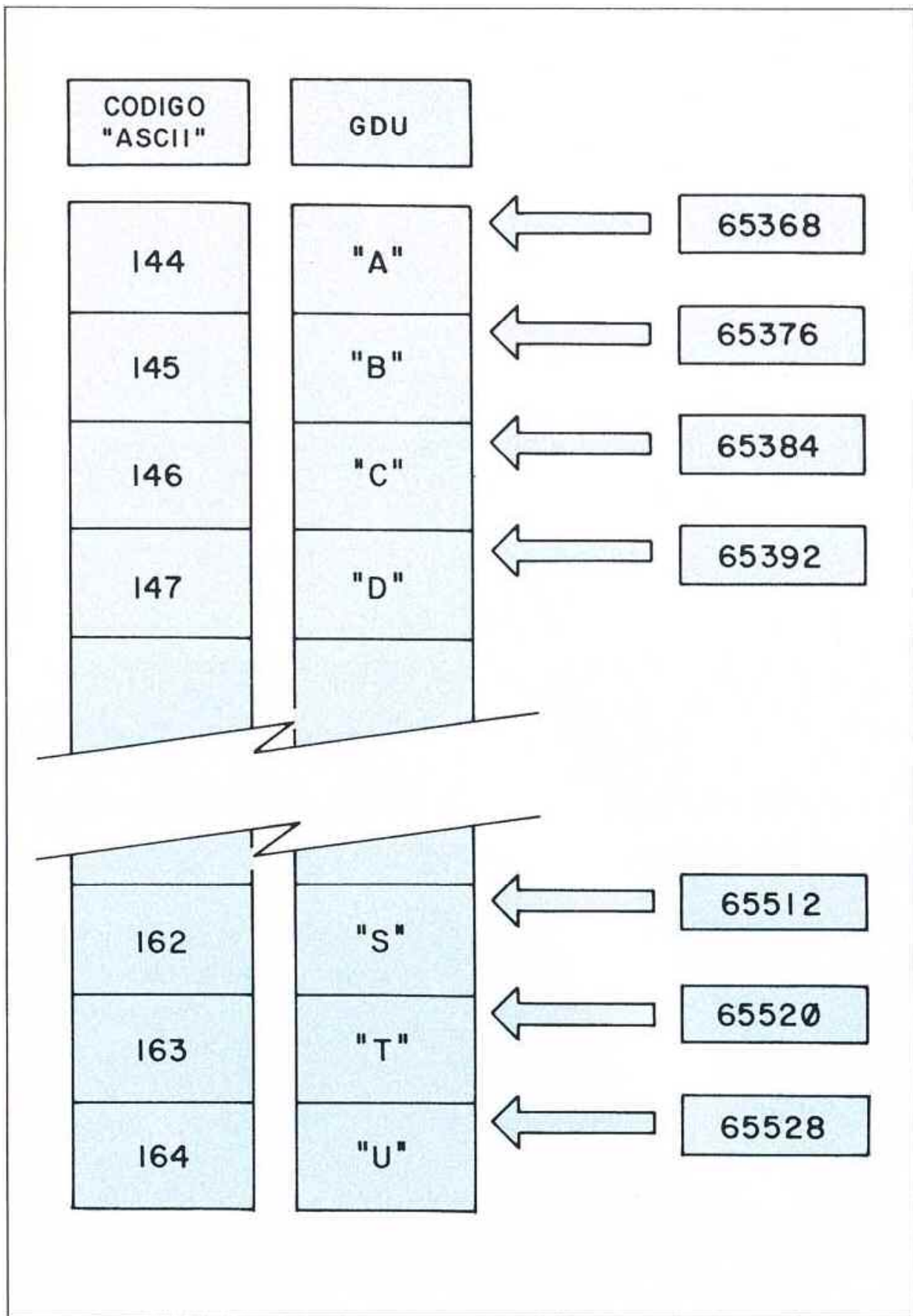
| | | | |
|---------|--|-----------|--|
| 10 | : Comentario con el nombre del programa. | | |
| 12 | : Asignación de los colores (azul para borde y fondo, y blanco) para los caracteres. | | |
| 20 | : Llamada a la subrutina de visualización de instrucciones. | | |
| 22 | : Asignación a la variable | 412 | : Incremento en una unidad |
| 24-28 | : Asignación de los valores iniciales de las variables (contador, tiempo) y (recorrido). | 418 | |
| 40-60 | : Rellenar el área de visualización con los muros del laberinto, utilizando el símbolo de (copyright) (c) en inversa y tinta verde. | 419 | |
| 80-160 | : Visualización de las habitaciones, dentro del muro, utilizando el carácter espacio. | 502 | |
| 180-210 | : Bucle para la lectura y visualización de los caminos. | 510 | |
| 220-270 | : Tabla con los datos correspondientes a las coordenadas (X) e (Y) de los caminos. | 520 | |
| 290 | : Visualización de los sacos, utilizando el símbolo del dólar con atributo temporal de (FLASH). | 530 | |
| 310-340 | : Generación aleatoria de la figura, en cualquiera de las posiciones de las líneas 0 o 21. | 540-580 | |
| 350 | : Visualización de la figura a desplazar utilizando uno de los símbolos predefinidos. | 590 | |
| 354-358 | : Visualización de los tres marcadores del juego: <ul style="list-style-type: none"> - Tiempo transcurrido. - Sacos recogidos. - Puntuación máxima. | 600 | |
| 380-410 | : Comprobación de la tecla pulsada, asignación de las nuevas coordenadas y llamada a la subrutina (detecta). | 610 | |
| 412 | : Incremento en una unidad | 620 | |
| | | 630 | |
| | | 1000 | |
| | | 1010-1040 | |
| | | 1050 | |
| | | 1052 | |
| | | 1060 | |
| | | 1070 | |
| | | 1080 | |
| | | 2000-2100 | |
| | | 2110-2260 | |
| | | 14-416 | : Visualización del tiempo, presentando las unidades en la misma posición. |
| | | 418 | : Comprueba si ha transcurrido el tiempo máximo y calcula la puntuación obtenida. |
| | | 419 | : Comprueba si se han recogido los cuatro sacos y calcula la puntuación. |
| | | 502 | : Borrado de la figura. |
| | | 510 | : Visualización de la puntuación obtenida. |
| | | 520 | : Cambia el color del papel y de la tinta, de la línea 0. |
| | | 530 | : Temporización. |
| | | 540-580 | : Comprueba si se desea jugar otra partida. |
| | | 590 | : Borrado del mensaje visualizado en el canal 0. |
| | | 600 | : Comprueba si los puntos obtenidos superan la puntuación máxima. |
| | | 610 | : Visualización del nuevo record. |
| | | 620 | : Borrado del mensaje de la línea 0. |
| | | 630 | : Inicialización de las variables (tiempo) y (contador). |
| | | 1000 | : Comienzo de la subrutina (detecta). |
| | | 1010-1040 | : Comprueban que la figura no se sale de la pantalla. |
| | | 1050 | : Comprueba si la nueva posición es un muro. |
| | | 1052 | : Comprueba si la nueva posición es un saco, y aumenta y visualiza el valor de la variable contador. |
| | | 1060 | : Borra la antigua posición. |
| | | 1070 | : Calcula la nueva posición. |
| | | 1080 | : Visualiza la posición de la figura. |
| | | 2000-2100 | : Presentación del juego. |
| | | 2110-2260 | : Presentación de las instrucciones. |



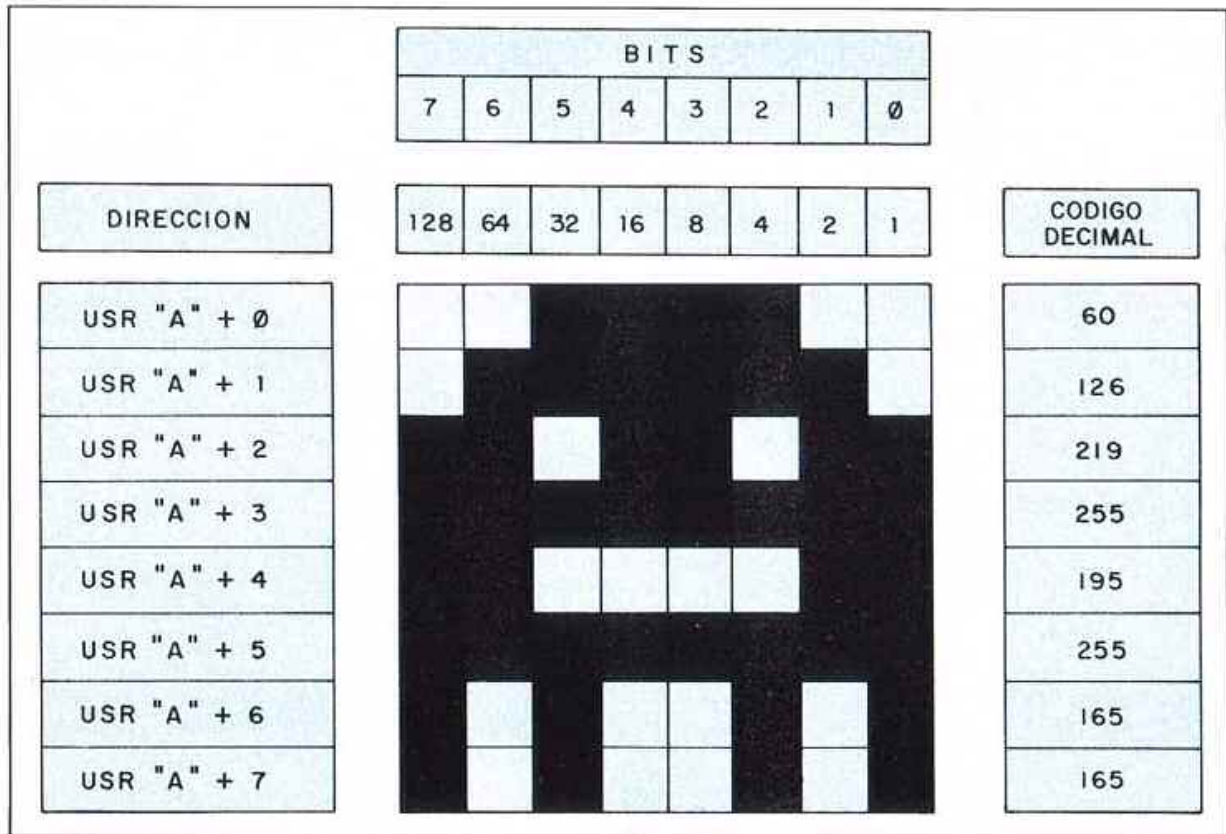
Rutina movimiento. Programa laberinto.



Subrutina detecta. Programa laberinto.



Direcciones de comienzo «GDU» (48 K).



«GDU» asignado a la «A».

daria de la siguiente manera:

```
POKE USR (a), dato
```

Definición de «GDU»

Para definir un gráfico deberemos primeramente dibujar una cuadrícula de ocho por ocho celdas o cuadraditos; sobre ésta sombrearemos aquellos cuadraditos que nos interesen para formar nuestro dibujo, de una forma similar a la representada en la figura del «terrible monstruo».

Una vez que tengamos el dibujo completo, pasaremos a su programación, para ello, utilizaremos el método más simple que consiste en utilizar la codificación binaria, en la que un pixel o cuadradito sombreado es un «1», y por el

contrario, uno no sombreado es un «0». Deberemos introducir la información «byte» a «byte» en direcciones de memoria consecutivas.

Ejemplo:

```
10 REM *****
      *      GDU      *
      *  METODO 1  *
      * ***** *
20 POKE USR "a"+0,BIN 00111100
30 POKE USR "a"+1,BIN 01111110
40 POKE USR "a"+2,BIN 10110111
50 POKE USR "a"+3,BIN 11111111
60 POKE USR "a"+4,BIN 11000011
70 POKE USR "a"+5,BIN 11111111
80 POKE USR "a"+6,BIN 10100101
90 POKE USR "a"+7,BIN 10100101
100 REM *****
110 PRINT RT 11,10,CHR$ 144
```

El anterior gráfico queda asignado a la letra «A»; para asignarlo a la letra «C», sustituye la expresión USR «a» por USR «c».

Un segundo método, más cómodo de teclear, es sustituir la codificación binaria por su correspondiente código

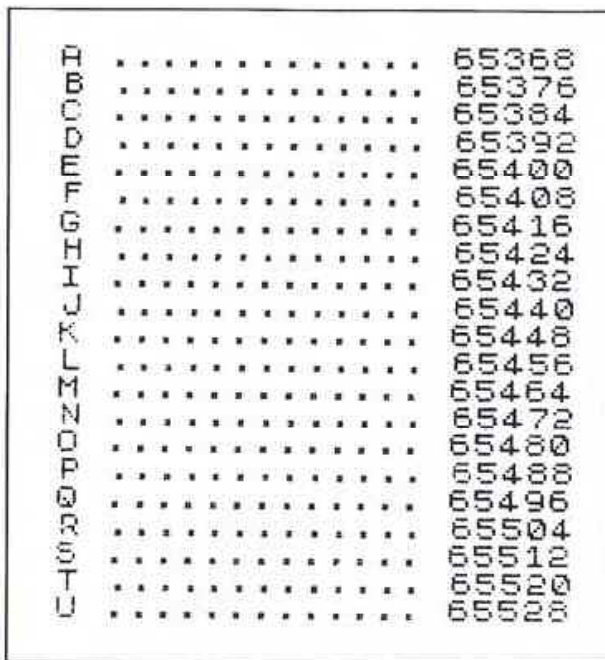
decimal; calcula dichos valores utilizando la función «BIN» en combinación con «PRINT». Ejemplo:

```
PRINT BIN 00111100
```

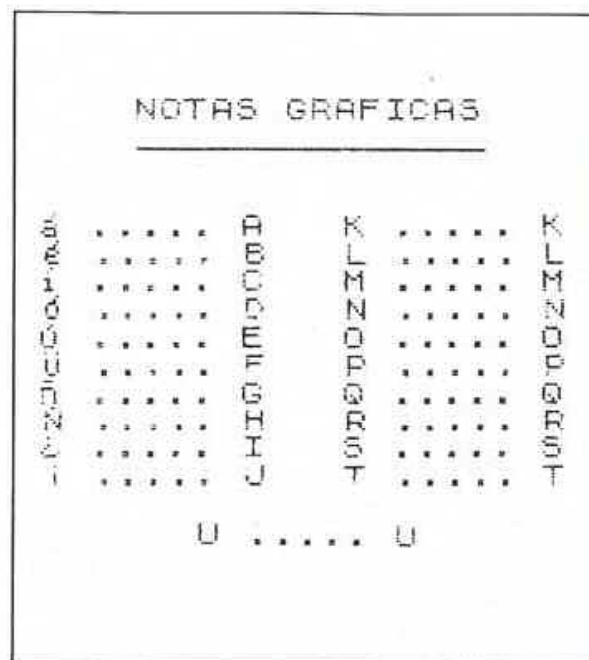
y así con cada uno de los distintos «bytes». El programa quedaría de la siguiente manera:

```
10 REM *****
      *      GDU      *
      *  METODO 2  *
      * ***** *
20 POKE USR "a"+0,60
30 POKE USR "a"+1,126
40 POKE USR "a"+2,219
50 POKE USR "a"+3,255
60 POKE USR "a"+4,195
70 POKE USR "a"+5,255
80 POKE USR "a"+6,165
90 POKE USR "a"+7,165
100 REM *****
110 PRINT RT 11,10,CHR$ 144
```

Un tercer método, utilizado con más frecuencia, es el de incluir los valores decimales en una tabla de datos, y me-



Direcciones de comienzo de los «GDU». 48 K.



Ejemplo Programa 1.

diante un bucle efectuar su lectura y posterior almacenamiento.

Ejemplo:

```

10 REM *****
      GDU
      METODO 3
*****
20 FOR n=USR "a" TO USR "a"+7
30 READ valor
40 POKE n,valor
50 NEXT n
60 REM TABLA DE DATOS
70 DATA 60,126,219,255,195,255,
165,165
80 REM MUESTRA GRABON
90 PRINT AT 11,16,CHR$ 144
  
```

Podemos utilizar varios «GDU» para formar un gráfico más grande y preciso. Para ello, deberemos definir inde-

pendientemente cada bloque de 64 pixel.

El siguiente ejemplo nos genera la figura de un helicóptero:

```

10 REM *****
      HELICOPTERO
*****
20 FOR n=USR "a" TO USR "a"+7
30 READ valor
40 POKE n,valor
50 NEXT n
60 REM TABLA DE DATOS
70 DATA 255,4,31,63,49,63,100,
100,245
80 DATA 224,0,129,195,255,240,
0,245
90 REM MUESTRA GRABON
100 PRINT AT 9,14,CHR$ 144 AT 9,
15,CHR$ 145
110 PRINT AT 11,16,CHR$ 144,CHR$
145
  
```

Utilización de los «GDU»

Para utilizar los «GDU» debe

seleccionarse el modo gráfico [G], para ello:

- Pulsar «CAPS SHIFT» simultáneamente con la tecla «9». Aparecerá parpadeante una G.

- Pulsar la tecla correspondiente al gráfico elegido. Si el gráfico aún no se ha generado, porque el programa no se ha ejecutado, aparecerá en su lugar la letra correspondiente a la tecla pulsada, en mayúsculas.

Al ejecutar el programa, la letra será sustituida por el «GDU» correspondiente y por tanto aparecerá en el listado.

- Para borrar, estando en modo gráfico, basta con pul-

PROGRAMA 3

```

10 REM *****
      * CURSO/BASICO *
      * *
      * LECTURA "GDU" *
      * *
*****
20 LET z=USR "a"
30 FOR x=0 TO 20
40 FOR n=z TO z+7
50 PRINT n;" .....",PEEK n
  
```

```

60 NEXT n
62 PRINT AT 16,0,"GRAFICO ","",
" ",CHR$ (65+x)," ",CHR$ (144+x)
)
63 IF x=20 THEN PAUSE 0: STOP
64 PRINT #0,"Pulsa una tecla p
ara continuar"

66 PAUSE 0
68 CLS
70 LET z=z+8
80 NEXT x
  
```



```

150 LET X=INT (RND*7)+13
160 GO SUB 8000
170 LET Y=7
172 LET X=INT (RND*7)+23
180 GO SUB 8000
190 REM LLAVES
200 LET X=INT (RND*14)+11
210 PRINT BRIGHT 1, INK 2:AT 13
220
230 LET X=INT (RND*7)+3
240 PRINT BRIGHT 1, INK 3:AT 9,
250
260 LET X=INT (RND*14)+7
270 PRINT BRIGHT 1, INK 4:AT 5,
280
290 LET X=INT (RND*22)+7
300 PRINT BRIGHT 1, INK 5:AT 1,
310
320 REM OBJETOS
330 PRINT INK 2,AT 15,29,"X"; I
340 INK 3,AT 11,29,"O"; INK 4,AT 7,2;
350 INK 5,AT 3,2,"I"
360 REM FIGURA
370 LET YF=15: LET XF=1
380 PRINT INK 4,AT YF,XF,"X"
390 REM CONTADORES
400 FOR N=0 TO 31
410 PRINT INK 2,AT 17,N,"■":AT
420 21,N,"■"
430 NEXT N
440 FOR N=18 TO 20
450 PRINT INK 2,AT N,0,"|":AT N
460 13,"|":AT N,31,"|"
470 NEXT N
480 PRINT INK 1,AT 19,1:"VIDAS:"
490 INK 4,"X X X"
500 PRINT INK 1,AT 19,15:"OBJET
510
520 REM MOVIMIENTO
530 LET PASO=0 LET INCREMENTO=
540 1
550 PRINT INK 6,AT 3,20+PASO,"■"
560 AT 7,8+PASO,"■":AT 11,3+PASO,"■"
570 AT 15,16+PASO,"■"
580 IF ATTR (YF,XF+1)=6 OR ATTR
590 (YF,XF-1)=6 THEN GO SUB 8100
600 IF INKEY$="q" THEN GO SUB 7
610
620 IF SUBIR=1 THEN GO TO 600
630 IF INKEY$="o" THEN LET salt
640 =1: LET XO=XF+1: LET YO=YF: GO
650 310 7000 IF OBJETO=4 THEN LET r
660 i:=1: GO TO 1000
670 IF INKEY$="b" THEN LET salt
680 =-1: LET XO=XF-1: LET YO=YF: GO
690 310 7000 IF OBJETO=4 THEN LET
700 fin:=1: GO TO 1000
710 IF INKEY$="a" THEN GO SUB 8
720
730 IF INKEY$="s" THEN GO TO 58
740
750 GO TO 500
760 IF INKEY$="c" THEN GO TO 60
770
780 GO TO 580
790 PRINT AT 3,20+PASO," " :AT 7
800 8+PASO," " :AT 11,3+PASO," " :AT
810 15,16+PASO," "
820 LET PASO=PASO+INCREMENTO
830 IF PASO=8 THEN LET INCREMENTO
840 =-1
850 IF PASO=0 THEN LET INCREMENTO
860 =1
870 GO TO 510
880 REM FIN
890 FOR Z=1 TO 10
900 FOR N=7 TO 0 STEP -1
910 BORDER N
920 NEXT N
930 NEXT Z
940 CLS
950 IF FIN<>0 THEN GO TO 1090
960 PRINT AT 4,2:"Lo siento, tu
970 esfuerzo ha sido "insuficient
980 e, por este motivo el "rey Gum
990 ersindo encerrara a Palitroque
000 que en los calabozos del si-
010 nuestro castillo."
020 GO TO 1100

```

```

1090 PRINT AT 4,2:"¡ ENHORABUEN
1100 A " por haberlo "conseguido,
1110 el rey Gumersindo fe" "licitara
1120 a Palitroque y le deja-" "ra e
1130 n custodia la llave del cin-"
1140 turon de castidad de la reina.
1150 PRINT #0,AT 1,0:"Deseas int
1160 entarlo otra vez (S/N)"
1170 IF INKEY$="n" THEN CLS ST
1180 OP
1190 IF INKEY$="s" THEN CLS : GO
1200 TO 5
1210 GO TO 1110
1220 REM SUB. DETECTA
1230 IF ATTR (YO,XO)=7 OR ATTR (
1240 YO,XO)=71 THEN GO TO 7030
1250 IF ATTR (YO,XO)=178 THEN LE
1260 T BORRA=1: GO TO 7030
1270 GO SUB 7100: IF BORRA=1 THE
1280 N GO TO 7030
1290 RETURN
1300 PRINT OVER 1, BRIGHT 1,AT Y
1310 F,XF,"X"
1320 LET XF=XO LET YF=YO
1330 IF BORRA=1 THEN PRINT INK 4
1340 AT YF,XF,"X" LET BORRA=0 RETU
1350 RN
1360 PRINT OVER 1, INK 4,AT YF,X
1370 F,"X"
1380 RETURN
1390 REM SUB. COG. OBJETO
1400 IF ATTR (YO,XO)=2 THEN PRIN
1410 T BRIGHT 1, INK 2,AT 19,POSICION
1420 "X" LET BORRA=1 LET POSICION=
1430 POSICION+2 GO TO 7160
1440 IF ATTR (YO,XO)=3 THEN PRIN
1450 T BRIGHT 1, INK 3,AT 19,POSICION
1460 "O" LET BORRA=1 LET POSICION=
1470 POSICION+2 GO TO 7160
1480 IF ATTR (YO,XO)=4 THEN PRIN
1490 T BRIGHT 1, INK 4,AT 19,POSICION
1500 "W" LET BORRA=1 LET POSICION=
1510 POSICION+2 GO TO 7160
1520 IF ATTR (YO,XO)=5 THEN PRIN
1530 T BRIGHT 1, INK 5,AT 19,POSICION
1540 "V" LET BORRA=1 LET POSICION=
1550 POSICION+2 GO TO 7160
1560 RETURN
1570 LET OBJETO=OBJETO+1
1580 RETURN
1590 REM SUB. SUBIR
1600 IF ATTR (YF-1,XF)=71 THEN L
1610 ET SUBIR=1: GO TO 7230
1620 IF SUBIR=1 THEN GO TO 7230
1630 RETURN
1640 IF ATTR (YF-1,XF)=6 THEN LE
1650 T YF=YF: LET XF=XF: GO SUB 8100
1660 PRINT AT YF,XF, BRIGHT 1, INK 7
1670 "A" RETURN
1680 LET PELDANO=PELDANO+1
1690 PRINT OVER 1, BRIGHT 1,AT Y
1700 F,XF,"X"
1710 LET YF=YF-1
1720 PRINT OVER 1, INK 4,AT YF,X
1730 F,"X"
1740 IF PELDANO=4 THEN LET SUBIR
1750 =0: LET PELDANO=0
1760 RETURN
1770 REM SUB. DIB. ESCALERA
1780 PRINT BRIGHT 1,AT Y,X,"H":A
1790 T Y-1,X,"A":AT Y-2,X,"A":AT Y-3,
1800 X,"A"
1810 RETURN
1820 REM SUB. ATRAPA
1830 FOR N=0 TO 10
1840 PRINT INK 4,AT YF,XF, OVER
1850 1,"X"
1860 FOR X=0 TO 5 NEXT X
1870 NEXT N
1880 PRINT AT YF,XF:" "
1890 IF VIDA=1 THEN GO TO 8170
1900 PRINT AT 19,POS," "
1910 LET POS=POS-2
1920 LET VIDA=VIDA-1
1930 IF VIDA=0 THEN LET FIN=0: G
1940 O TO 1000
1950 LET YF=15: LET XF=1
1960 PRINT AT YF,XF, INK 4,"X"
1970 LET SUBIR=0: LET PELDANO=0

```

```

8204 LET salto=1
8210 RETURN
8220 REM SUB. SALTO
8222 IF ATTR (yf,xf+salto)=50 TH
EN GO SUB 8100: RETURN
8224 PRINT OVER 1, BRIGHT 1, AT y
f,xf,"*"
8230 LET xf=xf+salto: LET yf=yf-
1
8240 PRINT OVER 1, INK 4, AT yf,x
f,"*"
8250 IF ATTR (yf+1,xf)=6 OR ATTR
(yf,xf+salto)=50 THEN GO SUB 81
00: RETURN
8252 GO SUB 8500
8260 PRINT OVER 1, BRIGHT 1, AT y
f,xf,"*"
8270 LET xf=xf+salto: LET yf=yf-
1
8280 PRINT OVER 1, INK 4, AT yf,x
f,"*"
8290 IF ATTR (yf,xf+salto)=50 TH
EN GO SUB 8100: RETURN
8300 GO SUB 8500
8310 PRINT OVER 1, BRIGHT 1, AT y
f,xf,"*"
8320 LET xf=xf+salto: LET yf=yf
8330 PRINT OVER 1, INK 4, AT yf,x
f,"*"
8340 IF ATTR (yf,xf+salto)=50 TH
EN GO SUB 8100: RETURN
8350 PRINT OVER 1, BRIGHT 1, AT y
f,xf,"*"
8370 LET xf=xf+salto: LET yf=yf+
1
8380 PRINT OVER 1, INK 4, AT yf,x
f,"*"
8390 IF ATTR (yf+1,xf)=6 OR ATTR
(yf,xf+salto)=50 THEN GO SUB 81
00: RETURN
8400 PRINT OVER 1, BRIGHT 1, AT y
f,xf,"*"
8410 LET xf=xf+salto: LET yf=yf+
1
8420 PRINT OVER 1, INK 4, AT yf,x
f,"*"
8430 RETURN
8500 REM SUB. REC. LLAVES
8510 IF ATTR (yf-1,xf+salto)=66
THEN PRINT AT yf-1,xf+salto,""
GO TO 8600
8520 IF ATTR (yf,xf+salto)=66 TH
EN PRINT AT yf,xf+salto,"" GO
TO 8600
8530 IF ATTR (yf-1,xf+salto)=67
THEN PRINT AT yf-1,xf+salto,""
GO TO 8610
8540 IF ATTR (yf,xf+salto)=67 TH
EN PRINT AT yf,xf+salto,"" GO
TO 8610
8550 IF ATTR (yf-1,xf+salto)=68
THEN PRINT AT yf-1,xf+salto,""
GO TO 8620
8560 IF ATTR (yf,xf+salto)=68 TH
EN PRINT AT yf,xf+salto,"" GO
TO 8620
8570 IF ATTR (yf-1,xf+salto)=69
THEN PRINT AT yf-1,xf+salto,""
GO TO 8630
8580 IF ATTR (yf,xf+salto)=69 TH
EN PRINT AT yf,xf+salto,"" GO
TO 8630
8590 RETURN
8600 PRINT FLASH 1, PAPER 6, INK
2, AT 15,27,"*": RETURN
8610 PRINT FLASH 1, PAPER 6, INK
2, AT 11,27,"*": RETURN
8620 PRINT FLASH 1, PAPER 6, INK
2, AT 7,4,"*": RETURN
8630 PRINT FLASH 1, PAPER 6, INK
2, AT 3,4,"*": RETURN
9000 REM SUB. GRAFICOS
9002 FOR n=USR "a" TO USR "j"+7
9010 READ dato
9020 POKE n,dato
9030 NEXT n
9040 DATA 252,252,252,0,231,231,
231,0
9050 DATA 126,66,66,66,126,66,66
,66

```

```

9060 DATA 4,10,17,10,20,32,60,32
9070 DATA 0,10,4,10,16,32,64,0
9080 DATA 0,24,60,60,24,60,60,0
9090 DATA 195,36,126,219,255,66,
126,219
9100 DATA 24,66,126,26,24,60,36,
102
9110 DATA 0,0,0,165,153,90,126,1
06
9120 DATA 124,124,124,124,56,16,
16,124
9130 DATA 127,119,119,65,119,119
,54,28
9140 RETURN
9200 REM CARATULA
9210 PLOT 0,0
9220 DRAW 0,175: DRAW 255,0
9230 DRAW 0,-175: DRAW -255,0
9240 PRINT AT 3,0,""
TROHOBBY
9250 PRINT AT 18,0,"" Por R
Brael Prades
9260 PRINT AT 7,11,"PALITROQUE",
AT 10,13,"Y LOS",AT 13,10,"TROG
LOLOCITOS"
9270 PRINT #0, AT 1,0,"Quieres la
s instrucciones (S/N)": PAUSE 0
9280 IF INKEY#="n" THEN CLS: RE
TURN
9290 IF INKEY#="s" THEN GO TO 93
10
9300 GO TO 9280
9310 CLS
9320 PRINT AT 0,2,"Palitroque, e
s encargado de ba" "jar a los s
otanos del castillo,"
9330 PRINT "para recoger las arm
as y objetos" "personajes del r
ey "Gumersindo" "que se va a
las Cruzadas."
9340 PRINT "La mision de Palit
roque no es" "sencilla, ya que
las habitaciones" "ness se encuent
ran cerradas."
9350 PRINT "Palitroque debera
recoger las" "llaves y tener cu
idado con los" "terribles Trogl
ocitos."
9360 PRINT #0, AT 1,1,"Pulsa una
tecla para continuar": PAUSE 0
9370 CLS
9380 PRINT AT 0,4,"PERSONAJES"
9390 PRINT AT 1,4,""
9400 PRINT BRIGHT 1, INK 4, AT 3,
4,"* Palitroque"
9410 PRINT BRIGHT 1, INK 6, AT 5,
4,"* Troglolocitos"
9420 PRINT AT 8,4,"OBJETOS"
9430 PRINT AT 9,4,""
9440 PRINT BRIGHT 1, INK 2, AT 11
,4,"* Espada Gloriosa"
9450 PRINT BRIGHT 1, INK 3, AT 13
,4,"* Escudo Protector"
9460 PRINT BRIGHT 1, INK 4, AT 15
,4,"* Corona Real"
9470 PRINT BRIGHT 1, INK 5, AT 17
,4,"* Nectar de los Dioses"
9480 PRINT BRIGHT 1, INK 6, AT 19
,4,"* Llaves"
9490 PRINT #0, AT 1,1,"Pulsa una
tecla para continuar": PAUSE 0
9500 CLS
9510 PRINT AT 0,4,"CONTROLES"
9520 PRINT AT 1,4,""
9530 PRINT AT 3,4,"0 - IZQUIERDA
"
9540 PRINT AT 5,4,"P - DERECHA"
9550 PRINT AT 7,4,"O - SUBIR"
9560 PRINT AT 9,4,"A - SALTAR"
9570 PRINT AT 12,4,"S - Para el
juego"
9580 PRINT AT 14,4,"C - Continua
"
9590 PRINT #0, AT 1,1,"Pulsa una
tecla para comenar": PAUSE 0
9600 CLS: RETURN

```

sar la tecla «Ø».

— Para retornar al modo anterior, pulsa la tecla «9».

También pueden utilizarse los «GDU», haciendo referencia a su correspondiente código «ASCII». Por ejemplo, para visualizar el gráfico asignado a la letra «A» utiliza:

```
PRINT CHR$ 144
```

para utilizar otro gráfico consulte la tabla de la página 41.

Al estar situada la zona de memoria de los «GDU» por encima de una *variable del sistema* conocida por el nombre de «RAMTOP», al ejecutar una sentencia del tipo «NEW» se borra la zona de memoria destinada para almacenar nuestro programa BASIC, pero en cambio, permanecen *inalterables* nuestros gráficos, a no ser que desconectemos el ordenador o hagamos un RESET.

Programas de aplicación

El programa número «1» nos permite conocer cuales son los GDU que tenemos almacenados en ese momento en el ordenador y a qué letras están asignados.

El programa «2» genera como GDU una serie de letras y símbolos utilizados en el idioma español que no existen en el teclado del Spectrum, como por ejemplo, la «ñ», la apertura de interrogación «¿», la u con diéresis «ü», etc. Al pulsar una tecla se ejecuta la sentencia NEW que nos borra el programa, pero nos respeta la zona de memoria de los GDU; podemos comprobarlo pasando a modo  y pulsando cualquier tecla de la «A» a la «J».

Esta aplicación nos permite

confeccionar textos en español.

Grabación de GDU

Podemos grabar en cinta la zona de memoria donde están almacenados los GDU, de esta manera podemos utilizarlos en otra ocasión sin necesidad de tenerlos que definir de nuevo.

La estructura de la sentencia «SAVE» es algo distinta de la utilizada en la grabación de programas, ya que debe especificarse en este caso la dirección de memoria a partir de la cual se desea grabar, así como la longitud en «bytes»:

```
SAVE (nombre) CODE comienzo,  
longitud
```

La palabra clave «CODE» identifica que no es la zona de memoria donde está almacenado el programa lo que se desea grabar, sino la relacionada en los parámetros «comienzo» y «longitud».

Ejemplo:

```
SAVE (gdu1) CODE USR (a), 168
```

De esta manera, se almacenarán en cinta los 21 posibles GDU. Si por el contrario deseáramos grabar los GDU correspondientes a las letras «C» a «I», ambos inclusive, utilizaríamos:

```
SAVE (gdu2) CODE USR (C), 56
```

ya que 56 es el resultado de multiplicar 7 gráficos por 8 bytes cada uno.

Para realizar el proceso inverso, es decir, almacenar en memoria los gráficos grabados en cinta, podemos utilizar cualquiera de estas opciones:

nes:

a) Si el siguiente programa a leer, grabado con «CODE» es el especificado:

```
LOAD "" CODE
```

b) Especificando el nombre:

```
LOAD (gdu2) CODE
```

c) Especificando la dirección de comienzo:

```
LOAD (gdu2) CODE USR (c)
```

Este método es más correcto ya que es independiente de la cantidad de memoria que posea el ordenador, por lo tanto él calcula la nueva dirección de carga.

d) Especificando también la longitud:

```
LOAD (gdu2) CODE USR (c), 56
```

Lectura de los GDU

El programa número «3» nos visualiza en la pantalla las direcciones de cada «byte» de un GDU, así como su contenido.

El correspondiente gráfico nos aparece en la parte inferior de la pantalla.

Programa generador de GDU

En las cintas demostración que acompañan tanto al Spectrum 16 o 48 K como el Plus, vienen grabados unos programas con los que se pueden generar con facilidad los GDU. Estos cuentan con una serie de opciones que permiten generar, modificar,

grabar o leer cualquier GDU, y asignarlo a la letra que se desea.

Programa

El programa número «4», que da fin a este capítulo, es una aplicación de los GDU a los juegos.

Las instrucciones son sencillas.

El personaje principal se maneja con las siguientes teclas:

| |
|---------------|
| O - Izquierda |
| P - Derecha |
| Q - Subir |
| A - Saltar |

Hay otras dos opciones que permiten parar el juego o continuar:

| |
|-------------------|
| S - Para el juego |
| C - Continúa |

La misión de «Palitroque», que es el personaje principal, consiste en recoger los diversos objetos que se encuentran diseminados por las habitaciones, pero no es tan sencilla, ya que las habitaciones se encuentran cerradas. Deberá recoger las llaves que se encuentran suspendidas del techo y tener cuidado con unos bichos llamados «Troglocitos», que le impedirán el paso.

«Palitroque» no dispone de ningún arma, el único modo de esquivar a los terribles «Troglocitos» es saltar sobre ellos, pero deberá tener cuidado de no pisarles y no chocarse con las paredes ya que cualquier golpe eliminará una de nuestros cuatro vidas.

Para pasar de un piso a otro disponemos de una escalera

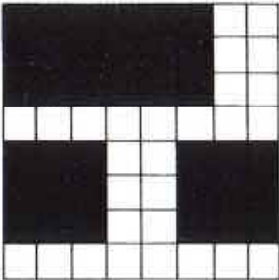
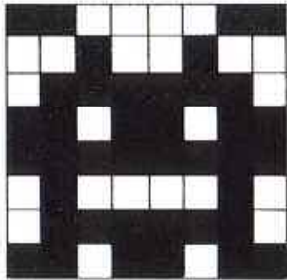
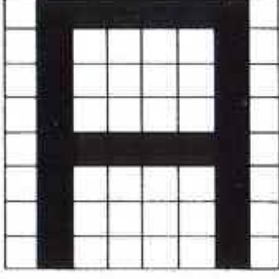
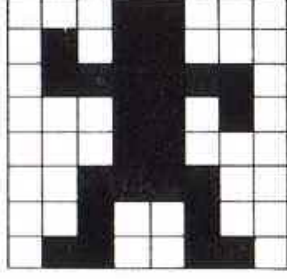
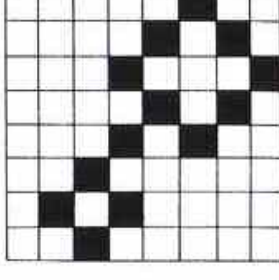
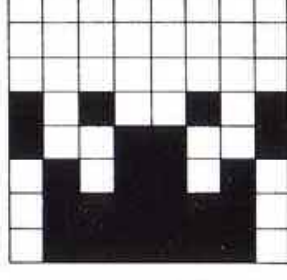
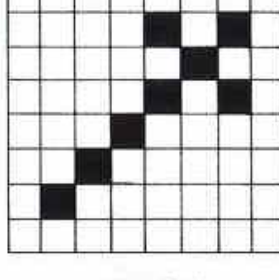
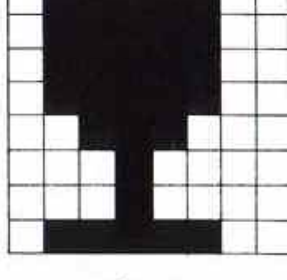
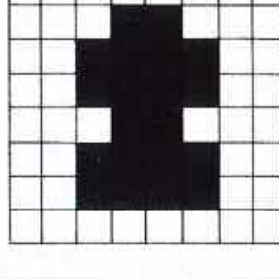
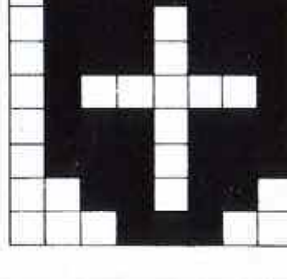
por la que se puede subir pero no bajar.

Si conseguimos realizar la misión, el rey «Gumersindo» premiará a «Palitroque» con su mayor tesoro.

La estructura del programa es la siguiente:

- 1 : Comentario con el nombre del programa.
- 2 : Mensaje de espera parpadeante.
- 3 : Realiza las siguientes tareas:
 - Inicialización de las tablas de datos.
 - Inicialización de la función aleatoria.
 - Llamada a la subrutina que define los GDU.
- 4 : Asignación de los colores negro con borde y papel, y blanco para los caracteres. Llamada a la subrutina que visualiza las instrucciones.
- 6-12 : Inicialización de las variables utilizadas en el juego.
- 16 : Utilización de la sentencia «POKE» para inicializar la variable del sistema «FLASG2» localizada en la dirección 23658 con el valor 0, de esta forma se selecciona el modo minúsculas [L] y se simplifica la tarea de detección de teclas pulsadas (INKEY\$).
- 70-90 : Visualización de los techos y suelos del castillo.
- 100-120 : Visualización de las paredes.
- 122-128 : Visualización de las habitaciones y de las cerraduras.
- 130-180 : Cálculo aleatorio de la posición (X) de la escalera y llamada a la subrutina de visualización.
- 200-270 : Cálculo aleatorio de la coordenada (X) de la llave y visualización de la misma.

- 290 : Visualización de los objetos dentro de las habitaciones.
- 310 : Asignación a las variables (x) e (y) de la posición inicial de «Palitroque».
- 320 : Visualización de «Palitroque».
- 340-390 : Visualización del recuadro inferior que sirve para el contador de vidas y de objetos.
- 400-410 : Visualización del número de vidas y de objetos.
- 502 : Inicialización de las variables (paso) e (incremento) utilizadas para el movimiento de los «Troglocitos».
- 510 : Visualización de los «Troglocitos» en los diversos pisos.
- 512 : Comprueba si el «Troglocito» se encuentra situado en la posición próxima a «Palitroque», en caso afirmativo se realiza un salto a la subrutina «atrapa» que entre otras cosas nos resta una vida.
- 520 : Comprueba si se pulsa la tecla «q» (subir) y salta a la subrutina «subir».
- 522 : Si no se ha completado el ciclo de subida por la escalera, el programa continúa en la línea 600.
- 530 : Comprueba si se pulsa la tecla «p» (derecha), calcula la nueva posición de «Palitroque», indica a la variable «salto» que si se realiza un salto será a la derecha, realiza una llamada a la subrutina «detecta» y comprueba si se han recogido los cuatro objetos.
- 540 : Idem, con la tecla «o» (izquierda).
- 550 : Comprueba si se pulsa la tecla «a» (saltar) para ir a la subrutina «salto».
- 560-590 : Si la tecla pulsada es la «s» (parada) se queda en un

| | | | | | | | | | | | | | | | | | | | | |
|--|--|-----|-----|-----|----|-----|-----|-----|----|---|---|---|-----|-----|-----|-----|-----|-----|-----|-----|
| <p style="text-align: center;">GDU "A"</p>  | <table border="1"> <tr><td>252</td></tr> <tr><td>252</td></tr> <tr><td>252</td></tr> <tr><td>∅</td></tr> <tr><td>231</td></tr> <tr><td>231</td></tr> <tr><td>231</td></tr> <tr><td>∅</td></tr> </table> | 252 | 252 | 252 | ∅ | 231 | 231 | 231 | ∅ | <p style="text-align: center;">GDU "F"</p>  | <table border="1"> <tr><td>195</td></tr> <tr><td>36</td></tr> <tr><td>126</td></tr> <tr><td>219</td></tr> <tr><td>255</td></tr> <tr><td>66</td></tr> <tr><td>126</td></tr> <tr><td>219</td></tr> </table> | 195 | 36 | 126 | 219 | 255 | 66 | 126 | 219 | |
| 252 | | | | | | | | | | | | | | | | | | | | |
| 252 | | | | | | | | | | | | | | | | | | | | |
| 252 | | | | | | | | | | | | | | | | | | | | |
| ∅ | | | | | | | | | | | | | | | | | | | | |
| 231 | | | | | | | | | | | | | | | | | | | | |
| 231 | | | | | | | | | | | | | | | | | | | | |
| 231 | | | | | | | | | | | | | | | | | | | | |
| ∅ | | | | | | | | | | | | | | | | | | | | |
| 195 | | | | | | | | | | | | | | | | | | | | |
| 36 | | | | | | | | | | | | | | | | | | | | |
| 126 | | | | | | | | | | | | | | | | | | | | |
| 219 | | | | | | | | | | | | | | | | | | | | |
| 255 | | | | | | | | | | | | | | | | | | | | |
| 66 | | | | | | | | | | | | | | | | | | | | |
| 126 | | | | | | | | | | | | | | | | | | | | |
| 219 | | | | | | | | | | | | | | | | | | | | |
| <p style="text-align: center;">GDU "B"</p>  | <table border="1"> <tr><td>126</td></tr> <tr><td>66</td></tr> <tr><td>66</td></tr> <tr><td>66</td></tr> <tr><td>66</td></tr> <tr><td>126</td></tr> <tr><td>66</td></tr> <tr><td>66</td></tr> <tr><td>66</td></tr> </table> | 126 | 66 | 66 | 66 | 66 | 126 | 66 | 66 | 66 | <p style="text-align: center;">GDU "G"</p>  | <table border="1"> <tr><td>24</td></tr> <tr><td>88</td></tr> <tr><td>126</td></tr> <tr><td>26</td></tr> <tr><td>24</td></tr> <tr><td>6∅</td></tr> <tr><td>36</td></tr> <tr><td>102</td></tr> </table> | 24 | 88 | 126 | 26 | 24 | 6∅ | 36 | 102 |
| 126 | | | | | | | | | | | | | | | | | | | | |
| 66 | | | | | | | | | | | | | | | | | | | | |
| 66 | | | | | | | | | | | | | | | | | | | | |
| 66 | | | | | | | | | | | | | | | | | | | | |
| 66 | | | | | | | | | | | | | | | | | | | | |
| 126 | | | | | | | | | | | | | | | | | | | | |
| 66 | | | | | | | | | | | | | | | | | | | | |
| 66 | | | | | | | | | | | | | | | | | | | | |
| 66 | | | | | | | | | | | | | | | | | | | | |
| 24 | | | | | | | | | | | | | | | | | | | | |
| 88 | | | | | | | | | | | | | | | | | | | | |
| 126 | | | | | | | | | | | | | | | | | | | | |
| 26 | | | | | | | | | | | | | | | | | | | | |
| 24 | | | | | | | | | | | | | | | | | | | | |
| 6∅ | | | | | | | | | | | | | | | | | | | | |
| 36 | | | | | | | | | | | | | | | | | | | | |
| 102 | | | | | | | | | | | | | | | | | | | | |
| <p style="text-align: center;">GDU "C"</p>  | <table border="1"> <tr><td>4</td></tr> <tr><td>1∅</td></tr> <tr><td>17</td></tr> <tr><td>1∅</td></tr> <tr><td>2∅</td></tr> <tr><td>32</td></tr> <tr><td>8∅</td></tr> <tr><td>32</td></tr> </table> | 4 | 1∅ | 17 | 1∅ | 2∅ | 32 | 8∅ | 32 | <p style="text-align: center;">GDU "H"</p>  | <table border="1"> <tr><td>∅</td></tr> <tr><td>∅</td></tr> <tr><td>∅</td></tr> <tr><td>165</td></tr> <tr><td>153</td></tr> <tr><td>9∅</td></tr> <tr><td>126</td></tr> <tr><td>126</td></tr> </table> | ∅ | ∅ | ∅ | 165 | 153 | 9∅ | 126 | 126 | |
| 4 | | | | | | | | | | | | | | | | | | | | |
| 1∅ | | | | | | | | | | | | | | | | | | | | |
| 17 | | | | | | | | | | | | | | | | | | | | |
| 1∅ | | | | | | | | | | | | | | | | | | | | |
| 2∅ | | | | | | | | | | | | | | | | | | | | |
| 32 | | | | | | | | | | | | | | | | | | | | |
| 8∅ | | | | | | | | | | | | | | | | | | | | |
| 32 | | | | | | | | | | | | | | | | | | | | |
| ∅ | | | | | | | | | | | | | | | | | | | | |
| ∅ | | | | | | | | | | | | | | | | | | | | |
| ∅ | | | | | | | | | | | | | | | | | | | | |
| 165 | | | | | | | | | | | | | | | | | | | | |
| 153 | | | | | | | | | | | | | | | | | | | | |
| 9∅ | | | | | | | | | | | | | | | | | | | | |
| 126 | | | | | | | | | | | | | | | | | | | | |
| 126 | | | | | | | | | | | | | | | | | | | | |
| <p style="text-align: center;">GDU "D"</p>  | <table border="1"> <tr><td>∅</td></tr> <tr><td>1∅</td></tr> <tr><td>4</td></tr> <tr><td>1∅</td></tr> <tr><td>16</td></tr> <tr><td>32</td></tr> <tr><td>64</td></tr> <tr><td>∅</td></tr> </table> | ∅ | 1∅ | 4 | 1∅ | 16 | 32 | 64 | ∅ | <p style="text-align: center;">GDU "I"</p>  | <table border="1"> <tr><td>124</td></tr> <tr><td>124</td></tr> <tr><td>124</td></tr> <tr><td>124</td></tr> <tr><td>56</td></tr> <tr><td>16</td></tr> <tr><td>16</td></tr> <tr><td>124</td></tr> </table> | 124 | 124 | 124 | 124 | 56 | 16 | 16 | 124 | |
| ∅ | | | | | | | | | | | | | | | | | | | | |
| 1∅ | | | | | | | | | | | | | | | | | | | | |
| 4 | | | | | | | | | | | | | | | | | | | | |
| 1∅ | | | | | | | | | | | | | | | | | | | | |
| 16 | | | | | | | | | | | | | | | | | | | | |
| 32 | | | | | | | | | | | | | | | | | | | | |
| 64 | | | | | | | | | | | | | | | | | | | | |
| ∅ | | | | | | | | | | | | | | | | | | | | |
| 124 | | | | | | | | | | | | | | | | | | | | |
| 124 | | | | | | | | | | | | | | | | | | | | |
| 124 | | | | | | | | | | | | | | | | | | | | |
| 124 | | | | | | | | | | | | | | | | | | | | |
| 56 | | | | | | | | | | | | | | | | | | | | |
| 16 | | | | | | | | | | | | | | | | | | | | |
| 16 | | | | | | | | | | | | | | | | | | | | |
| 124 | | | | | | | | | | | | | | | | | | | | |
| <p style="text-align: center;">GDU "E"</p>  | <table border="1"> <tr><td>∅</td></tr> <tr><td>24</td></tr> <tr><td>6∅</td></tr> <tr><td>6∅</td></tr> <tr><td>24</td></tr> <tr><td>6∅</td></tr> <tr><td>6∅</td></tr> <tr><td>∅</td></tr> </table> | ∅ | 24 | 6∅ | 6∅ | 24 | 6∅ | 6∅ | ∅ | <p style="text-align: center;">GDU "J"</p>  | <table border="1"> <tr><td>127</td></tr> <tr><td>119</td></tr> <tr><td>119</td></tr> <tr><td>65</td></tr> <tr><td>119</td></tr> <tr><td>119</td></tr> <tr><td>54</td></tr> <tr><td>28</td></tr> </table> | 127 | 119 | 119 | 65 | 119 | 119 | 54 | 28 | |
| ∅ | | | | | | | | | | | | | | | | | | | | |
| 24 | | | | | | | | | | | | | | | | | | | | |
| 6∅ | | | | | | | | | | | | | | | | | | | | |
| 6∅ | | | | | | | | | | | | | | | | | | | | |
| 24 | | | | | | | | | | | | | | | | | | | | |
| 6∅ | | | | | | | | | | | | | | | | | | | | |
| 6∅ | | | | | | | | | | | | | | | | | | | | |
| ∅ | | | | | | | | | | | | | | | | | | | | |
| 127 | | | | | | | | | | | | | | | | | | | | |
| 119 | | | | | | | | | | | | | | | | | | | | |
| 119 | | | | | | | | | | | | | | | | | | | | |
| 65 | | | | | | | | | | | | | | | | | | | | |
| 119 | | | | | | | | | | | | | | | | | | | | |
| 119 | | | | | | | | | | | | | | | | | | | | |
| 54 | | | | | | | | | | | | | | | | | | | | |
| 28 | | | | | | | | | | | | | | | | | | | | |

«GDU» del Programa «Palitroque y los Trogloditos».

| | | | | | |
|-----------|---|-----------|---|-----------|---|
| | bucle esperando que se pulse la tecla (C) (continuar). | 7120 | : Idem. (escudo protector). | 8190 | : Inicializa la posición de origen de (Palitroque). |
| 600 | : Borrado de los (Troglocitos). | 7130 | : Idem. (corona real). | 8200 | : Visualiza a (Palitroque). |
| 610 | : Incremento de la posición de los (Troglocitos). | 7140 | : Idem. (néctar de los dioses). | 8202-8204 | : Inicialización de las variables. |
| 620-630 | : Calcula si el próximo movimiento de los (Troglocitos) debe ser a la derecha o a la izquierda. | 7160 | : Aumenta en una unidad el contador de objetos recogidos. | 8220 | : Comienzo de la subrutina salto. |
| 640 | : Salto a la visualización de la nueva posición de los (Troglocitos). | 7200 | : Comienzo de la subrutina subir. | 8222-8430 | : Comprueba si durante el salto (Palitroque) choca con el muro, es atrapado por algún (Troglocito) o recoge alguna llave. |
| 1000 | : Fin del juego. | 7210-7212 | : Comprueba si (Palitroque) se encuentra en la escalera. | 8500 | : Comienzo de la subrutina de recogida de llaves. |
| 1002-1040 | : Efectos de color en el borde de la pantalla. | 7230 | : Comprueba si al salir de la escalera tropieza con un (Troglocito). | 8510-8520 | : Comprueban que se recoge la llave del piso inferior |
| 1050 | : Borrado de la pantalla. | 7232 | : Incrementa el valor de la variable que almacena el número de peldaños subidos. | 8530-8540 | : Idem. del piso primero. |
| 1060 | : Comprueba por que motivo se ha terminado el juego. | 7240-7260 | : Borra y visualiza la nueva posición de (Palitroque) en la escalera. | 8550-8560 | : Idem. del piso segundo. |
| 1070 | : Mensaje en caso de haberse quedado (Palitroque) sin vidas. | 7270 | : Comprueba si (Palitroque) ha terminado de subir la escalera. | 8570-8580 | : Idem. del piso superior. |
| 1090 | : Mensaje en caso de haberse completado la misión. | 8000 | : Comienzo de la subrutina que dibuja la escalera. | 8600 | : Parpadeo de la cerradura del piso inferior. |
| 1100-1130 | : Comprueba si se desea jugar otra vez o no. | 8010 | : Dibuja la escalera a partir de las coordenadas (x) e (y). | 8610 | : Idem. del piso primero. |
| 7000 | : Comienzo de la subrutina (detecta). | 8100 | : Comienzo de la subrutina (atrapa). | 8620 | : Idem. del piso segundo. |
| 7010 | : Comprueba si la nueva posición está libre o es una escalera. | 8110-8140 | : Parpadeo de (Palitroque) al chocar con la pared o al ser atrapado por algún (Troglocito). | 8630 | : Idem. del piso superior. |
| 7022 | : Comprueba si la puerta de la habitación está abierta. | 8150 | : Desaparición del (Palitroque). | 9000 | : Comienzo de la subrutina que genera los gráficos. |
| 7024 | : Llamada a la subrutina de coger objetos. | 8152-8160 | : Borrado de una vida en el contador de la parte inferior de la pantalla. | 9002-9030 | : Bucle para la lectura y generación de los GDU. |
| 7030-7050 | : Borra la antigua posición de (Palitroque) y visualiza la nueva. | 8170 | : Decremento de una vida. | 9040-9130 | : Tabla con los datos de los GDU. |
| 7100 | : Comienzo de la subrutina de coger objetos. | 8180 | : Comprueba si (Palitroque) se ha quedado sin vidas. | 9200 | : Subrutina (Carátula e Instrucciones). |
| 7110 | : Comprueba si el objeto cogido es la (espada glo- | | | 9210-9230 | : Recuadro de la pantalla. |
| | | | | 9240-9260 | : Mensajes de pantalla. |
| | | | | 9270-9300 | : Comprueba si se desea visualizar las instrucciones. |
| | | | | 9320-9360 | : Primera pantalla de información. |
| | | | | 9380-9490 | : Segunda pantalla de información. |
| | | | | 9510-9590 | : Última pantalla de información. |

SONIDO

El Spectrum tiene capacidad para producir una amplia variedad de notas musicales, con el manejo de una sola instrucción.

Los sonidos son escuchados a través de un altavoz interno, por lo tanto, el volumen es relativamente bajo, aunque puede mejorarse con el empleo de ciertos *periféricos*.

En un ordenador personal esta característica es muy apreciada, ya que da vida a ciertos programas que en un principio podrían parecernos «sosos».

Introduzca en el programa del capítulo anterior, «Palitroque y los trogloditos», las siguientes instrucciones y observe las diferencias:

a) Movimiento de los Trogloditos.

```
511 BEEP 0.01, -12
602 BEEP 0.01,, 0
```

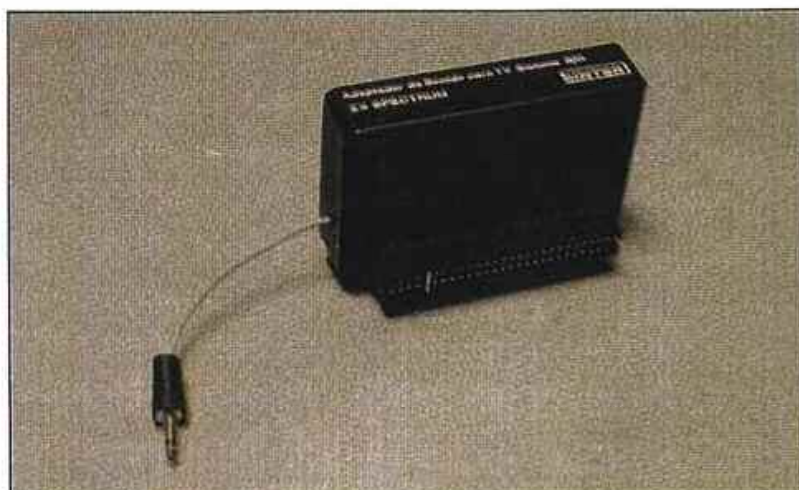
b) Pérdida de una vida.

```
8122 BEEP 0.01, 12
8132 BEEP 0.01, 24
```

c) Recogida de una llave. Antes de la sentencia «RETURN» de las líneas de programa.

```
8600
8610
8620
8630
```

incluir



Amplificador de sonido.

| VARIABLE | DIRECCION | VALOR INICIAL |
|----------|-----------|---------------|
| RASP | 23608 | 64 |
| PIP | 23609 | 0 |

RASP: Zumbador de alarma
PIP : Chasquido del teclado

Tabla I.

```
BEEP 0.01, 48
```

d) Fin de juego.

```
1012 BEEP 0.01, z
1014 BEEP 0.01, n
```

```
BEEP
```

Acceso al teclado

LN



MODO E

BEEP

SYMBOL SHIFT

PROGRAMA 1

```

10 REM *****
*   *
*   * CURSO/BASIC *
*   *
*   * *****
*   *
*   *   NAVIDAD   *
*   *
*   * *****
20 FOR c=-12 TO 0 STEP 12
30 READ nota,duracion
32 BORDER 7
40 IF NOT nota THEN RESTORE :
PAUSE 50: GO TO 70
50 BEEP duracion/3,nota+c
52 BORDER 2
60 GO TO 30
70 NEXT c
1000 REM TABLA DE DATOS

```

```

1010 DATA 16,4,17,1,16,1,15,1
1020 DATA 16,1,17,4,18,1,19,3
1030 DATA 21,2,23,1,24,1,26,1
1040 DATA 24,1,23,1,21,1,19,6
1050 DATA 12,1,14,1,16,2,16,2
1060 DATA 16,1,21,2,19,1,12,2
1070 DATA 12,2,12,1,19,2,17,1
1080 DATA 16,4,17,1,16,1,14,1
1090 DATA 12,1,14,8,16,4,17,1
1100 DATA 16,1,15,1,16,1,17,4
1110 DATA 18,1,19,3,21,2,23,1
1120 DATA 24,1,26,1,24,1,23,1
1130 DATA 21,1,19,6,12,1,14,1
1140 DATA 16,2,16,2,16,1,21,2
1150 DATA 19,1,24,6,12,1,14,1
1160 DATA 16,2,16,2,21,1,5,11,0,
5
1170 DATA 11,1,11,1,12,7,0,0

```

PROGRAMA 2

```

10 REM *****
*   *
*   * CURSO/BASIC *
*   *
*   * *****
*   *
*   *   DIAFONICA *
*   *
*   * *****
20 POKE 23658,0
22 GO SUB 1000
30 LET duracion=0.25
40 LET tono=0
50 LET tono=(2 AND INKEY$="s")
+(4 AND INKEY$="d")+(5 AND INKEY

```

```

$="f")+(7 AND INKEY$="g")+(9 AND
INKEY$="h")+(11 AND INKEY$="j")
60 IF tono=0 AND INKEY$<>"a" T
HEN GO TO 50
70 BEEP duracion,tono
80 GO TO 50
1000 REM DEFINICIONES
1010 PRINT AT 0,6;"ESCALA DIAFON
ICA"
1020 PRINT AT 10,2;"do re mi
fa sol la si"
1030 PRINT AT 12,3;"a s d
f g h j"
1040 PRINT #0;AT 1,2;"Toca tu me
lodia preferida"
1050 RETURN

```

Definición

La sentencia «BEEP» activa el altavoz, de manera que suene a una frecuencia proporcional al valor de *semitono* introducido y durante un tiempo determinado.

Su estructura general es:

| SENTENCIA | ARGUMENTO |
|-----------|----------------|
| BEEP | duración, tono |

Ejemplos:

- BEEP 1, 3
- BEEP 0.5, -2
- BEEP 10, nota
- BEEP j, a*2

El parámetro «duración» especifica el tiempo en segundos que dura el tono.

Este valor puede estar comprendido dentro de los márgenes: 0 a 10.

El valor del tono viene expresado en unidades de semitono, teniendo éstas una relación con el «do» central de un piano.

El valor del semitono es positivo si está por encima del «do» central y es negativo si está por debajo.

Los valores que pueda tomar el parámetro «tono», deben estar comprendidos dentro de los márgenes: -60 a 69.8.

Si se especifican otros valores, tanto por la duración como por el tono, se produce el error:

B Integer out of range

Introduzca el siguiente programa que le proporcionará todos los valores tonales que es capaz de reproducir el

Spectrum; primero en una escala creciente y posteriormente, en una decreciente:

```

10 REM *****
*   *
*   * ESCALAS TONALES *
*   *
*   * *****
20 FOR n=-60 TO 69
30 BEEP .05,n
40 NEXT n
50 PAUSE 50
60 FOR n=69 TO -60 STEP -1
70 BEEP .05,n
80 NEXT n

```

Durante la ejecución de una sentencia «BEEP» no puede realizarse «BREAK», ya que el sistema operativo no explorará dicha tecla hasta que no termine de ejecutarse; compuébelo con el comando directo:

BEEP 10, 0


```

300 DATA re5,negra,re,blanca,so
l,corchea,sol,corchea,sol,corche
a,sol,negra,sol,corchea,sol,corc
hea
310 DATA sol,blanca,la,corchea,
la,corchea,la,corchea,la,negra,l
a,corchea,la,corchea,la5,negra,l
a,blanca
320 DATA re,corchea,re,corchea,
re,corchea,re,corchea,ra,corchea,d
o,corchea,re,blanca
330 DATA re,corchea,re,corchea,
re,corchea,fa,negra,re5,corchea,
re,corchea,re5,negra,re,blanca
340 DATA do,corchea,re,corchea,
re5,corchea,la,negra,sol,corchea
,fa5,corchea,sol,negra,re,negra
350 DATA la5,corchea,la5,corche
a,la5,negra,la,blanca,sol,corche
a,fa5,corchea,la,negra,sol,blanc
a
360 DATA re,corchea,sol,corchea
,sol,corchea,sol,negra,sol,corch
ea,si,corchea,do+12,negra,do+12,
blanca
370 DATA si,corchea,do+12,corch

```

```

ea,do+12,corchea,si,corchea,do+1
2,corchea,sol,corchea,la5,semico
rchea,sol5,corchea,sol,blanca
380 DATA fa5,corchea,fa5,corche
a,fa5,corchea,fa5,negra,mi,corch
ea,re,corchea,mi,negra,re,blanca
390 DATA fa5,corchea,fa5,corche
a,fa5,corchea,mi,corchea,mi,corc
hea,re,corchea,mi,negra,re,blanc
a
400 DATA re,corchea,sol,corchea
,sol,corchea,sol,negra,sol,corch
ea,si,corchea,do+12,negra,do+12,
blanca
410 DATA si,corchea,do+12,corch
ea,do+12,corchea,si,corchea,do+1
2,corchea,sol,corchea,la5,semico
rchea,sol5,corchea,sol,blanca
420 DATA fa5,corchea,fa5,corche
a,fa5,corchea,fa5,negra,mi,corch
ea,re,corchea,mi,negra,re,blanca
430 DATA do,corchea,re,corchea,
mi,corchea,la,negra,sol,corchea,
fa5,corchea,la,negra,sol,blanca
440 DATA 0,0

```

| NOTAS | | VALOR DE LOS SEMITONOS | | | | | | | | | | |
|-------|------|------------------------|-----|-----|-----|-----|----|----|----|----|----|----|
| C | DO | -60 | -48 | -36 | -24 | -12 | 0 | 12 | 24 | 36 | 48 | 60 |
| C# | DO# | -59 | -47 | -35 | -23 | -11 | 1 | 13 | 25 | 37 | 49 | 61 |
| D | RE | -58 | -46 | -34 | -22 | -10 | 2 | 14 | 26 | 38 | 50 | 62 |
| D# | RE# | -57 | -45 | -33 | -21 | -9 | 3 | 15 | 27 | 39 | 51 | 63 |
| E | MI | -56 | -44 | -32 | -20 | -8 | 4 | 16 | 28 | 40 | 52 | 64 |
| F | FA | -55 | -43 | -31 | -19 | -7 | 5 | 17 | 29 | 41 | 53 | 65 |
| F# | FA# | -54 | -42 | -30 | -18 | -6 | 6 | 18 | 30 | 42 | 54 | 66 |
| G | SOL | -53 | -41 | -29 | -17 | -5 | 7 | 19 | 31 | 43 | 55 | 67 |
| G# | SOL# | -52 | -40 | -28 | -16 | -4 | 8 | 20 | 32 | 44 | 56 | 68 |
| A | LA | -51 | -39 | -27 | -15 | -3 | 9 | 21 | 33 | 45 | 57 | 69 |
| A# | LA# | -50 | -38 | -26 | -14 | -2 | 10 | 22 | 34 | 46 | 58 | |
| B | SI | -49 | -37 | -25 | -13 | -1 | 11 | 23 | 35 | 47 | 59 | |

↑ OCTAVA CENTRAL

SOSTENIDO

Correspondencia entre «notas» y «semitonos» de las diferentes octavas.

Nociones musicales

En este párrafo se van a explicar una serie de conceptos básicos útiles para aquellas personas que deseen programar una melodía en el

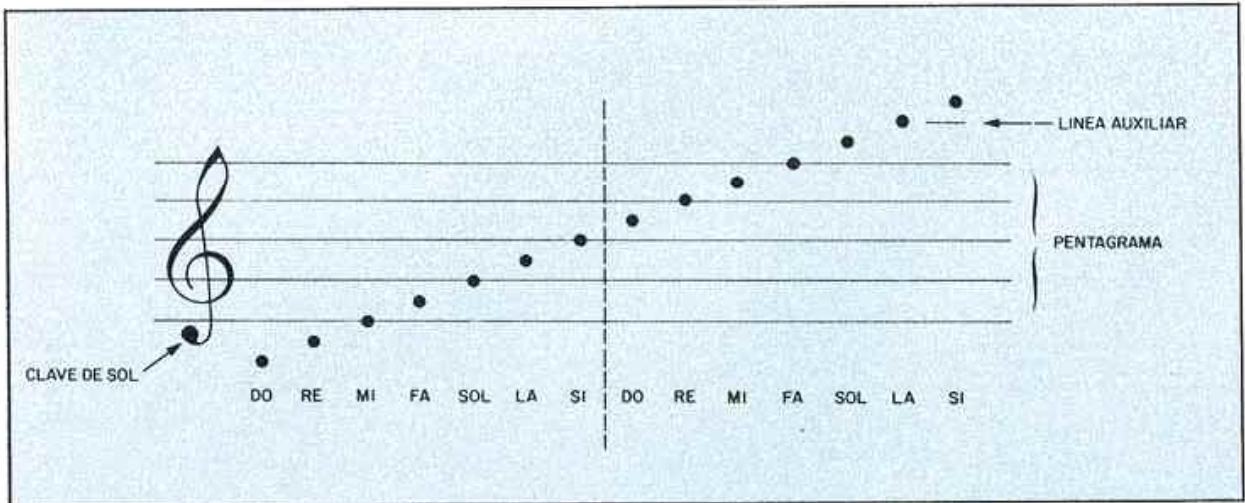
Spectrum.

Tono

El tono es el grado de elevación del sonido, conocido también como *altura tonal*. Siendo el tono más alto cuan-

do mayor es su *frecuencia* (tono agudo) y más bajo cuando es menor (tono grave).

En una escala *diatónica*, las notas musicales identificadoras del tono son:



Escala diafónica de dos octavas, en clave de sol.

| |
|-----|
| DO |
| RE |
| MI |
| FA |
| SOL |
| LA |
| SI |

En cambio en la escala cromática son doce:

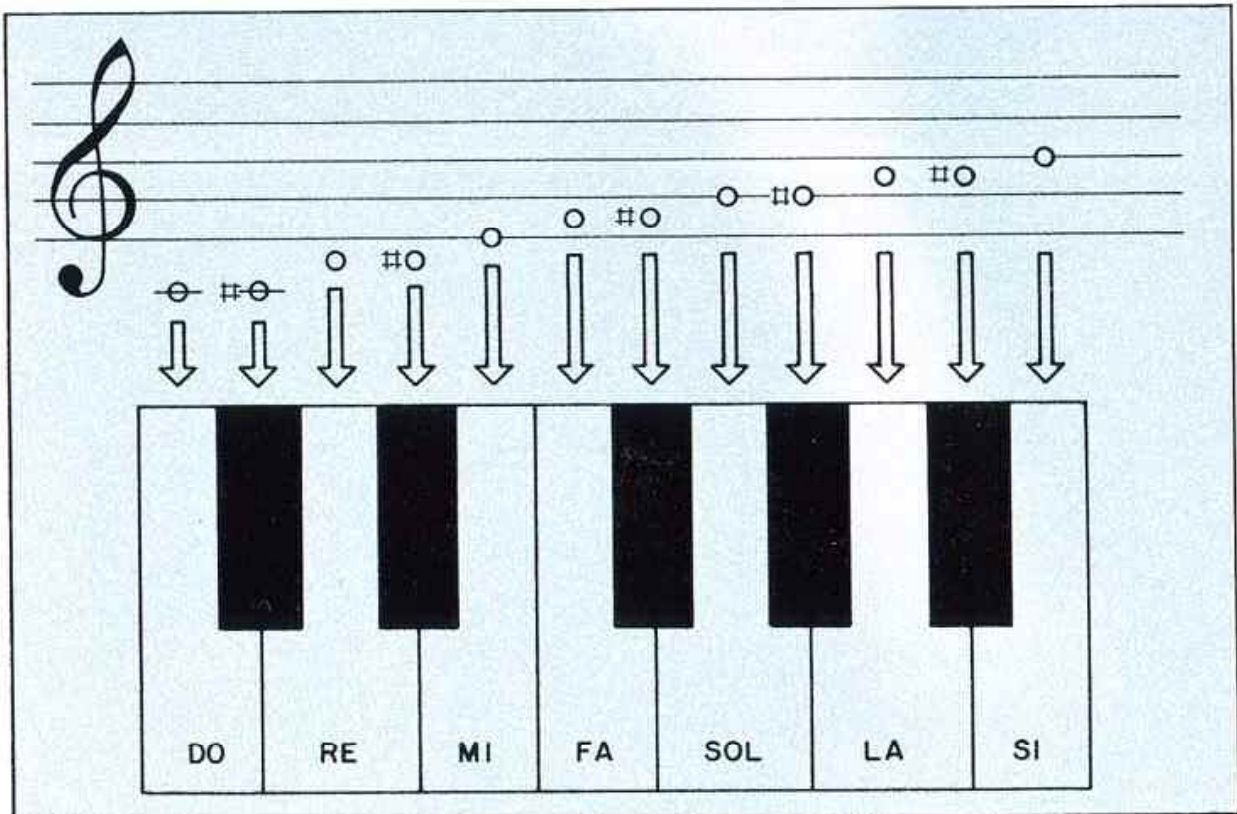
| |
|-------|
| DO # |
| DO # |
| RE # |
| RE # |
| MI # |
| FA # |
| FA # |
| SOL # |
| SOL # |
| LA # |
| LA # |
| SI # |

El símbolo # es el identificador del *sostenido*.

En un piano las teclas negras son los correspondientes a los sostenidos.

La unidad máxima de altura tonal es el *semitono*. Un semitono es el resultado de dividir una *octava* en doce partes iguales.

Entre dos notas consecuti-



Escala cromática de una octava y su correspondencia con un piano.

vas hay dos semitonos, excepto entre «MI» - «FA» y «SI» - «DO» que en ambos casos hay un solo semitono.

En un pentagrama cada nota viene identificada por la posición que ocupa dentro del mismo.

El programa número «2» convierte el teclado del Spectrum en un piano con escala diatónica y el número «3», en un cromático. Las teclas que deben pulsarse en cada caso están contenidas en las instrucciones del propio programa.



Curso de teoría de la música.

PROGRAMA 5

```
10 REM *****
*          *
*  CURSO/BASIC *
*          *
*****
*          *
*  EFECTO 1   *
*          *
*****
20 FOR i=30 TO 0 STEP -1
30 BEEP .005,i
40 NEXT i
```

PROGRAMA 6

```
10 REM *****
*          *
*  CURSO/BASIC *
*          *
*****
*          *
*  EFECTO 2   *
*          *
*****
20 FOR a=30 TO 0 STEP -1
30 BEEP .05,20-a
40 NEXT a
50 BEEP .5,-10
```

PROGRAMA 7

```
10 REM *****
*          *
*  CURSO/BASIC *
*          *
*****
*          *
*  EFECTO 3   *
*          *
*****
20 FOR a=1 TO 25
30 FOR b=1 TO 10
40 BEEP 0.05,INT (RND*b) -12
50 BEEP 0.05,INT (RND*b) +12
60 NEXT b
70 NEXT a
```

PROGRAMA 8

```
10 REM *****
*          *
*  CURSO/BASIC *
*          *
*****
*          *
*  EFECTO 4   *
*          *
*****
20 FOR a=1 TO 10
30 FOR b=10 TO 1 STEP -1
40 BEEP 0.05,a
50 BEEP 0.05,b
60 NEXT b
70 NEXT a
```

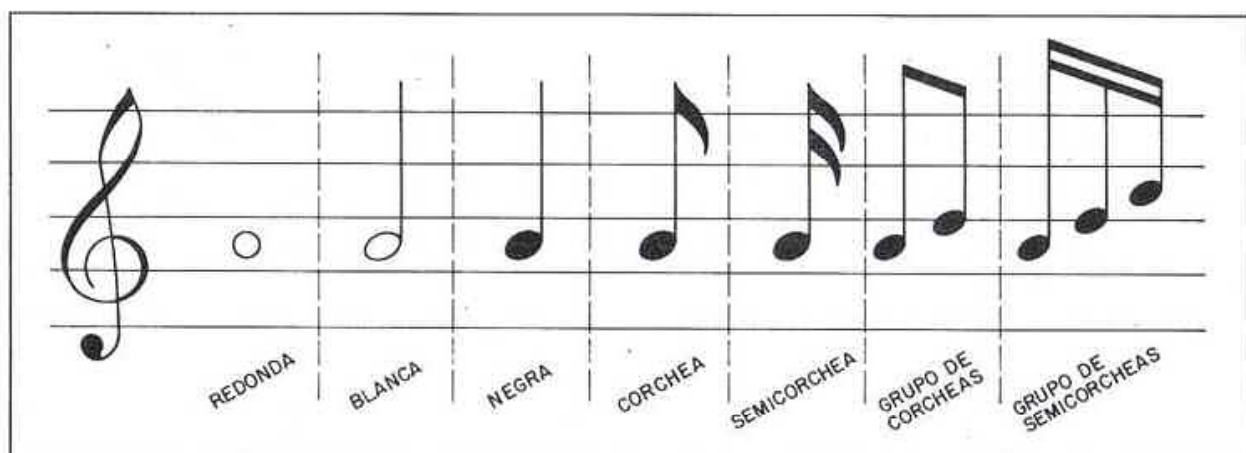
Duración

La duración de la nota viene identificada por unos símbolos característicos. Los más utilizados son:

REDONDA
BLANCA
NEGRA
CORCHEA
SEMICORCHEA

La que más dura, es la redonda, la blanca es la mitad que la redonda, la negra es la mitad que la blanca y así sucesivamente.

Por tanto, en cuanto a dura-



Simbología utilizada en la duración relativa de las notas.

INDICATIVO DEL COMPAS

COMPAS

DOS POR CUATRO

TRES POR CUATRO

CUATRO POR CUATRO

Diagrama que muestra tres ejemplos de compases en una línea de pentagrama. El primer ejemplo muestra un compás de 2/4 con una barra que abarca los primeros dos compases etiquetados como 'DOS POR CUATRO'. El segundo ejemplo muestra un compás de 3/4 con una barra que abarca los primeros tres compases etiquetados como 'TRES POR CUATRO'. El tercer ejemplo muestra un compás de 4/4 con una barra que abarca los primeros cuatro compases etiquetados como 'CUATRO POR CUATRO'. Cada ejemplo incluye una línea de pentagrama con una clave de sol y una línea de compases con una barra que indica la duración de los compases.

Ejemplos típicos de compases.

cassette en la salida «MIC» del Spectrum podrá grabar cualquier melodía que genere el ordenador, ya que la señal que activa el altavoz también

se encuentra presente en dicho conector.

También es posible conectar un amplificador a dicha salida, con lo que aumentará

el volumen de la melodía. Si conecta unos auriculares podrá escuchar simultáneamente los sonidos que el ordenador genere, aunque hay

que tener en cuenta que el altavoz interno no se desconecta.

Periféricos

Hay ciertos periféricos o dispositivos que podemos conectar al ordenador y que aumentan las posibilidades sonoras de éste.

– Amplificador de sonido que permite escuchar el sonido generado por el ordenador, a través del altavoz de nuestro receptor de T.V.; de esta manera se dispone de un control de volumen a gusto del usuario.

– Sintetizador de voz que permite al ordenador «hablar» de la misma manera que lo haría un robot. Los sonidos a

emitir deben ser asignados a cierta variable de cadena.

Software musical

Aunque no abundantes, existen en el mercado algunos programas musicales; unos son del tipo didáctico y permiten seguir un curso de «teoría de la música» a través del ordenador. Otro, conocido por el nombre comercial de «Melodian», permite asistir a todas las fases de una composición musical, permitiendo:

- Escribir partituras
- Ejecutarlas.
- Imprimirlas.
- Almacenarlas en cinta.

El programa viene acompañado de una plantilla especial de cartulina que convierte al Spectrum en un teclado musical simulado.

Efectos sonoros

Los programas «5» a «8» proporcionan diversos efectos sonoros que pueden ser utilizados en nuestros programas.

El programa número «9» proporciona un efecto de «TREMOLLO»; debe ser utilizado con la tabla de datos del programa número «1», (Navidad).

SENTENCIAS DE GRABACION Y CARGA

El capítulo número tres, "Almacenamiento de programas", fue una breve introducción a la grabación y recuperación de programas BASIC; este va a ser más completo, ya que analizará todos los posibles argumentos de las sentencias:

SAVE
VERIFY
LOAD
MERGE

SAVE

Acceso al teclado

RESTORE



MODO **K**

Definición

La sentencia "SAVE" permite almacenar en cinta cualquier programa escrito en lenguaje BASIC o en "Código Máquina", así como las matrices.

Su estructura general es:

| SENTENCIA | ARGUMENTO |
|-----------|---------------|
| SAVE | "nombre" tipo |

Ejemplo:

- SAVE "carg"
- SAVE "CR" CODE 204, 20
- SAVE "T01/2" LINE 40
- SAVE "Pat" DATA Q ()

El campo "nombre" debe ir entre comillas y no superar la cantidad de diez caracteres; estos pueden ser letras, números o símbolos.

Ejemplos:

"Tragon"
"EMULADOR"
"1234*a"
"\$*/?!"

A diferencia de las variables, un nombre de programa escrito en minúsculas es distinto de uno en mayúsculas; por tanto los nombres:

"COMPILADOR"
"Compilador"
"ComPiLaDor"

se refieren a distintos programas.

El campo "tipo" indica qué clase de grabación desea realizar; cuando este se omite, el ordenador interpreta que es la zona de memoria donde está almacenado nuestro programa "BASIC", la que se desea grabar.

Ejemplo:

SAVE "ON/OF"

De esta manera el programa BASIC denominado "ON/

OF" quedará almacenado en cinta.

Una forma particular de grabar un programa BASIC, es indicar el número de línea donde se desea que comience la *autoejecución*, así, cuando se realice una carga posterior, no será necesario pulsar "RUN" n, donde "n" es el número de línea.

Ejemplo:

SAVE "ON/OFF" LINE 1000

El programa "ON/OFF" se autoejecutará a partir de la línea 1000. Si se desea que un programa se autoejecute a partir de la primera sentencia, puede grabarse con LINE 1.

Grabación de código máquina

Para grabar C/M (Código Máquina), es decir, los programas escritos en el lenguaje interno del ordenador, se utiliza el tipo "CODE", donde es necesario indicar la dirección de memoria a partir de la cual se desea grabar y la longitud del programa expresada en bytes.

Ejemplo:

SAVE "S.O." CODE 0, 16384

de esta forma se almacena en cinta una copia del Sistema operativo del Spectrum, que comienza en la dirección "0"

| CINTA: MICROHOBBY-4 | | CARA: A | |
|---------------------|------|------------|----------------|
| N | Tipo | Nombre | Comienzo Long. |
| 1 | Prog | MICROHOBBY | 1 545 |
| 2 | Byte | CASSETTE | 32768 6912 |
| 3 | Prog | GARDEN | 1 237 |
| 4 | Byte | garden\$ | 32768 6912 |
| 5 | Byte | GARDEN1 | 65368 168 |
| 6 | Byte | GARDEN2 | 32492 40 |
| 7 | Prog | GARDEN3 | 1 8244 |
| 8 | Prog | SOLADOR | 1 20209 |
| 9 | Prog | CAZA | 1 5312 |
| 10 | Prog | RULETA | 1 11418 |

Presentación del programa «Listador»

y tiene una longitud de 16384 bytes o 16 Kbytes, por que:

1 Kbyte = 1024 bytes

Un caso particular, es la grabación de los datos de la zona de memoria correspon-

diente a la pantalla; ésta es la forma más usual de grabar las carátulas de los programas comerciales, recuerde la presentación de la cinta demostrativa del Spectrum 16 ó 48 K, "HORIZONTES".

Primero deberá dibujar en

pantalla el gráfico o texto que desee a base de "PLOT", "DRAW", "CIRCLE" o "PRINT". Cuando esté confeccionado, se podrá grabar en cinta de la forma:

SAVE "nombre" SCREEN\$

La palabra clave "SCREEN\$" es equivalente en este caso a:

CODE 16384, 6912

ya que la dirección de comienzo de la zona de pantalla es 16384 y tiene una longitud de 6912 bytes. Puede utilizar cualquiera de las dos fórmulas posibles.

Para almacenar la zona de memoria correspondiente a los GDU, puede utilizar, como

PROGRAMA 1

```

10 REM *****
*          *
*  CURSO/BASIC *
*          *
*****
*  LISTADOR *
*          *
*****

20 REM Domingo Gomez © 1984
30 CLEAR 29999
40 BORDER 4: PAPER 4: INK 0: C
LS
50 LET contador=1
60 RESTORE GO SUB 500
68 REM ENTRADA DATOS
70 INPUT "Nombre de la cinta >"; LINE t$
80 IF LEN t$ > 15 THEN LET t$=t$
(TO 15)
90 FOR n=1 TO (15-LEN t$)
100 LET t$=t$+CHR$ 32
110 NEXT n
120 INPUT "Cara >"; LINE f$
130 IF LEN f$ > 1 THEN LET f$=f$(
1)
140 IF f$="" THEN LET f$=CHR$ 3
2
148 REM CHECERA
150 LET d$=CHR$ 138+"CINTA:"+CHR$
R$ 32+t$+CHR$ 32+"CARA:"+CHR$ 32
+f$+CHR$ 133
160 LET i$=""
170 LET d$=d$+d$
180 LET i$=""
190 LET s$=CHR$ 138+"N Tipo No
mbre Comienzo Long."+CHR$ 133
200 LET d$=d$+i$+s$+i$
210 FOR n=2 TO 3
220 PRINT #n; d$
230 NEXT n
238 REM LISTADOR
240 LET a$=""
250 LET j$=STR$ contador
260 IF LEN j$ < 2 THEN LET j$=CHR$
$ 32+j$

```

```

270 RANDOMIZE USR 30000
280 FOR n=30021 TO 30030
290 IF PEEK n > 31 AND PEEK n < 144
THEN LET a$=a$+CHR$ PEEK n
300 IF PEEK n < 32 OR PEEK n > 143
THEN LET a$=a$+CHR$ 63
310 NEXT n
320 LET tipo=PEEK 30020
330 GO SUB 1000
340 LET w$=j$+CHR$ 32+t$+CHR$ 3
2+a$+CHR$ 32+c$+CHR$ 32+l$
350 PRINT w$: LPRINT w$
360 RANDOMIZE USR 30000
370 LET contador=contador+1
380 IF contador > 16 THEN PRINT A
T 1,0,d$: RANDOMIZE USR 3190: PR
INT AT 19,0
410 IF contador > 99 THEN LET con
tador=0
420 GO TO 240
499 STOP
500 REM Caratula C.M
502 FOR n=30000 TO 30013
510 READ dato
520 POKE n,dato
530 NEXT n
540 RETURN
550 REM Caratula C.M
560 DATA 62,0,221,33,68,117,17
570 DATA 17,0,55,205,86,5,201
1000 REM DATOS DEL PROGRAMA
1010 IF tipo=0 THEN LET t$="Prog"
1020 IF tipo=1 THEN LET t$="DatN"
1030 IF tipo=2 THEN LET t$="Dat$"
1040 IF tipo=3 THEN LET t$="Byte"
1050 LET c$=STR$ (PEEK 30033+256
+PEEK 30034): LET c$=""
LET c$=c$+(LEN c$-5) TO LEN c$
1060 IF VAL c$ > 9999 AND tipo < 3
THEN LET c$=""
1070 LET l$=STR$ (PEEK 30031+256
+PEEK 30032): LET l$=""
LET l$=l$+(LEN l$-5) TO LEN l$
1080 RETURN

```

```

DIRECTORIO  @  MICROHOBBY
          CARR. A

NOMBRE  MICROHOBBY

COMENTARIOS
Programas pertenecientes al nu
mero 4 de MICROHOBBY/CASSETTE

PROPIETARIO  R. P. R.

```

```

DIRECTORIO  @  MICROHOBBY

MENU DE OPCIONES

1  PRESENTACION.
2  DIRECTORIO.
3  EJECUCION.
4  INFORMACION.

```

```

DIRECTORIO  @  MICROHOBBY

INFORMACION

PROGRAMA  UVE ATAC

COMENTARIOS
Una aventura de Cosme y los al
ienigenas. Debe proteger el em
balse de sus ataques

PROGRAMADOR  PACO MARTIN

FECHA  MAYO 85          CONT. 120

```

```

DIRECTORIO  @  MICROHOBBY

CARGA DE PROGRAMAS

PROGRAMA  UVE ATAC

CONTADOR  120

PULSE LA TECLA "PLAY" DEL CA-
SSETTE Y A CONTINUACION PULSE
LA TECLA "ENTER".

```

| | | |
|----|------------|----|
| 1 | HECHIZO | 11 |
| 2 | HORMIGUERO | 12 |
| 3 | CARGADOR | 13 |
| 4 | OLYMPIA | 14 |
| 5 | SOLITARIO | 15 |
| 6 | UVE ATAC | 16 |
| 7 | CRETA | 17 |
| 8 | GUSANIN | 18 |
| 9 | SATURNO | 19 |
| 10 | GOMOKU | 20 |

TECLEE EL NUM. DE PROGRAMA. >

Diversas pantallas del programa «Directorio».

ya se vio en el capítulo co-
rrespondiente:

```
SAVE "nombre" CODE USR "a", 168
```

Grabación de matrices

Para la grabación de matri-
ces se utiliza el tipo "DATA".
Ejemplos:

```
SAVE "nombre" DATA X ( )
```

– Almacena en cinta el
contenido de la matriz numé-
rica "X".

```
SAVE "nombre" DATA N$ ( )
```

– Almacena en cinta el
contenido de la matriz de ca-
dena "N\$".

```
VERIFY
```

PROGRAMA 2

```

10 REM *****
* CURSO/BASIC *
* *****
* DIRECTORIO *
* *****

80 GO SUB 4200
90 BORDER 1: PAPER 1: INK 7: C
LS
100 REM PORTADA
105 GO SUB 5000
110 PRINT INVERSE 1; AT 3,12; "CA
RA"; PRINT " "; PRINT D$(21,1
1 TO 13)
120 PRINT INVERSE 1; AT 8,1; "NOM
BRE"; PRINT " "; PRINT D$(21,
1 TO 10)
130 PRINT INVERSE 1; AT 11,1; "CO
MENTARIOS"; PRINT
140 PRINT " "; D$(21,34 TO 63)
150 PRINT " "; D$(21,64 TO 93)
160 PRINT " "; D$(21,94 TO 123)
170 PRINT INVERSE 1; AT 18,1; "PR
OFIETARIO"; PRINT " "; PRINT
D$(21,14 TO 33)
180 PRINT #0; " PULSE ""ENTER""
PARA CONTINUAR."
190 PAUSE 0
200 LET a$=INKEY$
210 IF CODE a$ <> CHR$ 13 THEN BEEP 0.
2,-15: GO TO 190
215 REM OPCIONES
220 GO SUB 5000
240 PRINT AT 3,8; "MENU DE OPCIO
NES"
250 PRINT AT 4,8; "
260 PRINT INVERSE 1; AT 8,8; "1",
PRINT " PRESENTACION."
270 PRINT INVERSE 1; AT 11,8; "2"
PRINT " DIRECTORIO."
280 PRINT INVERSE 1; AT 14,8; "3"
PRINT " EJECUCION."
290 PRINT INVERSE 1; AT 17,8; "4"
PRINT " INFORMACION."
295 PRINT AT 21,1; "PULSE LA OPC
ION DESEADA."
300 REM SELECCION
305 PAUSE 0
308 PRINT AT 21,0; "
310 LET a$=INKEY$
320 IF a$="1" THEN PRINT FLASH
1; AT 8,11; "PRESENTACION."; PAUSE
75: GO SUB 5000: GO TO 110
330 IF a$="2" THEN PRINT FLASH
1; AT 11,11; "DIRECTORIO."; PAUSE
75: GO TO 1000
340 IF a$="3" THEN LET seleccio
n=3: PRINT FLASH 1; AT 14,11; "EJE
CUCION."; PAUSE 75: GO TO 2000
350 IF a$="4" THEN LET seleccio
n=4: PRINT FLASH 1; AT 17,11; "INF
ORMACION."; PAUSE 75: GO TO 2000
360 BEEP 0.2,-15: GO TO 295
1000 REM OPCION DIR
1010 GO SUB 4000
1020 PRINT AT 21,0; " PULSE ""ENT
ER"" PARA RETORNAR."
1030 PAUSE 0
1040 LET a$=INKEY$
1050 IF CODE a$ <> CHR$ 13 THEN BEEP 0.
2,-15: GO TO 1030
1060 GO TO 215
2000 REM OPCIONES 3 4 4
2010 GO SUB 4000
2020 GO SUB 4300
2030 LET flash=1
2050 GO SUB 4100
2060 PRINT AT 21,1; "PULSE ""ENTE
R"" O ""DELETE""
2065 PAUSE 0
2070 LET a$=INKEY$
2080 IF a$=CHR$ 12 THEN BEEP 0.0
5,20: LET flash=0: GO SUB 4100:
GO TO 2020
2090 IF a$=CHR$ 13 THEN GO TO 21
10
2100 BEEP 0.2,-15: GO TO 2065

```

```

2110 CLS : GO SUB 5000
2120 IF seleccion=3 THEN GO TO 2
500
2130 GO TO 3000
2500 REM OPCION LOAD
2510 PRINT AT 3,7; "CARGA DE PROG
RAMAS"
2520 PRINT AT 4,7; "
2530 PRINT INVERSE 1; AT 8,1; "PRO
GRAMA"; PRINT " "; D$(programa,
1 TO 10)
2540 PRINT INVERSE 1; AT 12,1; "CO
NTADOR"; PRINT " "; D$(programa
,11 TO 13)
2550 PRINT AT 16,1; "PULSE LA TEC
LA ""PLAY"" DEL CA"
2560 PRINT AT 18,1; "SSETTE Y A C
ONTINUACION PULSE"
2570 PRINT AT 20,1; "LA TECLA ""E
NTER""
2580 PAUSE 0
2590 LET a$=INKEY$
2600 IF a$ <> CHR$ 13 THEN BEEP 0.
2,-15: GO TO 2580
2610 BEEP 0.05,20
2615 CLS
2620 LOAD D$(programa,1 TO 10)
2999 STOP
3000 REM OPCION INFO
3010 PRINT AT 2,10; "INFORMACION"
3020 PRINT AT 3,10; "
3030 PRINT INVERSE 1; AT 7,1; "PRO
GRAMA"; PRINT " "; D$(programa,
1 TO 10)
3035 PRINT INVERSE 1; AT 10,1; "CO
MENTARIOS."
3040 PRINT AT 12,1; D$(programa,3
4 TO 63)
3050 PRINT AT 13,1; D$(programa,6
4 TO 93)
3060 PRINT AT 14,1; D$(programa,9
4 TO 123)
3070 PRINT INVERSE 1; AT 17,1; "PR
OGRAMADOR"; PRINT " "; D$(progr
ama,14 TO 25)
3080 PRINT INVERSE 1; AT 20,1; "FE
CHA"; PRINT " "; D$(programa,26
TO 33)
3090 PRINT INVERSE 1; AT 20,22; "C
ONT"; PRINT " "; D$(programa,11
TO 13)
3100 PRINT #0; " PULSE ""ENTER""
PARA RETORNAR."
3110 PAUSE 0
3120 LET a$=INKEY$
3130 IF a$ <> CHR$ 13 THEN BEEP 0.
2,-15: GO TO 3110
3140 CLS : GO TO 220
3999 STOP
4000 REM SUB DIR
4005 CLS : BEEP 0.05,20: LET J=-
1
4010 FOR I=1 TO 10
4020 IF I < 10 THEN PRINT INVERSE
1; AT I+J,2; I; PRINT " "; D$(I,1
TO 10); PRINT INVERSE 1; AT I+J,1
7; I+10; PRINT " "; D$(I+10,1 TO
10)
4030 IF I=10 THEN PRINT INVERSE
1; AT I+J,1; I; PRINT " "; D$(I,1
TO 10); PRINT INVERSE 1; AT I+J,1
7; I+10; PRINT " "; D$(I+10,1 TO
10)
4035 LET J=J+1
4040 NEXT I
4050 RETURN
4100 REM SUB FLASH
4120 IF programa < 11 THEN LET x=4
: LET y=(programa*2)-2: GO TO 41
40
4130 LET x=20: LET y=(programa-1
0)*2-2
4140 PRINT FLASH flash; AT y,x; D$
(programa,1 TO 10)
4150 RETURN
4200 REM SUB ULTIMO
4210 FOR I=1 TO 20
4220 IF D$(I,1 TO 10)="
" THEN LET ultimo=I-1: GO TO 42
50
4230 LET ultimo=I

```

```

4240 NEXT I
4250 RETURN
4300 REM SUB. PROGRAMA
4310 PRINT AT 21,1;"TECLEE EL NU
M. DE PROGRAMA."; FLASH 1;">";
PRINT ;
4320 PAUSE 0
4330 LET a$=INKEY$
4340 IF a$<"1" OR a$>"9" THEN BE
EP 0.2,-15: GO TO 4320
4350 PRINT AT 21,30;a$;: BEEP 0.
05,20
4360 LET programa=VAL (a$)
4380 PAUSE 0
4390 LET a$=INKEY$
4400 IF a$=CHR$ 13 THEN GO TO 44
40
4410 IF a$<"0" OR a$>"9" THEN BE
EP 0.2,-15: GO TO 4380
4420 PRINT ;a$;: BEEP 0.05,20

```

```

4430 LET programa=programa*10+VA
L (a$)
4435 PAUSE 0
4440 IF programa>ultimo THEN PRI
NT AT 21,30;" ";: GO TO 4320
4450 RETURN
5000 REM SUB. LOGO
5005 BEEP 0.05,20: CLS
5010 PRINT INVERSE 1;" DIRECTORI
O
© MICROHOBBY "
5020 RETURN
6000 REM CARGA DE DATOS
6010 LOAD "DIR" DATA D$()
6012 CLS
6020 PRINT FLASH 1;AT 10,10;"PAR
E LA CINTA"
6030 PRINT #0;"PULSE UNA TECLA P
ARA CONTINUAR.": PAUSE 0
6040 GO TO 80

```

Acceso al teclado

INT



MODO **E**
SYMBOL
SHIFT

VERIFY

Definición

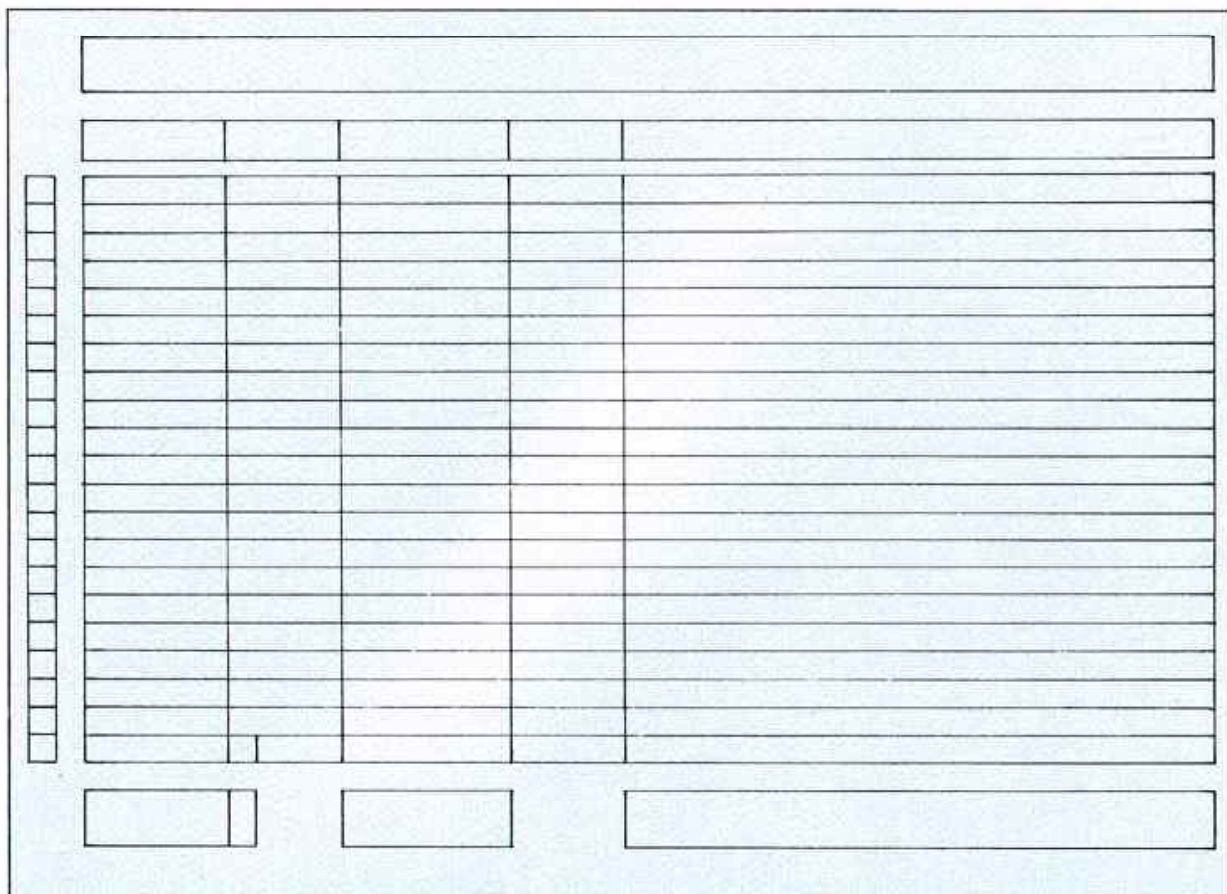
La sentencia "VERIFY" comprueba que el programa o datos se han grabado correctamente, para ello compara la información de la cinta con la resistencia en memoria.

Su estructura general es:

| SENTENCIA | ARGUMENTO |
|-----------|---------------|
| VERIFY | "nombre" tipo |

Ejemplos:

- VERIFY "CAO CAO"
- VERIFY "HOP" CODE
- VERIFY "S" DATA K()



Organización de la tabla de datos utilizada en el programa DIRECTORIO.

– VERIFY "T" DATA M\$ ()

Los campos "nombre" y "tipo" siguen las mismas reglas que en el caso de "SAVE".

a) Verificación de un programa BASIC.

```
VERIFY "nombre"
```

b) Verificación de un programa en C/M o de bytes. Pueden utilizarse cualquiera de estas opciones:

```
VERIFY "nombre" CODE comienzo, longitud
VERIFY "nombre" CODE comienzo
VERIFY "nombre" CODE
```

Existen pequeñas diferencias en cuanto a la forma de verificar de cada una de ellas, siendo la más exacta la primera.

c) Verificación de una matriz numérica.

```
VERIFY "nombre" DATA letra ( )
```

d) Verificación de una matriz de cadena.

```
VERIFY "nombre" DATA letra$ ( )
```

Los programas almacenados como:

```
SAVE "nombre" SCREEN$
0
SAVE "nombre" CODE 16384, 6912
```

no pueden verificarse ya que al detectar el ordenador la cabecera de el programa, aparece en pantalla:

```
Bytes: nombre
```

modificando por tanto la memoria de pantalla y dando lugar al correspondiente error.

LOAD

Acceso al teclado

VAL



VAL \$

Definición

La sentencia "LOAD" permite cargar en la memoria del ordenador un programa grabado en cinta de cassette.

"LOAD" borra el programa existente en ese momento en el ordenador y sus variables.

La estructura general es:

| SENTENCIA | ARGUMENTO |
|-----------|---------------|
| LOAD | "nombre" tipo |

Ejemplo:

```
– LOAD "OH"
– LOAD "T" CODE
– LOAD "PP" DATA Z ( )
– LOAD "NT" DATA P$ ( )
```

Los campos "nombre" y "tipo" siguen las mismas reglas que en el caso de "SAVE".

a) Carga de programas BASIC.

```
LOAD "nombre"
```

Si el programa fue almacenado con el tipo "LINE" automáticamente se autoejecutará.

b) Carga de programas en C/M o de bytes.

```
LOAD "nombre" CODE comienzo, longitud
LOAD "nombre" CODE comienzo
LOAD "nombre" CODE
```

como casos específicos puede utilizarse para la carga de pantallas:

```
LOAD "nombre" SCREEN$
```

La forma de cargar en pantalla una información es un tanto peculiar ya que su memoria está distribuida en tres zonas. Primero, se cargan los bytes cero de cada carácter (matriz de 8 por 8 pixel), de la primera zona; a continuación, los bytes uno y así sucesivamente hasta completar la primera zona. Las zonas dos y tres se cargan de la misma manera y al final se incluyen los atributos.

El siguiente programa ayudará a comprender la secuencia de carga de la memoria de pantalla.

y para los GDU

```
10 REM *****
   REM MEMORIA/PANTALLA
   REM *****
20 LET direccion=16384
30 LET longitud=6912
40 FOR n=direccion TO direccion+longitud
50 FOR m=n,255
60 NEXT m
```

```
LOAD "nombre" CODE USR "a"
```

c) Carga de matrices numéricas.

```
LOAD "nombre" DATA letra ( )
```

d) Carga de matrices de cadena

```
LOAD "nombre" DATA letra $ ( )
```

Al cargar una matriz, bien sea de cadena o numérica, no se borra el programa BASIC existente, pero si el contenido de cualquier matriz que anteriormente estuviera definida con el mismo nombre.

PROGRAMA 3

```

10 REM *****
  *
  * CURSO/BASIC *
  * ***** *
  * EDIT/DIR *
  * ***** *
12 BORDER 4 PAPER 4 INK 0 C
LS
14 POKE 23658,8
20 DIM D$(21,123)
22 REM DATOS CARATULA
30 INPUT "NOMBRE DE LA CINTA: "; LINE A$
32 IF LEN A$ > 10 THEN LET A$=A$(TO 10)
34 PRINT "NOMBRE > "; A$
36 LET D$(21,1 TO 10)=A$
40 INPUT "CARA: "; LINE A$
42 IF LEN A$ > 1 THEN LET A$=A$(TO 1)
44 PRINT "CARA > "; A$
46 LET D$(21,11)=A$
50 INPUT "PROPIETARIO: "; LINE A$
52 IF LEN A$ > 12 THEN LET A$=A$(TO 12)
54 PRINT "PROPIETARIO > "; A$
56 LET D$(21,14 TO 25)=A$
60 INPUT "COMENTARIOS: "; LINE A$
62 IF LEN A$ > 90 THEN LET A$=A$(TO 90)
64 PRINT "COMENTARIOS > "; A$
66 LET D$(21,34 TO 123)=A$
70 PRINT #0;"PULSE UNA TECLA PARA CONTINUAR."; PAUSE 0
80 CLS
82 REM DATOS PROGRAMAS
100 FOR X=1 TO 20
110 INPUT "PROGRAMA: "; LINE A$
115 IF LEN A$ > 10 THEN LET A$=A$(TO 10)
120 PRINT "PROGRAMA N. ";
122 IF X > 9 THEN GO TO 126
124 PRINT " ";
126 PRINT X; " > "; A$
130 LET D$(X,1 TO 10)=A$
140 INPUT "CONTADOR: "; LINE A$
145 IF LEN A$ > 3 THEN LET A$=A$(TO 3)

```

```

150 PRINT "CONTADOR > "; A$
160 LET D$(X,11 TO 13)=A$
170 INPUT "PROGRAMADOR: "; LINE A$
175 IF LEN A$ > 12 THEN LET A$=A$(TO 12)
180 PRINT "PROGRAMADOR > "; A$
190 LET D$(X,14 TO 25)=A$
200 INPUT "FECHA: "; LINE A$
205 IF LEN A$ > 8 THEN LET A$=A$(TO 8)
210 PRINT "FECHA > "; A$
220 LET D$(X,26 TO 33)=A$
230 INPUT "COMENTARIOS: "; LINE A$
235 IF LEN A$ > 90 THEN LET A$=A$(TO 90)
240 PRINT "COMENTARIOS > "; A$
250 LET D$(X,34 TO 123)=A$
252 POKE 23658,8
260 PRINT #0;"MAS PROGRAMAS (5/N)";
262 LET A$=INKEY$
270 IF A$="3" THEN GO TO 300
280 IF A$="N" THEN GO TO 1000
290 GO TO 262
300 CLS
310 NEXT X
1000 PRINT " <<<<<<< FIN DE EDICION >>>>>>>"
1002 INPUT 0
1010 PRINT #0;"PULSE UNA TECLA PARA CONTINUAR."; PAUSE 0
1020 CLS
1022 REM GRABACION DE DATOS
1030 PRINT "INSERTE EN EL CASSETTE LA CINTA " "DONDE QUIERA GRABAR LOS DATOS." "EDITADOS."
1040 PRINT #0;"PULSE UNA TECLA PARA GRABAR."; PAUSE 0
1045 CLS
1050 SAVE "DIR" DATA D$( )
1060 PRINT "GRABACION DE DATOS EFECTUADA"
1070 PRINT #0;AT 0,0;"REBOBINE LA CINTA PARA VERIFICARY PULSE UNA TECLA"; PAUSE 0
1080 CLS
1090 VERIFY "DIR" DATA D$( )
1092 CLS
1100 PRINT "VERIFICACION DE DATO 5 CORRECTA"
1110 PAUSE 0

```

MERGE

Acceso al teclado

RND



MODULO E

MERGE

SYMBOL SHIFT

Definición

La sentencia "MERGE" permite combinar varios programas BASIC.

Su estructura general es:

| SENTENCIA | ARGUMENTO |
|-----------|-----------|
| MERGE | "nombre" |

Ejemplo:

– MERGE "OK"

El argumento "nombre" sigue las mismas reglas que en el caso de "SAVE".

La secuencia de operación

es la siguiente:

a) Cargar el primer programa de la forma acostumbrada.

LOAD "nombre"

b) Cargar el siguiente de la forma:

MERGE "nombre"

Si hubiera que combinar más programas, se procedería de la misma manera que en el punto "b".

NOTA

Hay que tener cuidado en que no haya líneas de programa con el mismo número, ya que el resultado podría ser desastroso.

Comodidad de uso

Las sentencias "LOAD", "VERIFY" y "MERGE" tienen cierta particularidad que les permita cargar, verificar o combinar el primer programa que encuentren, aunque no conozcamos su nombre; para ello debemos sustituir el nombre por una cadena vacía.

Ejemplos:

```
LOAD ""  
LOAD "" CODE  
VERIFY ""  
VERIFY "" CODE  
MERGE ""
```

Búsqueda de programas

Durante la búsqueda de programas para su carga, verificación o fusión con otros, se van visualizando en pantalla todos aquellos programas o datos que el ordenador va encontrando. Esto puede servirnos de referencia para una mejor localización, ya que podemos utilizar el avance rápido hacia adelante o hacia atrás, de nuestro aparato de cassette, para ahorrar tiempo.

Referencias

Cuando el ordenador localiza un programa, se nos visualiza en pantalla su nom-

bre, así como una referencia indicativa del tipo de programa; éstas pueden ser de cuatro tipos:

a) Programas BASIC.

Program : nombre

b) Programas en C/M, pantallas o bytes.

Bytes : nombre

c) Matrices numéricas.

Number array : nombre

d) Matrices de cadena.

Character array : nombre

Errores

En el manejo de estas sentencias pueden aparecer cualquiera de estos errores:

a) Fuera de memoria.

4 out of memory

Este error se produce cuando no hay suficiente memoria para cargar o combinar un programa, bytes o matrices.

b) Nombre de programa inválido.

F Invalid file name

Ocurre al grabar un programa con un nombre de más de diez caracteres o al asignarle una cadena vacía.

c) Error de carga en la cinta.

R Tape loading error

Aparece cuando el ordena-

dor no puede cargar un programa o cuando la verificación ha sido incorrecta.

Este error puede ser debido entre otras causas a:

- Cabeza lectora del cassette sucia.
- Pilas gastadas en exceso.
- Cintas apelmazadas.
- Cintas deterioradas.
- Etc.

Programas

El programa número "1" permite conocer el contenido de una cinta de cassette, dándonos información de los programas almacenados, así como una serie de datos útiles: su comienzo y su longitud. Si tenemos conectada una impresora los datos saldrán simultáneamente.

Una vez ejecutado, nos pregunta el nombre que queremos dar a la cinta (máx. 15 caracteres) y a continuación la cara (ó). Posteriormente se pone el cassette, con la cinta, en marcha y a esperar que el ordenador visualice la información según vaya detectando las cabeceras de los programas.

El programa utiliza una pequeña rutina en código máquina, ya que parte de la información que suministra es inaccesible desde el BASIC.

El programa número "2" es un programa de utilidad que permite conocer el directorio de una cinta, es decir los programas que hay grabados, así como una serie de datos anexos: Nombre del programador, marcaje del contador del cassette, etc. También permite elegir un programa y cargarlo automáticamente.

Las instrucciones de manejo se encuentran en el propio programa.

Los datos referentes a esa cinta deben ser editados con el programa número "3" que permite confeccionar la información de un máximo de 20 programas.

Los programas deben ser grabados en el siguiente orden:

- a) El programa número "2" como LINE 6000.
- b) La tabla de datos que genera el programa número "3".
- c) A continuación los pro-

```
PROGRAMA N. 2 > QWERTY
CONTADOR      > 123
PROGRAMADOR   > D.F.G.
FECHA         > 23/04/85
COMENTARIOS   > ESTE PROGRAMA
EXPLORA EL TECLADO DEL SPECTRUM
```

Edición de datos con el "EDIT/DIR".

gramas correspondientes a esa cinta. Como el programa "directorio" es el primero de la cinta, para ejecutarlo rebo-

bina la cinta y teclea:

LOAD ""

GESTION DE IMPRESORA

Las impresoras son unos periféricos similares a una máquina de escribir, pero sin teclado incorporado, ya que las órdenes de escritura proceden del ordenador al que están conectadas.

Las utilidades de una impresora son diversas, desde una simple obtención en papel de los listados de programas hasta la confección, por ejemplo, de cartas con un sofisticado "PROCESADOR DE TEXTOS".

En este capítulo se van a explicar aquellas sentencias, que de un modo directo, gestionan las impresoras "ZX" o similares:

```
LPRINT
LLIST
COPY
```

```
LPRINT
```

Acceso al teclado

LPRINT



MODO E

PAPER

Definición

La sentencia "LPRINT" realiza la misma tarea que

"PRINT" pero obteniendo los resultados por impresora en lugar de la pantalla.

Su estructura general es idéntica a la de "PRINT"; por tanto si tiene alguna duda consulte la página 71 y sucesivas.

Ejemplos:

- LPRINT "ORDENADOR"
- LPRINT a
- LPRINT 3/4 + 7
- LPRINT K\$

Con "LPRINT" pueden utilizarse las sentencias auxiliares "AT" y "TAB".

a) LPRINT AT

```
10 REM *****
      ELEMENTO AT *****
20 FOR N=0 TO 21
30 LPRINT AT N,0 "MICROHOBBY"
40 NEXT N
```

b) LPRINT TAB

```
10 REM *****
      ELEMENTO TAB *****
20 FOR N=21 TO 0 STEP -1
30 LPRINT TAB N "MICROHOBBY"
40 NEXT N
```

En general, el argumento de "LPRINT" no se imprime directamente, sino que se almacena en una memoria intermedia denominada "BUFFER DE IMPRESION DE LI-

NEA"; cuando este *buffer* se llena, la impresión se realiza.

Ejemplo:

```
10 REM *****
      BUFFER INTERMEDIO *****
20 FOR N=0 TO 31
30 INPUT "LETRA " : A$
40 LPRINT A$;1;
50 NEXT N
```

A pesar de que en la línea 40 hay una impresión del elemento primero de la variable de cadena A\$, ésta no se realiza hasta que se completa el bucle Z (0 - 31); ya que 32 caracteres son los que se almacenan en la denominada memoria intermedia.

Hay otras ocasiones en las que se realiza la impresión sin estar lleno el buffer:

a) Cuando se utiliza "LPRINT" en comandos directos, independientemente del formato de impresión (coma o punto y coma).

Ejemplos:

```
LPRINT "hola"
LPRINT "pepe";
LPRINT "juan";
LPRINT "pan";
```

b) Dentro de un programa.

- Cuando "LPRINT" no va acompañado de ningún formato.

Ejemplo:

PROGRAMA 1

```

10 REM *****
+ CURSO/BASIC *
*****
+ IMPRESORA *
*****
12 LET longitud=31
20 LET primero=33
30 LET ultimo=63
32 PRINT AT 0,0: INVERSE 1;"..
TEST DE IMPRESORA
34 PRINT #0;" Pulse una tecla
    
```

```

Para comenzar"
36 PAUSE 0: INPUT 0
40 FOR n=0 TO ultimo
50 LET caracter=primero+n
70 FOR x=caracter+longitud TO
caracter STEP -1
80 LPRINT CHR$ x;
90 NEXT x
100 NEXT n
102 FOR z=0 TO 5
104 LPRINT
106 NEXT z
110 PRINT AT 21,0: INVERSE 1;"..
FIN DEL TEST
120 PAUSE 0
    
```

```

10 LPRINT "disco"
20 LPRINT "lápiz"
    
```

— Cuando se requiera una nueva línea por alguno de los siguientes formatos:

COMA

```

10 LPRINT "Erase una vez un lobo",
20 LPRINT "que quería..."
    
```

APOSTROFE

```

10 LPRINT "Peras" ' "Manzanas"
    
```

ELEMENTO "AT"

```

10 LPRINT AT 4,5: "AJAJA"
20 LPRINT AT 10,0: "TURURU"
    
```

Inserte un punto y coma al final de la línea 10 y observe la diferencia. Con la sentencia "AT" se ignora el indicativo de número de línea.

ELEMENTO "TAB"

```

10 FOR n = 21 TO 0
20 LPRINT TAB n; "*";
30 NEXT n
    
```

c) Al finalizar un programa, si se ha quedado algo sin imprimir

```

10 LPRINT "adiós"
20 PAUSE 0
    
```

Al pulsar cualquier tecla, se imprime la cadena "adiós",

```

@?>=<;:9876543210/.,+*)('&%$#"'!
A@?>=<;:9876543210/.,+*)('&%$#"'
BA@?>=<;:9876543210/.,+*)('&%$#
CBA@?>=<;:9876543210/.,+*)('&%$
DCBA@?>=<;:9876543210/.,+*)('&%
EDCBA@?>=<;:9876543210/.,+*)('
FEDCBA@?>=<;:9876543210/.,+*)('
GFEDCBA@?>=<;:9876543210/.,+*)('
HGFEDCBA@?>=<;:9876543210/.,+*)
IHGFEDCBA@?>=<;:9876543210/.,+*
JIHGFEDCBA@?>=<;:9876543210/.,+
KJIHGFEDCBA@?>=<;:9876543210/.,+
LKJIHGFEDCBA@?>=<;:9876543210/.,+
MLKJIHGFEDCBA@?>=<;:9876543210/.,+
NMLKJIHGFEDCBA@?>=<;:9876543210/.,+
ONMLKJIHGFEDCBA@?>=<;:9876543210/.,+
PONMLKJIHGFEDCBA@?>=<;:9876543210/.,+
QPONMLKJIHGFEDCBA@?>=<;:9876543210/.,+
RQPONMLKJIHGFEDCBA@?>=<;:9876543210/.,+
SRQPONMLKJIHGFEDCBA@?>=<;:9876543210/.,+
TSRQPONMLKJIHGFEDCBA@?>=<;:9876543210/.,+
UTSRQPONMLKJIHGFEDCBA@?>=<;:9876543210/.,+
    
```

Ejemplo del programa "Test".

que se encontraba almacenada en el buffer de impresión.

LLIST

Acceso al teclado

L LIST



MODO **E**

FLASH

Definición

"LLIST" es el equivalente a

"LIST" en impresora.

Su estructura general es:

| SENTENCIA | ARGUMENTO |
|-----------|--------------|
| LLIST | n." de línea |

Ejemplos:

- LLIST
- LLIST 30
- LLIST a
- LLIST 5 * b

Cuando el número de línea se omite, el ordenador interpreta que el listado es a partir de la primera línea.

A diferencia de "LIST" el listado se imprime todo seguido, ya que no aparece el conocido mensaje:

scroll?

Los listados por impresora son muy convenientes, ya que ayudan en la depuración de programas al poder localizar con mayor facilidad las sentencias.

```
COPY
```

Acceso al teclado

LN

MODO K

Z COPY

BEEP

Definición

La sentencia "COPY" permite sacar por impresora una copia de la imagen visualizada en pantalla.

"COPY" no necesita de ningún argumento para ejecutarse.

La reproducción que hace "COPY" de la pantalla se basa en la impresión de los pixel con color de "tinta", por tanto, no se sorprenda si al intentar realizar un "COPY" del dibujo proporcionado por el programa de la página 227, no se imprime absolutamente nada, ya que está realizado a base de bloques de color "papel" con el carácter "espacio"; si desea imprimirlo, modifique la siguiente línea:

```
1030 PRINT INK color; AT y, x; █
```

Tampoco se imprime nada si intenta realizar un "COPY" de los listados que proporciona el ordenador al pulsar la



Sólo los pixel con color de tinta se imprimen al realizar un "copy".

tecla "ENTER", ya que éstos se borran al ejecutarse una sentencia o comando.

Realice prácticas con los ejercicios de las sentencias "PLOT", "DRAW" y "CIRCLE".

Programa de utilidad

El programa n.º 1 permite realizar uno de los típicos TEST de impresora, en los

que los caracteres se van desplazando hacia la derecha; de esta manera se puede, mediante una rápida ojeada, comprobar el correcto funcionamiento de una impresora.

Otras impresoras

En el mercado existen diversos Interfaces para poder

Letra Normal de matriz de puntos
Letra Itálica o cursiva
Letra de calidad (NLQ)
Escritura en negrita
Escritura subrayada
 Paso de escritura condensado
 Paso de escritura "Elite"
 Paso de escritura "Pica"
Itálica condensada
 Línea con **superíndices** y **subíndices**
 La letra más pequeña posible
Normal expandido

Itálica expandida

Condensado expandido

Itálica condensada expandida

Diversos tipos de letra, proporcionados por una impresora de calidad.

acoplar cualquier impresora, que no sea "ZX", al Spectrum.

Los protocolos de comunicación más importantes, por los que se rigen estas impresoras, son:

CENTRONICS
RS-232

El protocolo "CENTRONICS" se base en el envío *paralelo* de la información, mientras que la "RS-232" lo realiza en *serie*.

La mayoría de estos Interfases ocupan una pequeña parte de la memoria del Spectrum y necesitan de un *Software* que las gestione.

Tipos de Impresora

En la actualidad existe una gran variedad. Atendiendo a

su sistema de funcionamiento se pueden clasificar de la siguiente manera:

MARGARITA
 AGUJAS
 TERMICAS
 LINEAS
 INYECCION DE TINTA
 ELECTROSTATICAS
 LASER
 ETC...

Las impresoras de *Margarita* tienen la ventaja de tener un tipo de letra idéntico al proporcionado por una máquina de escribir, pero tienen el inconveniente de que su velocidad expresada en "cps" (caracteres por segundo) es relativamente baja.

Las impresoras de *agujas* son las más utilizadas en los

ordenadores personales, ya que su relación servicio/precio es bastante adecuada.

La velocidad de estas impresoras es superior a las de Margarita, pero su tipo de letra es más imperfecto, ya que las realiza a base de puntos.

Las impresoras *térmicas* también se utilizan en los ordenadores personales; en éstas, el tipo de papel utilizado es especial, ya que cambian de color al calentarse.

El resto de impresoras no son de uso frecuente con ordenadores personales, ya que su precio es bastante elevado y están diseñadas para equipos donde el volumen de impresión es grande; por ejemplo, las de *líneas* imprimen una línea de una sola vez y las de *láser* hasta una página.


```

FIRMWARE ID: 21066 REV A
FIRMWARE ID: 21065 REV A
GENERATOR INTERNAL
1K RAM

```

```

FORM LENGTH 12 INCH
BOTTOM OF FORM SKIP YES
LINES PER INCH 6
CHARACTER PER INCH 10
CR = CR LF YES
LF AT LINE END NO
CHARACTER SET US ASCII
SLASHED ZERO YES
TEST MODUS ROLLING ASCII
END OF MENU.

```

```

CHANGE MENU YES
CHANGE FORM LENGTH YES
4 INCH NO
5 INCH NO
6 INCH NO
8 INCH NO
8.5 INCH NO
11 INCH NO
12 INCH NO
14 INCH NO CHANGE

```

```

BOTTOM OF FORMAT SKIP YES
CHANGE PRINT FORMAT YES
CHANGE LPI YES
6 NO
8 NO CHANGE
CHANGE CPI YES
10 NO
12.5 NO
16 2/3 NO
20 NO CHANGE

```

```

CR = CR LF YES
LF AT LINE END NO
CHANGE CHARACTER SET YES
US ASCII NO
UK ASCII NO
FRENCH/BELGIAN NO
GERMAN NO
ITALIAN NO
SWEDISH/FINNISH NO
DANISH/NORWEGIAN NO
SPANISH YES
SLASHED ZERO YES
CHANGE TEST MODUS YES
ROLLING ASCII NO
GENERATOR TEST YES
END OF MENU.

```

Menú de opciones de impresión programables por el usuario.

280 MICROBASIC

Elección de una impresora

A la hora de elegir una impresora deben tenerse en cuenta ciertos factores que intervendrán directamente en su precio.

- La velocidad de impresión en cps.

- Si imprime en dos sentidos (bidireccional).

- Número de caracteres por línea.

- Tipo de papel:

- Rollo
- Perforado
- Térmico
- Hojas

- Tipo de tinta.

- Juegos de caracteres que incorpora.

- Si tiene un sistema de *Fricción* y de *Arrastre* combinado que permite introducir o bien hojas sueltas o bien papel perforado.

- Si imprime en varios colores.

La elección deberá ir en concordancia con el tipo de utilidad a la que se vaya a destinar y con el volumen de trabajo que deberá soportar.

Caracteres de control

La mayoría de las impresoras utilizan algunos caracteres ASCII para cambiar el tamaño de las letras, la distancia entre líneas, el tipo de letra, si deben ir subrayadas o en negrita, etc.

Estos códigos dependen del tipo de impresora, por tanto deberá leer las instrucciones específicas de la que esté manejando; pero en general la forma de introducir estos comandos es:

```
LPRINT CHR$ n
```

donde "n" es el código de control a enviar.

US ASCII

```
!"#$%&'()*+,-./0123456789:;<=>?  
@ABCDEFGHIJKLMNPOQRSTUVWXYZ[\]^_`abcdefghijklmnopqrstu  
vwxyz{|}~  
!"#$%&'()*+,-./0123456789:;<=>?  
@ABCDEFGHIJKLMNPOQRSTUVWXYZ[\]^_`abcdefghijklmnopqrstu  
vwxyz{|}~
```

UK ASCII

```
!"£$%&'()*+,-./0123456789:;<=>?  
@ABCDEFGHIJKLMNPOQRSTUVWXYZ[\]^_`abcdefghijklmnopqrstu  
vwxyz{|}~  
!"£$%&'()*+,-./0123456789:;<=>?  
@ABCDEFGHIJKLMNPOQRSTUVWXYZ[\]^_`abcdefghijklmnopqrstu  
vwxyz{|}~
```

FRENCH/BELGIAN

```
!"£$%&'()*+,-./0123456789:;<=>?  
àABCDEFGHIJKLMNPOQRSTUVWXYZ°ç$^`abcdefghijklmnopqrstu  
vwxyzéùè"  
!"£$%&'()*+,-./0123456789:;<=>?  
àABCDEFGHIJKLMNPOQRSTUVWXYZ°ç$^`abcdefghijklmnopqrstu  
vwxyzéùè"
```

GERMAN

```
!"#$%&'()*+,-./0123456789:;<=>?  
$ABCDEFGHIJKLMNPOQRS TUVWXYZÀÖÜ`_`abcdefghijklmnopqrstu  
vwxyzäöüß  
!"#$%&'()*+,-./0123456789:;<=>?  
$ABCDEFGHIJKLMNPOQRS TUVWXYZÀÖÜ`_`abcdefghijklmnopqrstu  
vwxyzäöüß
```

ITALIAN

```
!"£$%&'()*+,-./0123456789:;<=>?  
$ABCDEFGHIJKLMNPOQRSTUVWXYZ°#è^`_`abcdefghijklmnopqrstu  
vwxyzàòèì  
!"£$%&'()*+,-./0123456789:;<=>?  
$ABCDEFGHIJKLMNPOQRSTUVWXYZ°#è^`_`abcdefghijklmnopqrstu  
vwxyzàòèì
```

SWEDISH/FINNISH

```
!"0#%&'()*+,-./0123456789:;<=>?  
åABCDEFGHIJKLMNPOQRSTUVWXYZÉÄÖÅ`_`abcdefghijklmnopqrstu  
vwxyzéäöå  
!"0#%&'()*+,-./ü123456789:;<=>?  
åABCDEFGHIJKLMNPOQRSTUVWXYZÉÄÖÅ`_`abcdefghijklmnopqrstu  
vwxyzéäöå
```

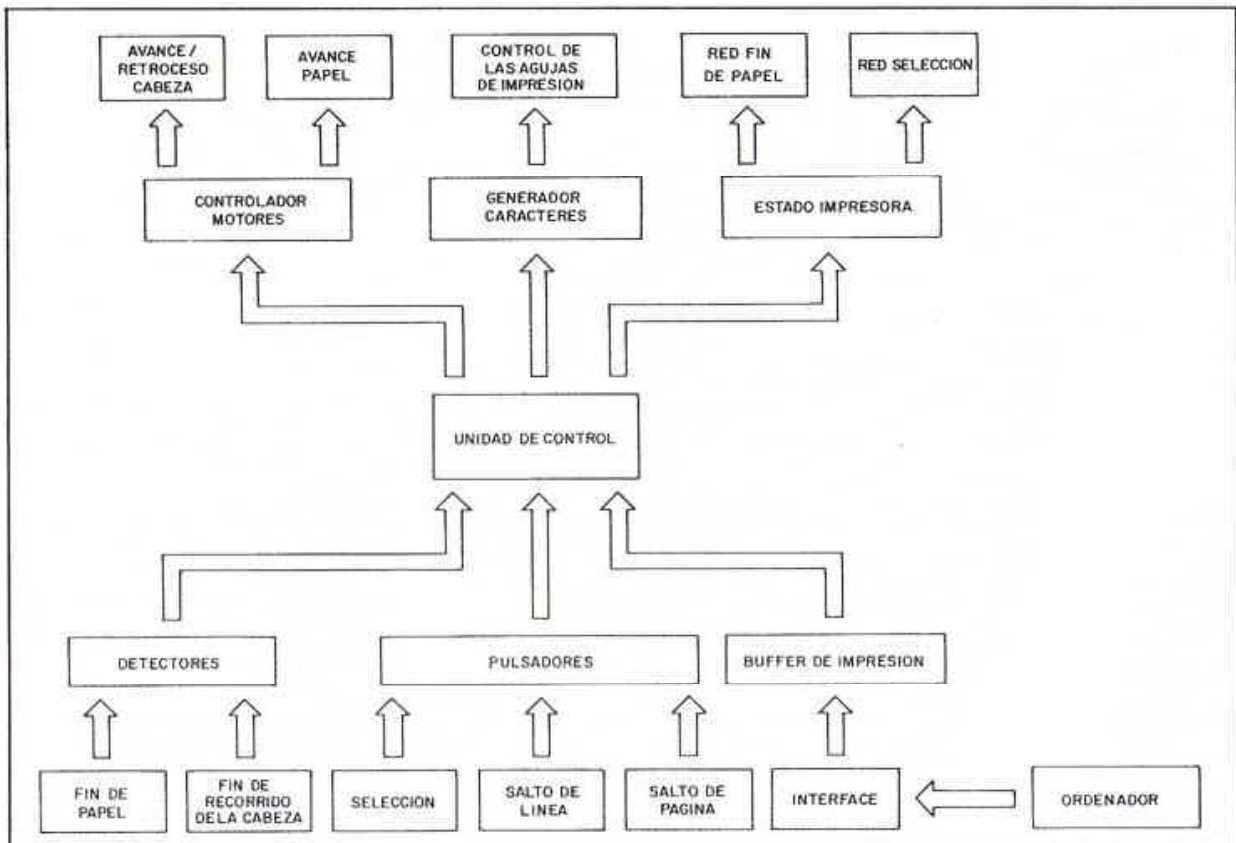
DANISH/NORWEGIAN

```
!"0#%&'()*+,-./0123456789:;<=>?  
$ABCDEFGHIJKLMNPOQRSTUVWXYZ@#øå`_`abcdefghijklmnopqrstu  
vwxyz`eøå  
!"0#%&'()*+,-./~123456789:;<=>?  
$ABCDEFGHIJKLMNPOQRSTUVWXYZ@#øå`_`abcdefghijklmnopqrstu  
vwxyz`eøå
```

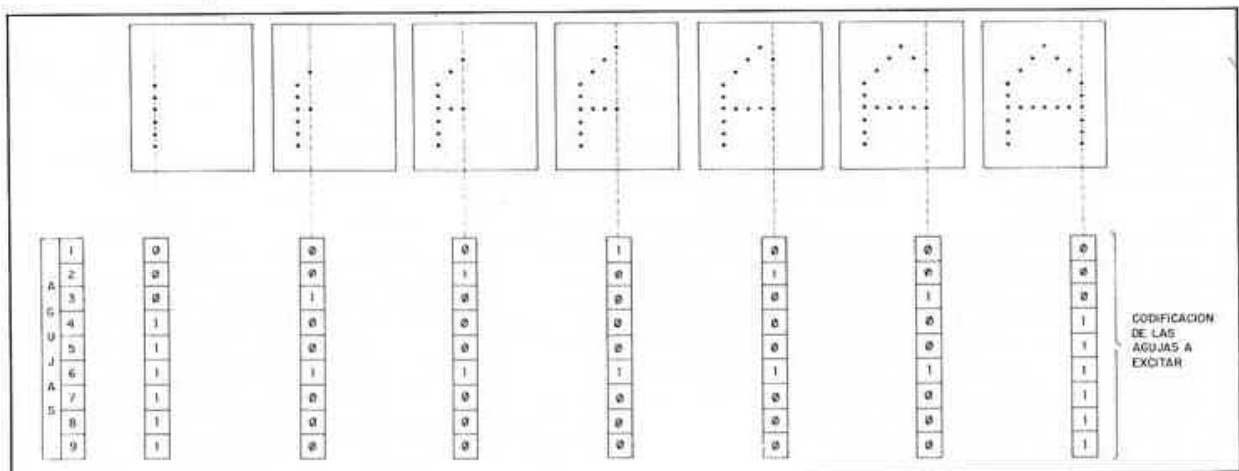
SPANISH

```
!"R$%&'()*+,-./0123456789:;<=>?  
&ABCDEFGHIJKLMNPOQRSTUVWXYZiñú`_`abcdefghijklmnopqrstu  
vwxyzíñóú  
!"R$%&'()*+,-./0123456789:;<=>?  
&ABCDEFGHIJKLMNPOQRSTUVWXYZiñú`_`abcdefghijklmnopqrstu  
vwxyzíñóú
```

ACTUAL CHARACTERSET SPANISH



Esquema de bloques de una impresora.



Formación del carácter "A" con una impresora de agujas de 9 x 7.

- GESTION DE IMPRESORAS -

Las impresoras son unos periféricos similares a una máquina de escribir, pero sin teclado incorporado, ya que las ordenes de escritura proceden del ordenador al que estén conectados.

Las utilidades de una impresora son diversas, desde la simple obtención en papel de los listados de programas hasta la confección, por ejemplo, de cartas con un sofisticado "PROCESADOR DE TEXTOS".

Las impresoras son una herramienta imprescindible en un "Procesador de Textos".

INTERFACE-1

El "Interface-1" es un *controlador de periféricos* que permite manejar con el empleo de ciertas sentencias:

- Hasta un máximo de ocho *Microdrives* (unidades de almacenamiento).

- La *red de área local*, mediante la cual se pueden comunicar hasta 64 Spectrum.

- El Interface RS-232, para conectar aquellos periféricos que utilicen este protocolo de transmisión de datos en serie.

Prolonga el conector de expansión, de manera que se puedan añadir más periféricos al Spectrum.



Canales y Corrientes

El intercambio de datos entre el Spectrum y sus periféricos se realiza a través de los denominados *canales de comunicación*, (ver pag. 75).

Para identificar una comunicación debe especificarse el *canal* y la *corriente*.

Los canales pueden ser de entrada, salida o de entrada/salida.

ENTRADA:

- Teclado.

SALIDA:

- Pantalla del televisor o monitor.

- Impresora.

ENTRADA/SALIDA:

- Fichero Microdrive.
- Un Spectrum conectado a la red.

- El interface RS-232.

Conexión del Microdrive.

Los distintos indicativos del canal están especificados en la figura adjunta.

El flujo de datos puede llegar a recibirse, de estos canales, por varios caminos o *corrientes*; en total el Spectrum dispone de 16, numerados del # 0 al # 15.

Las corrientes # 0 a # 3 tienen una asignación definida por el sistema operativo; las restantes # 4 a # 15 están libres para poder ser definidas por el propio usuario.

Asignación de canales y corrientes

Para poder asignar un "canal" a una «corriente» debe utilizarse la sentencia "OPEN #".

```
OPEN #
```

Acceso al teclado

GREEN
INV. VIDEO

MODO **E**



SYMBOL SHIFT

OPEN #

Su estructura general es:

| SENTENCIA | ARGUMENTO |
|-----------|------------------|
| OPEN # | corriente, canal |

Ejemplos:

a) Asignación de la corriente # 6 a un fichero de la unidad "3" Microdrive denominado "curso".

```
OPEN # 6; "m"; 3; "curso"
```

b) Asignación de la corriente # 8 a la estación 4 de la red de área local.

OPEN # 8; "n"; 4

c) Asignación de la corriente # 15 a una impresora conectada al interface RS-232.

OPEN #15; "i"

d) Para asignar una corriente a cualquiera de los canales: "k" (teclado) "s" (pantalla) o "p" (impresora) debe utilizarse necesariamente el separador "coma"; en los demás canales, puede utilizarse indistintamente "coma" o "punto y coma".

OPEN # 5; "s"
OPEN # 7; "p"
OPEN # 10; "k"

Desactivación de canales y corrientes

Para quitar la asignación de un "canal" a su correspondiente "corriente", se utiliza la sentencia "CLOSE #".

CLOSE #

Acceso al teclado

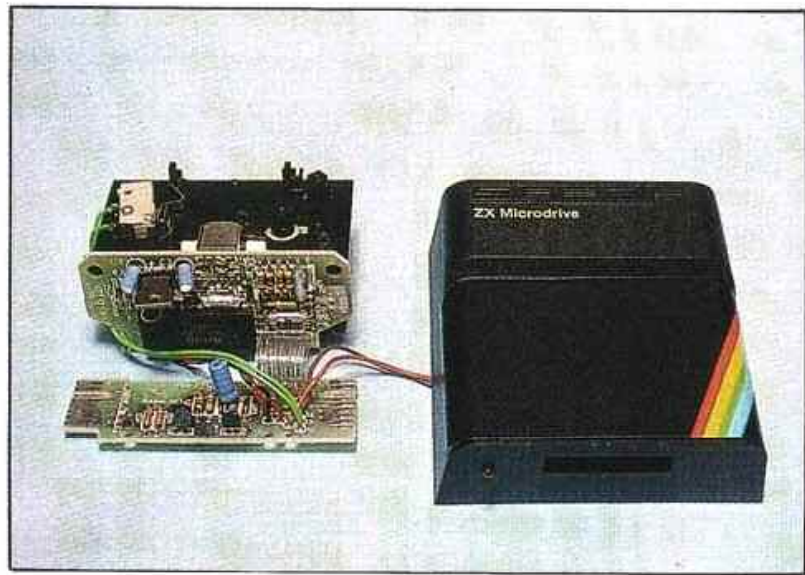
CYAN



CLOSE #

MODO E

SYMBOL SHIFT



Interior del Microdrive

Su estructura general es:

| SENTENCIA | ARGUMENTO |
|-----------|-----------|
| CLOSE # | corriente |

Ejemplos:

- CLOSE # 6
- CLOSE # n
- CLOSE # z + 1
- CLOSE # VAL a\$

Al realizar la desactivación, los datos que estuvieran almacenados en la memoria intermedia serán transmitidos a su correspondiente canal.

El Microdrive

Es un periférico diseñado para almacenar gran cantidad de datos, con un tiempo de acceso de lectura/escritura superior al proporcionado por un cassette. La información se almacena en unos cartuchos intercambiables.

Las operaciones que pueden realizarse con el Microdrive son:

- Formatear o inicializar cartuchos.
- Almacenar y verificar

programas en cartucho.

- Cargar programas.
- Borrar *ficheros* de datos y programas.
- Combinar los programas residentes en memoria con los archivados en cartucho.
- Catalogar cartuchos, es decir, obtener un *directorio* o listado de los programas grabados en él.
- Abrir y cerrar ficheros de datos.
- Grabar y leer datos de un fichero.
- Mover datos entre ficheros.

FORMAT

Acceso al teclado

BLACK

DELETE

MODO E



SYMBOL SHIFT

FORMAT

PROGRAMA 1

```

10 REM *****
*          *
*  CURSO/BASIC  *
*          *
*****
*          *
*  FICHERO      *
*          *
*****

20 BORDER 4: PAPER 4: INK 0: C
LS
22 POKE 23656,8
30 LET ultimo=0
40 LET e$="": LET f$=""
100 DIM a$(100,40)
500 REM MENU
510 CLS : CLOSE #4
520 PRINT INVERSE 1;"          FICHE
RO BIBLIOTECARIO
522 PRINT INVERSE 1;AT 5,10;"
OPCIONES "
530 PRINT AT 9,10;"1 GENERAR"
540 PRINT AT 12,10;"2 ALMACENAR
"
550 PRINT AT 15,10;"3 RECUPERAR
"
552 PRINT AT 18,10;"4 VISUALIZA
R"
560 PRINT #0;AT 1,4;"Pulse la o
pcion deseada"
562 PAUSE 0
570 LET z$=INKEY$
580 IF z$<"1" OR z$>"4" THEN BE
EP 0.2,-10: GO TO 562
590 BEEP 0.2,10: GO TO (VAL z$)
*1000
1000 REM GENERAR
1002 IF ultimo=0 THEN GO TO 1020
1004 CLS
1006 IF ultimo>0 AND ultimo<=99
THEN GO TO 1015
1010 IF ultimo>99 THEN PRINT INV
ERSE 1;AT 4,5;" FICHERO COMPLET
O "; INVERSE 0;AT 10,7;"1 NUEVO
FICHERO";AT 15,7;"2 MENU OPCIONE
S";#1;AT 1,4;"Pulse la opcion de
seada"
1011 PAUSE 0: LET z$=INKEY$
1012 IF z$="N" THEN BEEP 0.2,10:
DIM a$(100,40): LET ultimo=0: L
ET e$="": LET f$="": GO TO 1020
1013 IF z$="H" THEN BEEP 0.2,10:
GO TO 500
1014 BEEP 0.2,-10: GO TO 1011
1015 PRINT AT 7,8;"0 NUEVO FICHE
RO";AT 14,8;"3 CONTINUAR";#1;AT
1,4;"Pulse la opcion deseada"
1016 PAUSE 0: LET z$=INKEY$
1017 IF z$="N" THEN BEEP 0.2,10:
DIM a$(100,40): LET ultimo=0: L
ET e$="": LET f$="": GO TO 1020
1018 IF z$="C" THEN BEEP 0.2,10:
CLS : GO TO 1020
1019 BEEP 0.2,-10: GO TO 1015
1020 CLS : PRINT INVERSE 1;"
GENERAR FICHA
1030 PRINT AT 7,0;" NOMBRE "
1040 PRINT AT 14,0;" AUTOR "
1050 FOR n=ultimo+1 TO 100
1052 LET n$=STR$ n+
"
1054 PRINT AT 0,22; INVERSE 1;n$
(1 TO 3)
1060 INPUT "Nombre> "; LINE b$:
BEEP 0.2,10
1062 IF b$="FIN" THEN GO TO 500
1064 IF b$="" THEN GO TO 1060
1070 IF LEN b$>20 THEN LET b$=b$
(1 TO 20)
1080 PRINT AT 7,9;b$
1090 LET a$(n,1 TO 20)=b$
1100 INPUT "Autor> "; LINE c$: B
EEP 0.2,10
1102 IF c$="" THEN GO TO 1100
1110 IF LEN c$>20 THEN LET c$=c$
(1 TO 20)
1120 PRINT AT 14,9;c$
1130 LET a$(n,21 TO 40)=c$
1140 LET ultimo=ultimo+1

```

```

1150 PAUSE 50
1160 PRINT AT 7,9;"
"
1170 PRINT AT 14,9;"
"
1180 NEXT n
1190 GO TO 500
2000 REM ALMACENAR
2010 CLS
2020 PRINT INVERSE 1;"          GRAB
ACION DE FICHERO
2030 PRINT AT 5,0;" NOMBRE ";AT
10,0;" FECHA ";AT 15,0;" CLAVE
"
2040 INPUT "Nombre> "; LINE b$:
BEEP 0.2,10
2050 IF b$="" THEN GO TO 2040
2060 IF LEN b$>6 THEN LET b$=b$(
1 TO 6)
2070 PRINT AT 5,9;b$
2080 INPUT "Fecha> "; LINE c$: B
EEP 0.2,10
2090 IF c$="" THEN GO TO 2080
2100 IF LEN c$>8 THEN LET c$=c$(
1 TO 8)
2110 PRINT AT 10,9;c$
2112 LET d$=""
2120 FOR n=0 TO 9
2130 PAUSE 0
2140 IF INKEY$=CHR$ 13 THEN BEEP
0.05,10: GO TO 2180
2150 LET d$=d$+INKEY$
2160 PRINT AT 15,9+n;"*"
2162 BEEP 0.05,10
2170 NEXT n
2180 PRINT AT 21,0; INVERSE 1;"
INTRODUZCA EL CART. MICRODRIVE "
: PAUSE 100
2190 PRINT #1;AT 1,1;"Pulse una
tecla para comenzar": PAUSE 0: I
NPUT 0: BEEP 0.2,10
2200 PRINT AT 21,0; INVERSE 1;"
GRABANDO CARTUCHO MICRODRIVE "
2210 OPEN #4;"m";1;b$+".CLU"
2220 PRINT #4;b$c$d$ultimo-1
2230 CLOSE #4
2240 OPEN #4;"m";1;b$+".DAT"
2250 FOR n=1 TO ultimo
2260 PRINT #4;a$(n)
2262 NEXT n
2270 CLOSE #4
2272 BEEP 0.2,10
2280 PRINT AT 21,0; INVERSE 1;"
GRABACION EFECTUADA
2290 PRINT #1;AT 1,1;"Pulse una
tecla para retornar"
2300 PAUSE 0: BEEP 0.2,10: GO TO
500
3000 REM RECUPERAR
3010 CLS
3020 PRINT INVERSE 1;"          LECT
URA DE FICHERO
3030 PRINT AT 5,0;" NOMBRE ";AT
10,0;" CLAVE ";
3040 INPUT "Nombre> "; LINE b$:
BEEP 0.2,10
3050 IF b$="" THEN GO TO 3040
3060 IF LEN b$>6 THEN LET b$=b$(
1 TO 6)
3070 PRINT AT 5,9;b$
3072 LET d$=""
3080 FOR n=0 TO 9
3082 PAUSE 0
3090 IF INKEY$=CHR$ 13 THEN BEEP
0.2,10: GO TO 3140
3100 LET d$=d$+INKEY$
3110 PRINT AT 10,9+n;"*"
3120 BEEP 0.05,10
3130 NEXT n
3140 PRINT AT 21,0; INVERSE 1;"
INTRODUZCA EL CART. MICRODRIVE "
: PAUSE 100
3150 PRINT #1;AT 1,1;"Pulse una
tecla para comenzar": PAUSE 0: I
NPUT 0: BEEP 0.2,10
3160 PRINT AT 21,0; INVERSE 1;"
CARGANDO CARTUCHO MICRODRIVE "
3170 OPEN #4;"m";1;b$+".CLU"

```

```

3180 INPUT #4; LINE e$; LINE f$;
LINE g$;ultimo
3190 IF g$<>d$ THEN NEW
3192 LET d$="": LET g$=""
3200 CLOSE #4
3202 DIM a$(100,40)
3210 OPEN #4;"m";1;b$+".DAT"
3220 FOR n=1 TO ultimo
3230 INPUT #4; LINE h$
3240 LET a$(n)=h$
3242 NEXT N
3244 CLOSE #4
3250 PRINT AT 21,0; INVERSE 1;"
FICHERO CARGADO
3252 BEEP 0.2,10
3270 PRINT #1;AT 1,1;"Pulse una
tecla para retornar"
3280 PAUSE 0: BEEP 0.2,10: GO TO
500
4000 REM VISUALIZAR
4002 CLS : PRINT INVERSE 1;"
VISUALIZACION DE FICHERO "
4004 PRINT AT 10,10;"9 PANTALLA"
;AT 15,10;"9 IMPRESORA";#1;AT 1,
4;"Pulse la opcion deseada"
4006 PAUSE 0

```

```

4008 IF INKEY$="P" THEN BEEP 0.2
,10: CLS : LET periferico=2: GO
TO 4020
4010 IF INKEY$="I" THEN BEEP 0.2
,10: CLS : LET periferico=3: GO
TO 4014
4012 BEEP 0.2,-10: GO TO 4006
4014 PRINT AT 21,0; INVERSE 1;"
CONECTE LA IMPRESORA
4016 PAUSE 100: PRINT #1;" Pulse
una tecla para continuar": PAUS
E 0: BEEP 0.2,10: CLS
4020 PRINT #periferico;"FICHERO
"e$:"FECHA","f$
4022 FOR n=1 TO ultimo
4030 PRINT #periferico; INVERSE
1;n; INVERSE 0;TAB 4;a$(n,1 TO 2
0)TAB 4;a$(n,21 TO 40)
4040 NEXT n
4042 IF periferico=3 THEN FOR n=
1 TO 15: LPRINT : NEXT n
4044 BEEP 0.2,10
4050 PRINT #1;AT 1,1;"Pulse una
tecla para retornar"
4060 PAUSE 0: BEEP 0.2,10: GO TO
500

```

Definición

La sentencia "FORMAT" permite entre otras cosas, formatear o inicializar un cartucho.

Su estructura general es:

| SENTENCIA | ARGUMENTO |
|-----------|-----------------------|
| FORMAT | "m"; unidad; "nombre" |

Ejemplos:

- FORMAT "m"; 1; "BAS"
- FORMAT "m"; 5; "tron"
- FORMAT "m"; 3; "Aster"
- FORMAT "m"; 8; "BUG"

El indicativo "m" hace referencia al *canal* seleccionado (Microdrive).

El parámetro *unidad* es un número entero comprendido entre 1 y 8, e indica la unidad de Microdrive que deseamos seleccionar.

El parámetro "nombre" es una cadena formada por un máximo de 10 caracteres, e indica al nombre asignado al cartucho; este se visualiza al pedir el *directorio* del cartucho.

El formateado de un cartucho "virgen" debe ser realizado antes de su utilización.

Mientras se realiza la inicialización de un cartucho, el Sistema Operativo chequea las zonas donde no puede LEER o ESCRIBIR, marcándolas, para tenerlas en cuenta en operaciones de acceso posteriores.

ADVERTENCIA

- Al formatear un cartucho con información, ésta se destruye.

- No debe sacarse un cartucho cuando el "LED" indicativo de acceso se encuentra encendido.

- No debe desconectarse el Spectrum con un cartucho insertado en una unidad Microdrive.

En los dos últimos casos deberá procederse a una nueva inicialización del cartucho.

La capacidad de almacenamiento libre de un cartucho formateado es superior a 85

Kbytes, pudiendo llegar hasta 100.

100 Kbytes = 102400 bytes

CAT

Acceso al teclado

GRAPHICS



MODO **E**

CAT

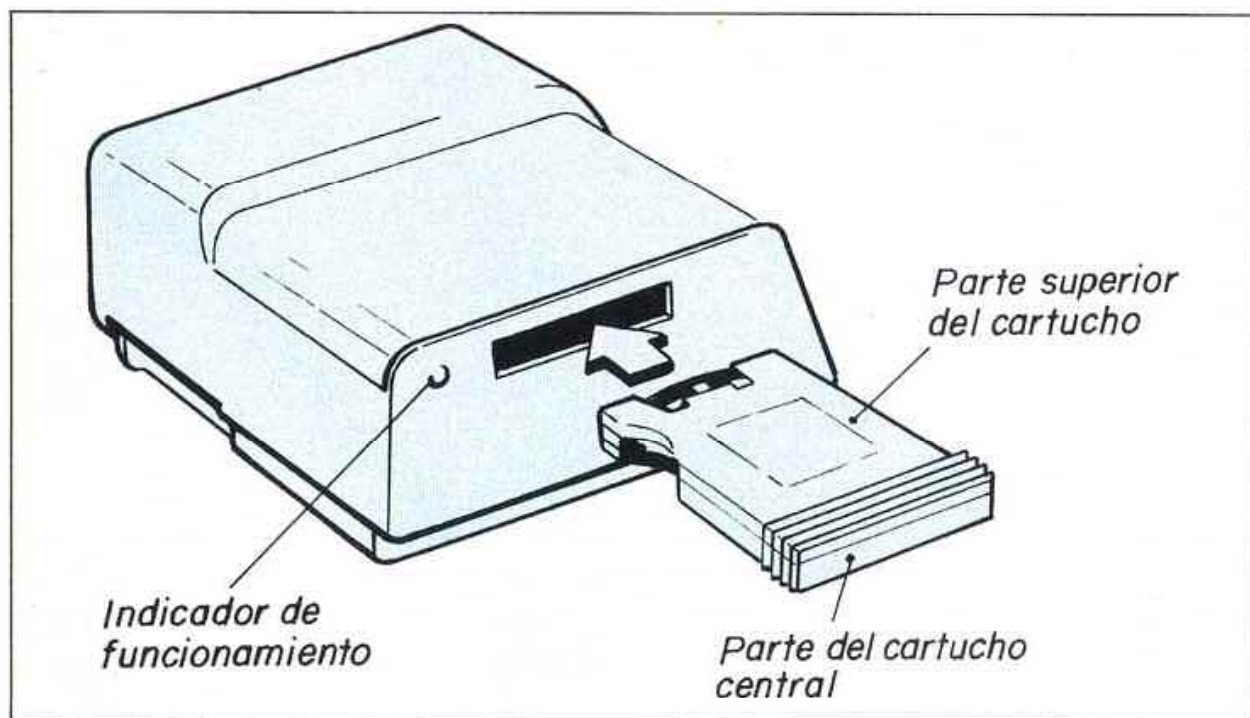
SYMBOL SHIFT

Definición

Permite obtener una lista o catálogo de todos los ficheros almacenados en un cartucho.

Su estructura general es:

| SENTENCIA | ARGUMENTO |
|-----------|-----------|
| CAT | unidad |



Unidad de Microdrive mostrando la correcta inserción del cartucho.

Ejemplos:

- CAT 5
- CAT j
- CAT (n - 1) * 2
- CAT VAL b\$

La forma de presentar un catálogo o directorio es la siguiente:

- Nombre asignado al formatear.
- Lista ordenada por orden alfabético de todos los ficheros.
- Capacidad libre expresada en Kbytes.

Utilizando otra estructura se puede enviar el catálogo a otro canal de comunicación.

| SENTENCIA | ARGUMENTO |
|-----------|---------------------|
| CAT | # corriente; unidad |

Ejemplo:

| |
|------------|
| CAT # 3; 5 |
|------------|

La anterior instrucción envía el catálogo de la unidad Microdrive número 5 a una impresora del tipo ZX.

Grabación y carga

Todas las opciones de grabación y carga de programas en cassettes están disponibles para su utilización con el Interface-1, solamente varía la sintaxis de la instrucción.

GRABACION

| SENTENCIA | ARGUMENTO |
|-----------|-------------------------|
| SAVE | * "m"; unidad; "nombre" |

Ejemplos:

- SAVE * "m"; 2; "cuat"
- SAVE * "m"; 3; "polo"

El símbolo del asterisco indica al sistema operativo que la grabación debe efectuarse a través del Interface -1, y no del cassette.

VERIFICACION

| SENTENCIA | ARGUMENTO |
|-----------|-------------------------|
| VERIFY | * "m"; unidad; "nombre" |

Ejemplos:

- VERIFY * "m"; 8; "vale"
- VERIFY * "m"; 1; "OKEY"

CARGA

| SENTENCIA | ARGUMENTO |
|-----------|-------------------------|
| LOAD | * "m"; unidad; "nombre" |

Ejemplos:

- LOAD * "m"; 5; "Pelota"
- LOAD * "m"; 3; "ZX"

COMBINACION

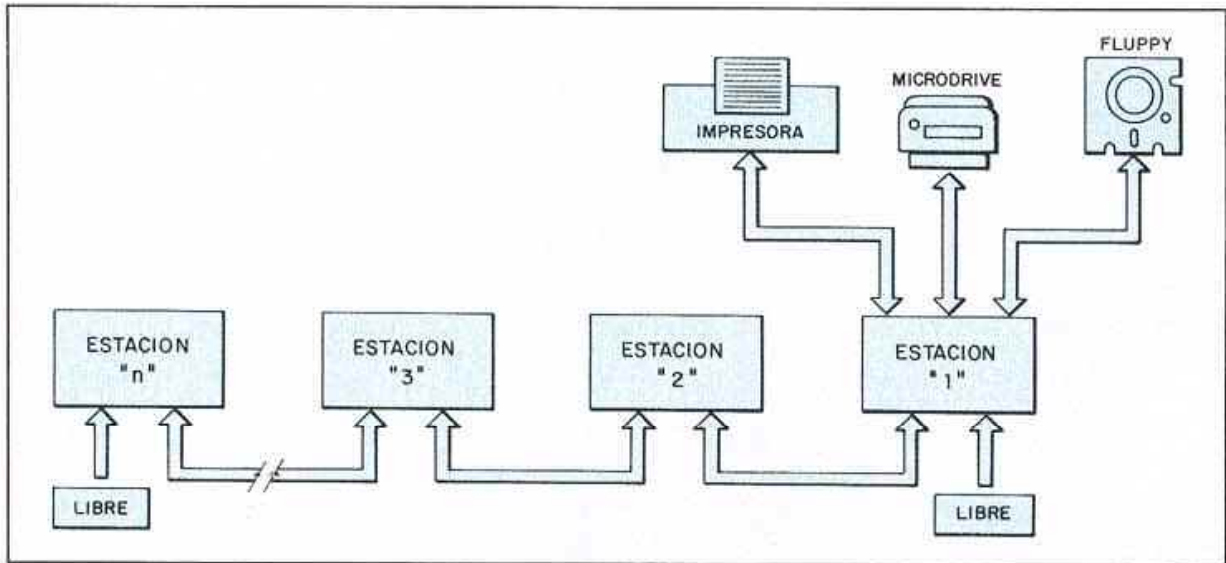
| SENTENCIA | ARGUMENTO |
|-----------|-------------------------|
| MERGE | * "m"; unidad; "nombre" |

Ejemplos:

- MERGE * "m"; 7; "GO"
- MERGE * "m"; 2; "toro"

Borrado de programas

Una operación engorrosa es el borrado de programas en cassette, sin embargo, cualquier programa grabado en Microdrive puede ser borrado con facilidad utilizando el comando "ERASE", que se encarga, entre otras cosas,



Estructura de conexionado.

de modificar el catálogo para que no se visualice el nombre del programa borrado.

ERASE

Acceso al teclado

WHITE



ERASE

Su estructura general es:

| SENTENCIA | ARGUMENTO |
|-----------|-----------------------|
| ERASE | "m"; unidad; "nombre" |

Ejemplos:

- ERASE "m"; 1; "pepe"
- ERASE "m"; 5; "BANK"

Ejecución automática

Utilizando el Microdrive y siguiendo unas cuantas reglas podrá ejecutar de forma automática aquel programa

que más utilice de un cartucho.

El programa deberá grabarse de la siguiente forma:

SAVE * "m"; 1; "run" LINE n

donde "n" es el número de línea para su autoejecución.

Para utilizar la facilidad de ejecución automática tendrá que:

- Insertar el cartucho en la unidad primera del Microdrive.

- Utilizar el programa después de haber conectado la alimentación del Spectrum o después de haber introducido el comando "NEW".

Teclee el comando "RUN" sin ningún argumento y el programa grabado con el nombre "run" se cargará en la memoria del Spectrum y se autoejecutará.

Protección de ficheros

Básicamente existen tres métodos para proteger sus programas:

a) Quitar la lengüeta de plástico situada en uno de los

laterales del cartucho. De esta manera no se podrán regrabar o borrar ningún programa.

b) Grabar los programas a proteger con "LINE n". De esta manera no se podrá realizar "MERGE" para visualizar su contenido.

c) Al realizar la grabación de un fichero anteponga el código 0 al nombre

SAVE * "m"; 1; CHR\$ 0 + "nombre"

Cuando intente obtener un catálogo del cartucho, el nombre de ese fichero no se visualizará, por tanto deberá recordarlo para poder cargarlo en otra ocasión.

LOAD * "m"; 1; CHR\$ 0 + "nombre"

Ficheros de datos

Aparte de los ficheros de programas, el usuario puede abrir y cerrar ficheros de datos tanto para grabación como para lectura.

Apertura de ficheros

Utilizando la sentencia "OPEN # " podemos asignar un fichero Microdrive a una corriente; si el fichero no existe, el sistema operativo interpreta que es de *escritura*; por el contrario, si existe, es de *lectura*.

Por ejemplo:

```
OPEN # 5; "m"; 1; "basic"
```

Grabación de datos

Podremos grabar datos en un fichero siempre y cuando haya sido abierto para escritura.

Para grabar datos en un fichero se utiliza la sentencia PRINT.

| SENTENCIA | ARGUMENTO |
|-----------|--------------------|
| PRINT | # corriente; datos |

Ejemplos:

- PRINT # 5; 20
- PRINT # 5; 3'7'8
- PRINT # 5; a\$
- PRINT # 5; 89'b\$n\$

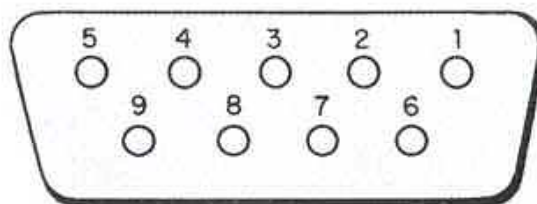
Para una posterior lectura, los datos deben ir seguidos del código "ENTER"; para ello introduzcamos los datos uno a uno.

```
PRINT # 5; 30  
PRINT # 5; 15  
PRINT # 5; 7
```

o utilice el signo del apóstrofe para separarlos

```
PRINT # 5; 30'15'7
```

Los datos no se graban directamente en el cartucho, si-



CONECTOR "HEMBRA" TIPO "D" 9 TERMINALES

| TERMINAL | NOMBRE |
|----------|-----------------------------|
| 1 | NC (NO CONECTADO) |
| 2 | TD (TRANSMIT DATA) |
| 3 | RD (RECEIVE DATA) |
| 4 | DTR (DATA TERMINAL READY) |
| 5 | CTS (CLEAR TO SEND) |
| 6 | NC |
| 7 | MASA |
| 8 | NC |
| 9 | +9V |

Conexión "RS 232".

no que se almacenan en una memoria intermedia de 512 bytes (0,5 Kbytes), cuando ésta se llena, se realiza la transferencia al Microdrive. La grabación se realiza por bloques.

Cierre de ficheros

Cuando la introducción de datos se termina, el fichero debe cerrarse, ya que de lo contrario no podría efectuarse una lectura posterior.

Al cerrar un fichero se transfieren al Microdrive los datos que estuvieran almacenados en la memoria intermedia.

La forma de cerrar un fichero se realiza utilizando la sentencia "CLOSE #".

Ejemplo:

```
CLOSE # 5
```

Lectura de ficheros

Para leer datos de un fichero es necesario que esté abierto para lectura.

Los datos se leen utilizando la sentencia "INPUT".

| SENTENCIA | ARGUMENTO |
|-----------|------------------------|
| INPUT | # corriente; variables |

- INPUT # 5; a
- INPUT # 5; b\$
- INPUT # 5; c; n\$; k
- INPUT # 5; LINE T\$

Si se leen varios datos seguidos, estos deben ir separados por "punto y coma".

```
INPUT # 5; d, e, n; k$
```

En la lectura de datos de cadena es conveniente utilizar el siguiente formato:

```
INPUT # 5; LINE c$
```

ya que si la cadena contiene comillas, interpretará que son el final de la cadena.

Ampliación de ficheros

Para realizar una ampliación de un fichero de datos ya creado, deberá seguir los siguientes pasos:

- Abrir el fichero antiguo para lectura.

CANALES DE COMUNICACION

"k" = Teclado

"s" = Pantalla

"p" = Impresora

"m" = Microdrive

"n" = Red de area local

"t" = RS232 Texto

"b" = RS232 Binario

CORRIENTES

```
#0 Salida y entrada de datos
#1 de la parte inferior de
  la pantalla.
#2 Salida de datos de la parte
  superior de la pantalla.
#3 Salida de datos a través
  de la impresora.
#4-15 Libres para ser definidos
  por el usuario.
```

- Abrir un nuevo fichero para escritura.

- Copiar el fichero antiguo en el nuevo.

- Editar a continuación los datos suplementarios en el fichero nuevo.

- Cerrar los dos ficheros.

- Borrar el fichero antiguo.

Ejemplo:

Tenemos un fichero llamado "DATOS" con 20 datos y deseamos incluir cinco más:

```
10, 84, 32, 15 y 49
```

- a) Abrir fichero antiguo

```
OPEN # 4; "m"; 1; "DATOS"
```

- b) Abrir fichero nuevo

```
OPEN # 5; "m"; 1; "DATOS.1"
```

- c) Copiar fichero

```
FOR n = 1 TO 5
INPUT # 4; dato
PRINT # 5; dato
NEXT n
```

```

BASIC

ASTEROIDE
EDIT
ENSAMBLE
ESTELAR
G.D.U.
KILFREE
LISTADOR
MARTE.COD
PRONOFF
QUERTY
START
TRON.BAS
TRON.COD
VELOZ
ZZZ.ZZZ

24

```

Catálogo de un cartucho "microdrive".

d) Editar nuevos datos

```

PRINT # 5; 10'84'32
PRINT # 5; 15'49

```

e) Cerrar ficheros

```

CLOSE # 4
CLOSE # 5

```

f) Borrar fichero antiguo

```

ERASE "m"; 1; "DATOS"

```

Programa

El programa número "1" permite generar un fichero Bibliográfico de hasta 100 títulos de libros con sus correspondientes autores.

Tanto los nombres de los libros como sus autores deberán tener una longitud máxima de 20 caracteres.

Para salir de la opción "1" deberá teclearse la palabra

```

FICHERO LIBROS
FECHA 21/06/85

1 LA REGENTA
CLARIN

2 1984
ORWELL

3 EL PADRINO
MARIO PUZO

4 RIMAS Y LEYENDAS
BECQUER

5 MACBETH
SHAKESPEARE

```

Programa "Fichero".

"FIN" como título de libro.

La opción "2" permite grabar el fichero en un cartucho Microdrive. Los datos que debemos dar al programa son:

NOMBRE: nombre del fichero hasta seis caracteres, ya que el programa incluye cuatro más, como extensión del mismo.

FECHA: Hasta un máximo de ocho caracteres.

CLAVE: Tiene un máximo de diez caracteres, y permite proteger el fichero contra posibles lecturas de personas no autorizadas.

Los datos se graban en dos ficheros distintos

```

NOMBRE .CLV
NOMBRE .DAT

```

En el primero se graban los datos correspondientes al nombre del fichero, la fecha, la clave y el último elemento almacenado.

En el segundo los datos ge-

nerados.

La opción "3" permite leer un fichero generado anteriormente para su ampliación o visualización.

Deberemos proporcionar el nombre sin *extensión* y la clave.

Si la clave almacenada en el fichero con extensión "CLV" no coincide con la teclada, el programa se autodestruye, no permitiendo el acceso a personas que no conozcan la clave.

Por último la opción "4" permite visualizar un fichero, bien sea en pantalla o impresora.

```

MOVE

```

Acceso al teclado

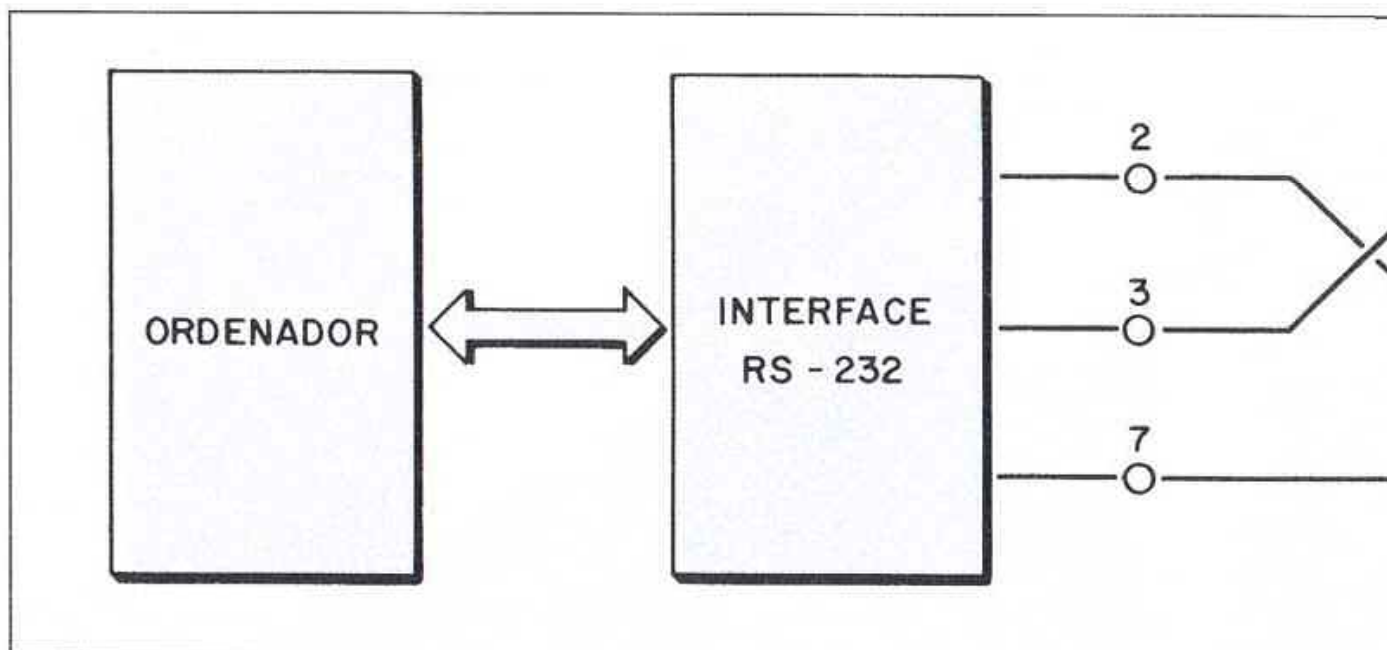
YELLOW



MODO E



MOVE SYMBOL SHIFT



Conexión simple entre interfaces RS 232.

Definición

La sentencia "MOVE" transfiere los datos de un canal de comunicación (fuente) a otro distinto (destino).

La estructura general es:

| SENTENCIA | ARGUMENTO |
|-----------|-------------------|
| MOVE | fuelle TO destino |

Ejemplos:

- MOVE # 1 TO # 2
- MOVE "m"; 1; "as" TO # 3
- MOVE "m"; 1; "Tn" TO "m"; 2; "Tn"
- MOVE # 1 TO # 3

La "fuente" y el "destino" pueden ser canales o números de corriente.

Esta sentencia es muy útil ya que se puede realizar:

- a) Copias de ficheros de datos

| |
|-------------------------------------|
| MOVE "m"; 1; "CAN" TO "m"; 1; "CAN" |
| MOVE "m"; 1; "SET" TO "m"; 2; "SET" |

- b) Visualización de ficheros

MOVE "m"; 1; "TOR" TO # 2

- c) Impresión de ficheros

MOVE "m"; 1; "LAR" TO # 3

Con "MOVE" no se pueden realizar copias de programas, solamente de ficheros de datos.

"MOVE" se encarga de abrir y cerrar los ficheros, por tanto no es necesario incluir las sentencias "OPEN # " ni "CLOSE # ".

La red era local

La red de área local o LAN (Local Area Network) permite conectar hasta 64 Spectrum formando una red de comunicación.

Con el uso de la red se pueden establecer comunicaciones entre los diversos usuarios o estaciones.

Los periféricos comunes (Microdrive, impresora, floppy, etc.) pueden estar conectados a una sola estación, de

manera que todas las demás puedan hacer uso de ellos.

Una vez establecida la red, con sus correspondientes cables, es necesario identificar cada estación, para ello utilice la sentencia "FORMAT".

| SENTENCIA | ARGUMENTO |
|-----------|---------------|
| FORMAT | "n"; estación |

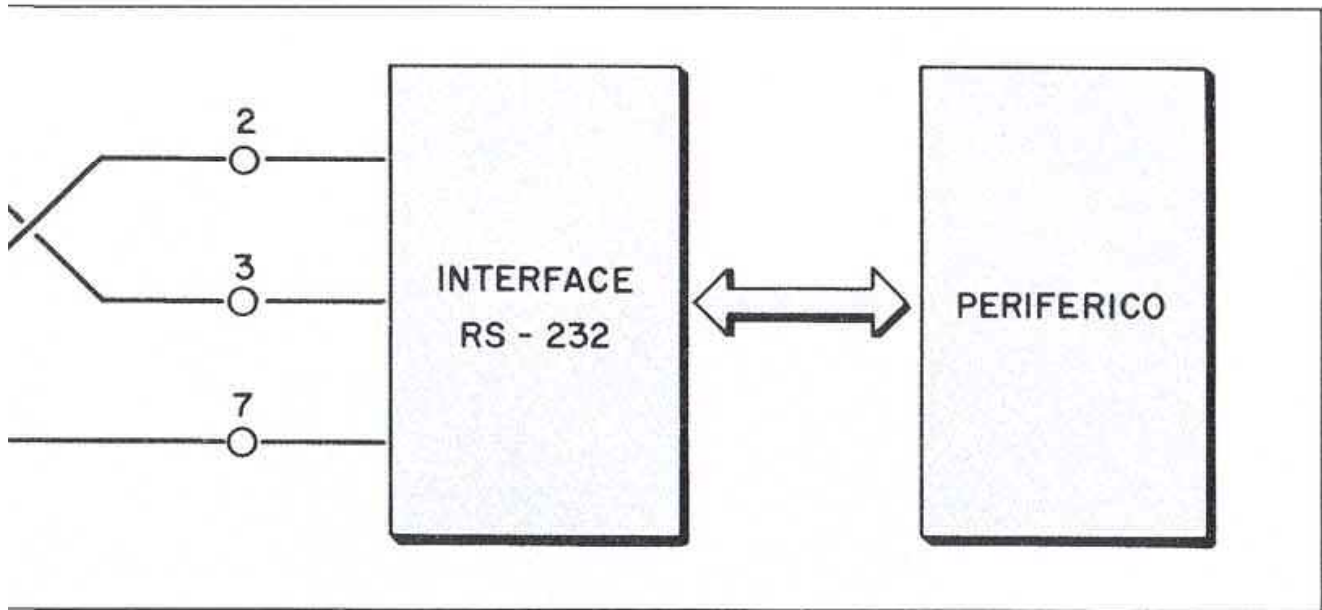
Ejemplos:

- FORMAT "n"; 2
- FORMAT "n"; VAL N\$
- FORMAT "n"; h
- FORMAT "n"; a * 5

La cadena "n" es el indicativo del canal correspondiente a la red, y "estación" en un número entero entre 1 y 64.

Para enviar o recibir programas de la red se utilizan las sentencias "SAVE", "VERIFY", "LOAD" y "MERGE" con las siguientes variaciones:

| |
|-------------------------------|
| SAVE * "n"; estación destino |
| VERIFY * "n"; estación fuente |
| LOAD * "n"; estación fuente |
| MERGE * "n"; estación fuente |



Como podrá observar, la red no utiliza nombres de programas.

Cuando se desea establecer un diálogo entre una estación y todas las demás, simultáneamente, se utiliza el canal de difusión.

Las estaciones receptoras deben teclear:

```
LOAD * "n"; 0
```

y la transmisora

```
SAVE * "n" 0
```

Interface RS-232

Con el uso del Interface RS-232, el Spectrum puede dialogar con cualquier periférico que utilice este protocolo de comunicación serie (Impresoras, modem, otro ordenador, etc.).

Es necesario realizar algunos ajustes en el periférico antes de utilizarlo.

– Desactivar el *avance de línea automático*.

– Desactivar el *control de paridad*.

– Seleccionar *ocho bits*.

– Seleccionar un solo *bit de parada*.

– Seleccionar la *velocidad de transmisión*, expresada en *baudios* (bits por segundo), entre cualquiera de las mostradas a continuación.

El Interface RS-232 utiliza dos canales de comunicación:

– El "t" para enviar textos en ASCII.

– El "b" para enviar códigos de ocho bits en binario.

Para utilizar cualquiera de los dos canales primero debe indicar al Spectrum la velocidad de transmisión a la que ha ajustado el periférico.

Ejemplos:

```
FORMAT "t"; 2400
FORMAT "b"; 600
```

El canal "t" envía el código ASCII de acuerdo al siguiente formato:

– Los códigos del 0 al 31 no se envían, excepto el código 13 (ENTER) que se envía seguido del 10 (cursor abajo) es decir, lo que se conoce por "retorno de carro" y "avance de línea".

– Los códigos del 32 al 127 ("espacio" al "copyright") se envían sin ninguna alteración.

– Los códigos 128 al 164 (gráficos) no se envían, en su lugar se envía el código 63 (?).

– Los códigos del 165 al 255 (tokens) se expanden a los códigos 32-127.

Para enviar un listado a una impresora con Interface RS-232 y velocidad de transmisión de 300 baudios, se utilizaría:

```
FORMAT "t"; 300
OPEN # 5; "t"
LIST # 5
```

Para recibir datos desde un terminal transmisor con una

velocidad de 9600 baudios, seria:

```
10 FORMAT "t": 9600
20 OPEN # 6: "t"
30 PRINT INKEY$ # 6;
40 GOTO 30
```

El canal "b" se utiliza para enviar los códigos de control a las impresoras, para utilizar un *modem*, etc.

Con el canal «b» se pueden utilizar las sentencias "SAVE" y "LOAD".

```
SAVE * "b"
LOAD * "b"
```

dependiendo de si es transmisor o receptor.



El Spectrum, con el nuevo dispositivo (interface-1) y dos unidades de microdrive, puede acceder a 180 K de memoria externa.

Para enviar un código de control a una impresora, por ejemplo el 18, se utilizaría:

```
OPEN # 7: "b"
PRINT # 7: CHR$ 18
CLOSE # 7
```

LA MEMORIA

Todo lo que introducimos en el ordenador, tiene que almacenarse dentro de alguna forma. Para ello todos los ordenadores disponen de lo que se denomina *memoria*.

Podemos imaginar la memoria de nuestro ordenador como un conjunto de 65536 cajitas, cada una de ellas puede contener un número comprendido entre 0 y 255. Cada cajita tiene, a su vez, un número que actúa como "nombre propio" y la distingue de las demás; la primera casilla se llama "0" y la última "65535". Este número se de-

nomina "Dirección" y el número que contienen se denomina "Dato".

El dato contenido en las primeras 16384 cajitas es fijo, no podemos alterarlo; es como si estas cajitas estuvieran cubiertas por un cristal, podemos ver el número que hay en su interior pero no podemos cambiarlo por otro.

En las restantes cajitas, si podemos alterar el contenido. Veamos ahora las sentencias de Basic que trabajan directamente sobre la memoria.

POKE

Acceso al teclado

PEEK

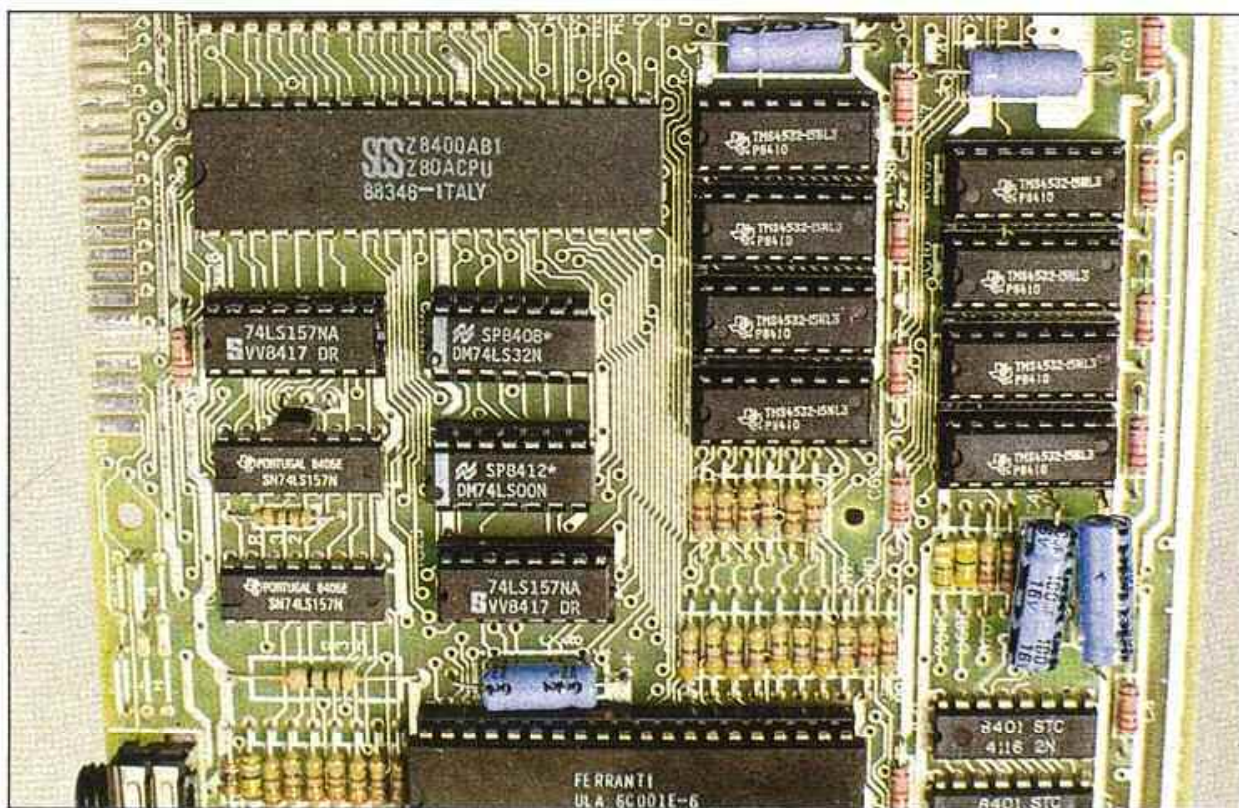


OUT

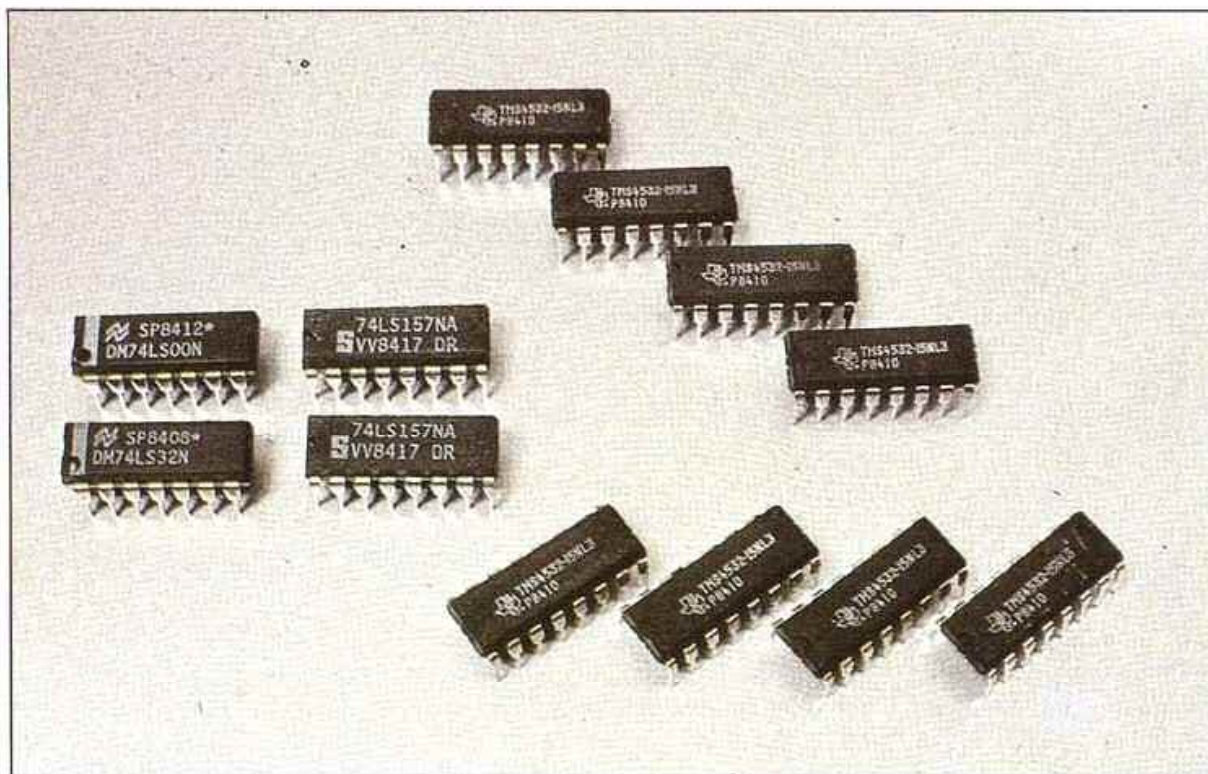
Definición

El comando POKE almacena un dato en una dirección de memoria.

Su estructura general es:



Los "chips" que componen la memoria del Spectrum.



Aspecto de la memoria del Spectrum compuesta por 16 chips de RAM dinámica y una de ROM.

| SENTENCIA | ARGUMENTO |
|-----------|-----------------|
| POKE | Dirección, Dato |

Ejemplos:

- POKE 23300,255
- POKE 65342,a
- POKE d+38, -37
- POKE h+2, a-25

La dirección ha de estar comprendida entre 0 y 65535 (si bien, intentar "POKEar" en una dirección menor de 16384, no tiene sentido) y el dato ha de estar entre -255 y 255 (un dato negativo equivale a 256 menos ese número).

Si alguno de estos números estuviera fuera de este margen, se produciría el error:

B Integer out of range

La razón de que no tenga sentido "POKEar" en direcciones inferiores a 16384 es que, como hemos dicho an-

tes, el contenido de estas direcciones no se puede alterar.

PEEK

Acceso al teclado

PEEK



MODOS E

OUT

Definición

La función PEEK devuelve el contenido de una posición de memoria, cuya dirección es el argumento de PEEK.

Su estructura general es:

| SENTENCIA | ARGUMENTO |
|-----------|-----------|
| PEEK | Dirección |

Ejemplos:

- PRINT PEEK 23657
- LET a=PEEK (60000+b)
- LET a=PEEK b
- PRINT 17+PEEK 56432

La dirección puede ser cualquier número entero comprendido entre "0" y "65535". Si estuviera fuera de este margen, se produciría el error:

B Integer out of range

Las sentencias POKE y PEEK nos dan un inmenso poder sobre el ordenador al actuar directamente sobre la memoria, ya que todo lo que hace el ordenador depende de los datos almacenados en su memoria.

Tipos de memoria

Existen un gran número de

tipos de memoria, ROM, RAM, PROM, EPROM, EAROM, etc. Nuestro ordenador dispone de 16384 (16 K) direcciones o posiciones de memoria ROM y 49152 (48 K) de memoria RAM (en la versión de 16 K, son sólo 16384 posiciones de RAM).

ROM significa "Read Only Memory" (Memoria de sólo lectura), como su nombre indica, es una memoria que sólo se puede leer, y en la que no se puede escribir, su contenido ha sido grabado en fábrica de forma indeleble.

RAM significa «Random Access Memory» (Memoria de acceso aleatorio), en realidad, la ROM también es de acceso aleatorio, ya que podemos leer cualquiera de sus datos sin leer los precedentes, pero a la RAM se la llama así para distinguirla de la ROM, ya que en memoria RAM podemos tanto leer como escribir.

El mapa básico de memoria para ambas versiones sería el representado en la FIGURA 1.

Bit y byte

Hasta ahora hemos dicho que los números se almacenan en las posiciones de memoria, pero no hemos explicado la forma en que se almacena un número determinado.

Se puede imaginar que cada posición de memoria (cajita) es una fila de ocho interruptores, a un interruptor encendido le llamamos "1" y a uno apagado le llamamos "0". Cada número comprendido entre "0" y "255" se forma por combinación de "unos" y "ceros", es lo que se

posición de memoria almacena un byte.

La correspondencia entre cada número y su combinación binaria, se haya sabiendo que cada bit tiene un valor determinado según su situación en el byte. Empezando por la derecha, el primer bit vale "1", el segundo vale "2", el tercero "4", y el cuarto "8", y así sucesivamente, cada uno vale el doble del anterior, hasta el octavo que vale "128". Para hayar el valor decimal de un número binario, se suman los valores de los bits que están a "1". Veamos unos cuantos ejemplos:

| 128 | 64 | 32 | 16 | 8 | 4 | 2 | 1 | | |
|-----|----|----|----|---|---|---|---|------------------------|-------|
| 1 | 0 | 0 | 1 | 1 | 0 | 1 | 0 | = 128+16+8+2 | = 154 |
| 0 | 1 | 1 | 0 | 0 | 1 | 0 | 1 | = 64+32+4+1 | = 101 |
| 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | = 128+64+32+16+8+4+2+1 | = 255 |
| 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | = 8 +4+2+1 | = 15 |

llama "Notación Binaria". A cada "1" o "0" se le denomina "Bit" (abreviatura de "Binary Digit"), y al conjunto de ocho "unos" o "ceros" se le denomina "Byte", por tanto, cada

Memoria ROM

El motivo de que la memoria ROM venga grabada de fábrica y sus datos no puedan ser

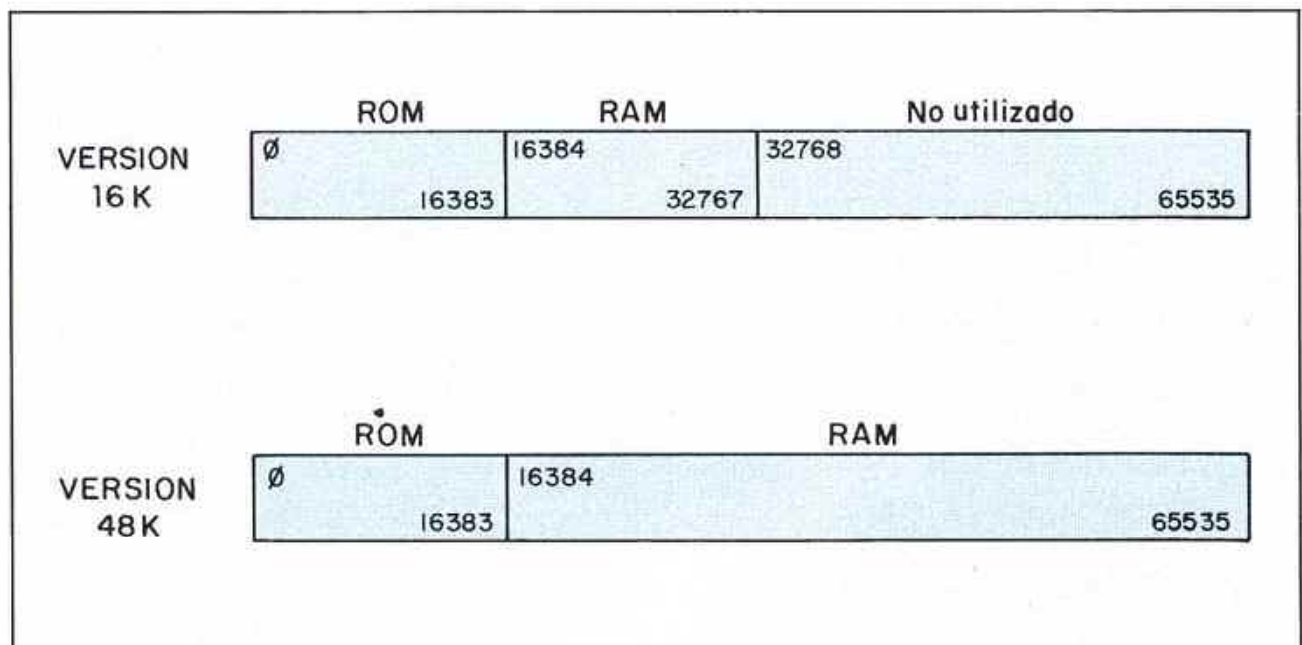


Figura 1. Mapa de memoria básico de ambas versiones del Spectrum.

| | | BITS | | | | | | | | | |
|-----------|--|------|----|----|----|---|---|---|---|----------------|--|
| | | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | | |
| DIRECCION | | 128 | 64 | 32 | 16 | 8 | 4 | 2 | 1 | CODIGO DECIMAL | |
| 15880 | | | | | | | | | | 0 | |
| 15881 | | | | | | | | | | 60 | |
| 15882 | | | | | | | | | | 66 | |
| 15883 | | | | | | | | | | 66 | |
| 15884 | | | | | | | | | | 126 | |
| 15885 | | | | | | | | | | 66 | |
| 15886 | | | | | | | | | | 66 | |
| 15887 | | | | | | | | | | 0 | |

Figura 3. La letra "A" tal como está definida en la ROM del Spectrum

alterados, es que contiene lo que se denomina el "SISTEMA OPERATIVO" del ordenador, un conjunto de programas escritos en código máquina que permiten al ordenador operar desde el mismo momento en que se conecta.

También contiene la ROM el juego de caracteres, se encuentran a partir de la dirección 15616 y están definidos de la misma forma que los UDG. Como ejemplo, la letra "A" está definida de la forma que puede verse en la FIGURA 3.

Parte del Sistema Operativo del ordenador, lo constituye el denominado "INTERPRETE DE BASIC" que es precisamente, el programa que nos

permite utilizar este lenguaje para programar el ordenador.

La memoria RAM

En las direcciones que están a continuación de la ROM, se encuentra la memoria RAM. No toda ella está a nuestra disposición, ya que una parte la necesita el ordenador para sus propios datos.

Los primeros 6144 bytes (desde la dirección 16384 hasta la 22527) están ocupados por el archivo de presentación visual, en esta zona se encuentra almacenado todo lo que vemos en la pantalla.

Los 768 bytes siguientes

(desde la dirección 22528 hasta la 23295) contienen el archivo de atributos, que almacena los colores de todos los caracteres de la pantalla.

Entre la dirección 23296 y la 23551 (256 bytes) se encuentra la memoria intermedia de impresora. A continuación, entre la 23552 y la 23733 (182 bytes) están las variables del sistema.

A partir de la dirección 23734 se encuentra el área de información para canales; en la versión básica, esta zona ocupa 21 bytes, pero se expande al conectar el INTERFACE 1 y trabajar con MICRODIVE o con la ZX-NET.

A continuación, viene la zona donde almacenamos el

programa Basic y las variables, la longitud de estas zonas no es fija ya que se van expandiendo a medida que vamos almacenando datos en ellas. Las direcciones de inicio de cada una de estas zonas están contenidas en las variables del sistema CHANS, PROG y VARS que se actualizan continuamente por el sistema operativo.

Hay una serie de zonas más cuyas longitudes y direcciones de inicio tampoco son fijas, pero se encuentran anotadas en las correspondientes variables del sistema. Estas zonas son: Area de edición, zona de trabajo y pila del calculador.

A continuación viene una zona llamada de reserva, que contiene los bytes que no han sido utilizados, y por tanto, se contrae cada vez que se expande cualquiera de las otras zonas. Este área es considerablemente mayor en la versión de 48 K que en la de 16 K.

Todas las zonas que hemos visto hasta ahora se expanden hacia "arriba", pero hay dos zonas por encima del área de reserva que se expanden hacia "abajo", se trata de las pilas de máquina y GO SUB.

Finalmente, los últimos 168 bytes de la RAM se encuentran ocupados por los Gráficos Definidos por el Usuario (UDG).

En la FIGURA 2 se puede ver el mapa de memoria del Spectrum, los números situados a la izquierda son direcciones, y los situados a la derecha indican la cantidad de bytes ocupados por la zona correspondiente en el momento de conectar el ordenador.

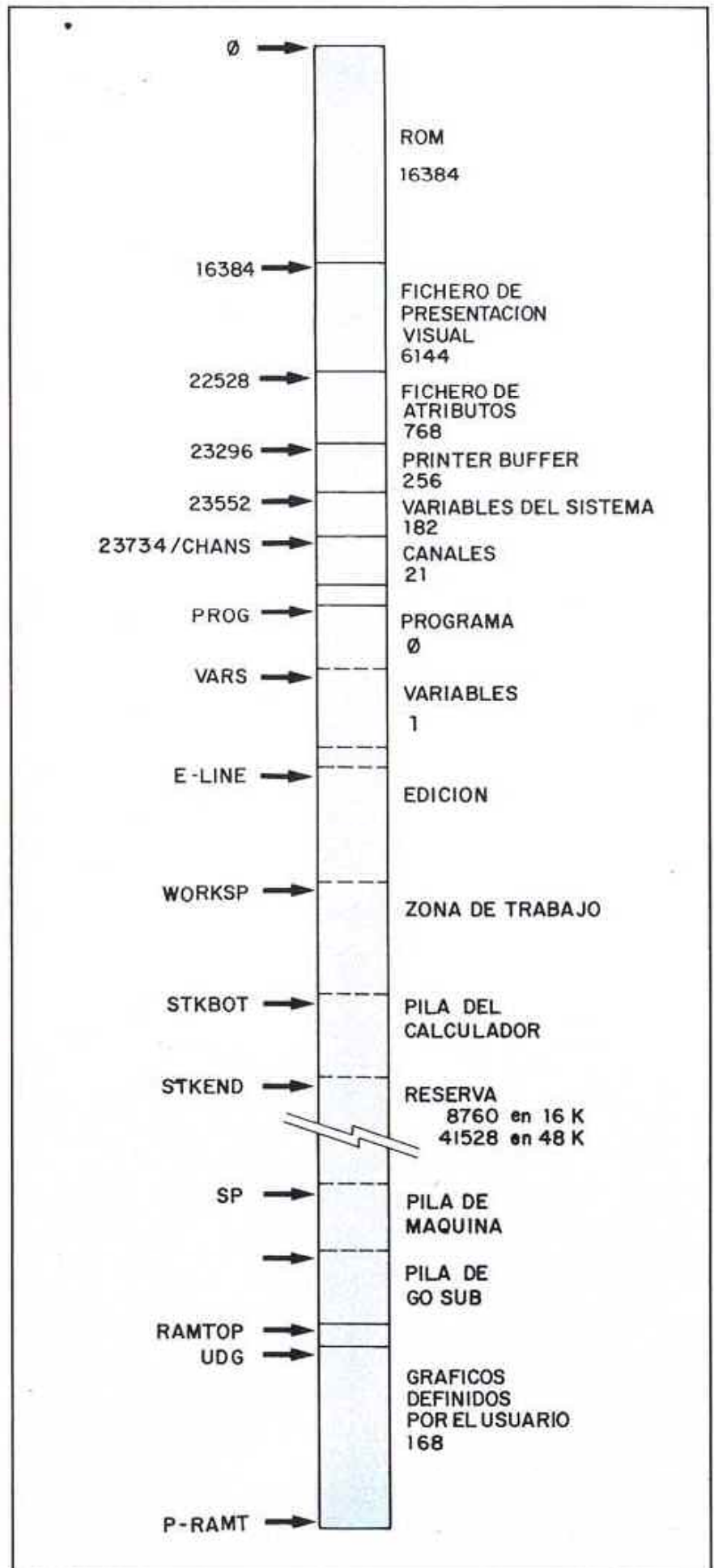


Figura 2. Mapa completo de memoria del Spectrum.

| DIRECCION | DATO | |
|-----------|------|---|
| PROG | 0 | } Número de linea |
| PROG + 1 | 10 | |
| PROG + 2 | 12 | } Longitud de la linea + ENTER (12 bytes) |
| PROG + 3 | 0 | |
| PROG + 4 | 241 | Código de LET |
| PROG + 5 | 97 | Código de "a" |
| PROG + 6 | 61 | Código de "=" |
| PROG + 7 | 50 | } Representación ASCII de "27" |
| PROG + 8 | 55 | |
| PROG + 9 | 14 | Indicador de número |
| PROG + 10 | 0 | } Representación en coma flotante de "27" |
| PROG + 11 | 0 | |
| PROG + 12 | 27 | |
| PROG + 13 | 0 | |
| PROG + 14 | 0 | |
| PROG + 15 | 13 | Código de ENTER (fin de linea) |

Figura 4. Una línea de programa en Basic, tal como se almacena en la zona de programa de la memoria RAM.

Almacenamiento de programas

Ahora vamos a ver como se almacenan en la memoria las líneas de Basic y las variables que introducimos cuando escribimos o cargamos un programa.

A partir de la dirección apuntada por la variable del sistema PROG, se almacena la primera línea del programa. En primer lugar, el número de línea que ocupa dos bytes (el más significativo primero); a continuación, viene la longitud de la línea que también ocupa dos bytes, pero con el orden invertido.

Los siguientes bytes constituyen el texto de la línea propiamente dicha, donde hay un número, está primero su representación ASCII, a continuación el código 14 y después la representación del número en coma flotante, que es la que realmente utiliza el ordenador.

Supongamos la siguiente línea:

```
10 LET a=27
```

En el interior de la memoria esta línea quedaría almacenada como se ve en la FIGURA 4.

Variables

La forma en que se almacenan las variables es algo más compleja, ya que depende del tipo de variable de que se trate.

El tipo de variable viene dado por la configuración de los tres primeros bits del primer byte, de la forma siguiente:

Variable numérica cuyo nombre es una sola letra: 011
 Variable numérica cuyo nombre son varias letras: 101
 Variable de cadena de caracteres: 010
 Variable de control de bucle FOR-NEXT: 111
 Matriz de números: 100
 Matriz de caracteres: 110

A continuación, vamos a ver cada una detenidamente y con algún ejemplo.

Variable numérica cuyo nombre es una sola letra

Ejemplo:

```
a=27
```

| | | |
|----|----|-----------------|
| a | 97 | 0 1 1 0 0 0 0 1 |
| 27 | 0 | 0 0 0 0 0 0 0 0 |
| | 0 | 0 0 0 0 0 0 0 0 |
| | 27 | 0 0 0 1 1 0 1 1 |
| | 0 | 0 0 0 0 0 0 0 0 |
| | 0 | 0 0 0 0 0 0 0 0 |

Variable numérica cuyo nombre son varias letras

En este caso, la última letra tiene el primer bit a "1" para indicar que es el fin del nombre.

Ejemplo:

```
abcd=27
```

| | | |
|----|-----|-----------------|
| a | 161 | 1 0 1 0 0 0 0 1 |
| b | 98 | 0 1 1 0 0 0 1 0 |
| c | 99 | 0 1 1 0 0 0 1 1 |
| d | 228 | 1 1 1 0 0 1 0 0 |
| 27 | 0 | 0 0 0 0 0 0 0 0 |
| | 0 | 0 0 0 0 0 0 0 0 |
| | 27 | 0 0 0 1 1 0 1 1 |
| | 0 | 0 0 0 0 0 0 0 0 |
| | 0 | 0 0 0 0 0 0 0 0 |

Variable de cadena de caracteres

Los dos bytes que siguen al nombre indican la longitud.

Ejemplo:

```
a$="HOLA"
```

| | | |
|----------|----|-----------------|
| A | 65 | 0 1 0 0 0 0 0 1 |
| Long = 4 | 4 | 0 0 0 0 0 1 0 0 |
| | 0 | 0 0 0 0 0 0 0 0 |
| H | 72 | 0 1 0 0 1 0 0 0 |
| D | 79 | 0 1 0 0 1 1 1 1 |
| L | 76 | 0 1 0 0 1 1 0 0 |
| A | 65 | 0 1 0 0 0 0 0 1 |

Variable de control de bucle FOR-NEXT

Los cinco primeros bytes que siguen al nombre indican el valor inicial en coma flotante, los cinco siguientes indican el límite, los cinco siguientes el "paso", los dos siguientes indican la línea donde se ha definido el bucle y el último indica el número de sentencia dentro de la línea.

Ejemplo:

```
10 FOR a=1 TO 7 STEP 2
```

| | | |
|------------|-----|-----------------|
| a | 238 | 1 1 1 0 1 1 1 0 |
| valor = 1 | 0 | 0 0 0 0 0 0 0 0 |
| | 0 | 0 0 0 0 0 0 0 0 |
| | 1 | 0 0 0 0 0 0 0 1 |
| | 0 | 0 0 0 0 0 0 0 0 |
| | 0 | 0 0 0 0 0 0 0 0 |
| límite = 7 | 0 | 0 0 0 0 0 0 0 0 |
| | 0 | 0 0 0 0 0 0 0 0 |
| | 7 | 0 0 0 0 0 1 1 1 |
| | 0 | 0 0 0 0 0 0 0 0 |
| | 0 | 0 0 0 0 0 0 0 0 |
| paso = 2 | 0 | 0 0 0 0 0 0 0 0 |
| | 0 | 0 0 0 0 0 0 0 0 |
| | 2 | 0 0 0 0 0 0 1 0 |
| | 0 | 0 0 0 0 0 0 0 0 |
| | 0 | 0 0 0 0 0 0 0 0 |

| | | |
|---------------|---------|------------------------------------|
| línea = 10 | 10 0 | 0 0 0 0 1 0 1 0 0 0 0 0 0 0 0 0 |
| sent. = 1 | 1 | 0 0 0 0 0 0 0 1 |

Matriz de números

Los dos bytes siguientes al nombre, almacenan la longitud de la variable más sus dimensiones, el siguiente byte contiene el número de dimensiones, a partir de ahí y de dos en dos bytes, se almacenan las dimensiones y a continuación, los elementos en coma flotante, ordenados según los subíndices.

Ejemplo:

```
DIM A(2,2): LET a(1,1)=15:
LET a(1,2)=26:
LET a(2,1)=12: LET a(2,2)=7
```

| | | |
|----------------|-------------------|--|
| a | 129 | 1 0 0 0 0 0 0 1 |
| long. = 25 | 25 0 | 0 0 0 1 1 0 0 1 0 0 0 0 0 0 0 0 |
| dim. = 2 | 2 | 0 0 0 0 0 0 1 0 |
| 1 dim. | 2 0 | 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 |
| 2 dim. | 2 0 | 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 |
| a(1,1) = 15 | 0 15 0 0 | 0 1 1 1 1 0 0 0 0 0 0 0 0 |
| a(1,2) = 26 | 0 26 0 0 | 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 0 1 0 0 0 0 0 0 0 0 0 |
| a(2,1) = 12 | 0 12 0 0 | 0 1 1 0 0 0 0 0 0 0 0 0 0 |
| a(2,2) = 7 | 0 7 0 0 | 0 1 1 1 0 0 0 0 0 0 0 0 |

Matriz de caracteres

Los dos primeros bytes que siguen al nombre, indican la longitud de la variable más sus dimensiones, el siguiente indica el número de dimensiones, a continuación vienen las dimensiones, y finalmente, el texto.

Ejemplo:

```
DIM a$(2,8): LET a$(1)="BASIC":
LET a$(2)="SINCLAIR"
```

| | | |
|---------------|---------|------------------------------------|
| a | 193 | 1 1 0 0 0 0 0 1 |
| long. = 21 | 21 0 | 0 0 0 1 0 1 0 1 0 0 0 0 0 0 0 0 |
| dim. = 2 | 2 | 0 0 0 0 0 0 1 0 |
| 1 dim. | 2 0 | 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 |
| 2 dim. | 8 0 | 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 |
| B | 66 | 0 1 0 0 0 0 1 0 |
| A | 65 | 0 1 0 0 0 0 0 1 |
| S | 83 | 0 1 0 1 0 0 1 1 |
| I | 73 | 0 1 0 0 1 0 0 1 |
| C | 67 | 0 1 0 0 0 0 1 1 |
| " " | 32 | 0 0 1 0 0 0 0 0 |
| " " | 32 | 0 0 1 0 0 0 0 0 |
| " " | 32 | 0 0 1 0 0 0 0 0 |
| S | 83 | 0 1 0 1 0 0 1 1 |
| I | 73 | 0 1 0 0 1 0 0 1 |
| N | 78 | 0 1 0 0 1 1 1 0 |
| C | 67 | 0 1 0 0 0 0 1 1 |
| L | 76 | 0 1 0 0 1 1 0 0 |
| A | 65 | 0 1 0 0 0 0 0 1 |
| I | 73 | 0 1 0 0 1 0 0 1 |
| R | 82 | 0 1 0 1 0 0 1 0 |

Borrado de variables

Ejecute el siguiente programa:

```
10 LET a=27
20 CLEAR
30 PRINT a
```

Verá que se detiene con el informe:

```
2 Variable not found, 30:1
```

La línea 30 no encuentra la variable "a" a pesar de haber sido definida en la línea 10. Lo que ocurre es que la variable ha sido borrada en la línea 20, ésta es una de las utilidades del comando CLEAR.

```
CLEAR
```

Acceso al teclado

LEN



Definición

El comando CLEAR borra la pantalla, las variables, restaura la posición de PRINT a la esquina superior izquierda, restaura la posición de PLOT a la esquina inferior izquierda, restaura el puntero de DATA y, caso de tener un argumento numérico, cambia la dirección de RAMTOP si ello fuera posible.

Su estructura general es:

| SENTENCIA | ARGUMENTO |
|-----------|--------------|
| CLEAR | Nueva RAMTOP |

Vuelva a mirar la figura 2 donde se muestra el mapa de memoria, casi al final verá una dirección apuntada por una variable que se llama RAM-

TOP. Es la dirección más alta que puede utilizar el Basic, algo así como un "límite" del sistema.

Podemos variar la posición de este límite, dentro de unos márgenes, lo cual puede ser muy útil en determinados casos.

Normalmente, por encima de RAMTOP sólo se encuentran los Gráficos Definidos por el Usuario, pero podemos bajar la RAMTOP y hacer sitio para colocar algo que no deseemos que el Basic pueda borrar, por ejemplo, un programa en código máquina. Lo que coloquemos por encima de RAMTOP queda a salvo de borrados incluso con NEW.

Si desea saber en qué dirección se encuentra la RAMTOP de su ordenador, teclee:

```
PRINT PEEK 23730+256*PEEK 23731
```

La respuesta normal será: 32599 para el modelo de 16 K, y 65367 para el de 48 K. Tenga en cuenta que algunos interfaces pueden alterar este valor.

Si desea bajar la RAMTOP teclee el comando CLEAR seguido de la nueva dirección. Por ejemplo, supongamos que su versión es de 16 K, en ese caso la RAMTOP estará en 32599. Supongamos ahora que quiere conseguir un espacio libre de 300 bytes, teclee:

```
CLEAR 32299
```

Ahora las direcciones libres van desde la 32300 hasta la 32599, ambas inclusive.

La dirección más baja que puede usar como argumento de CLEAR es 23821, y la más

alta es 32767 en la versión de 16 K y 65535 en la de 48 K. Si el argumento de CLEAR está fuera de este margen, obtendrá el informe:

```
M RAMTOP no good,
```

En este caso, CLEAR habrá hecho todo (borrar variables, restaurar punteros, etc.) menos cambiar la RAMTOP.

Cuando se llena la memoria

Es posible que haya tenido ya alguna experiencia de lo que ocurre cuando se llena la memoria de su ordenador, si jugamos un poco con CLEAR bajando mucho la RAMTOP, seremos capaces de verlo claramente. Si tiene el INTERFACE 1, desconéctelo para hacer estas pruebas. Teclee:

```
CLEAR 23850
```

Ahora intente pulsar cualquier tecla, verá que el ordenador no responde, y en su lugar, emite un «pitido» que dura un par de segundos. Hemos bajado tanto la RAMTOP que no le hemos dejado sitio al Basic para trabajar. No tendremos más remedio que desconectar el ordenador y volverlo a conectar de nuevo para que todo vuelva a la normalidad.

El "pitido" que hemos oído es la señal de alarma del Spectrum, indica que la memoria está totalmente llena. Ahora teclee:

```
CLEAR 23900
```

Ahora introduzca las siguientes líneas:

```
10 REM xxxxxxxxxxxx
20 REM xxxxxxxxxxxx
30 REM xxxxxxxxxxxx
```

No se asuste, si cuando ha ido a introducir la línea 30 el ordenador no la ha aceptado, y le ha respondido con el mensaje:

```
G No room for line. 0:1
```

Esto le indica que el área de Basic está llena, y no cabe ninguna línea más. Ahora haga NEW y teclee:

```
10 REM xxxxxxxxxxxxxxxx
20 REM xxxxxxxxxxxxxxxxxxxxxxxx
```

Esta vez, no sólo no ha podido introducir la línea 20, sino que ni siquiera ha podido terminarla, cuando llevaba 22 "x" pulsadas e iba a pulsar la 23, el ordenador se ha bloqueado de nuevo con el "pitido", la memoria está tan llena que no hay sitio suficiente en el área de edición para construir la nueva línea. Borre toda la línea 20 pulsando DELETE, e introduzca:

```
CLEAR 24000: NEW
```

Ahora teclee:

```
DIM a(10,10)
```

Al pulsar ENTER, el ordenador le responderá con el

RENUMERADOR - LISTADO ASSEMBLER

| Direcc. | Cód. Mq. | Listado Assembler |
|---------|--------------|-------------------|
| 23296 | 42,83,92 | LD HL,(PROG) |
| 23299 | 17,10,0 | LD DE,+0A |
| 23302 | 229 | BUCLE PUSH HL |
| 23303 | 237,75,75,92 | LD BC,(VARS) |
| 23307 | 237,66 | SBC HL,BC |
| 23309 | 40,18 | JR Z,(FINAL) |
| 23311 | 225 | POP HL |
| 23312 | 114 | LD (HL),D |
| 23313 | 35 | INC HL |
| 23314 | 115 | LD (HL),E |
| 23315 | 35 | INC HL |
| 23316 | 79 | LD C,(HL) |
| 23317 | 35 | INC HL |
| 23318 | 70 | LD B,(HL) |
| 23319 | 9 | ADD HL,BC |
| 23320 | 35 | INC HL |
| 23321 | 235 | EXX DE,HL |
| 23322 | 1,10,0 | LD BC,+0A |
| 23325 | 9 | ADD HL,BC |
| 23326 | 235 | EXX DE,HL |
| 23327 | 24,229 | JR (BUCLE) |
| 23329 | 207 | FINAL RST 0B |
| 23330 | 255 | DEFB +FF |
| | | PROG EQU +5C53 |
| | | VARS EQU +5C4B |

Figura 5. Un sencillo renumerador en Cdigo Mquina para el Spectrum.

mensaje.

4 Out of memory. 0:1

Hemos intentado dimensionar una matriz de 10 por 10, lo que requiere 508 bytes disponibles en el rea de variables, pero tenemos tan baja la RAMTOP que no hay espacio suficiente.

Este mensaje se presentar cada vez que intente hacer algo para lo que necesite ms memoria que la que tiene disponible.

Siempre que utilice CLEAR en un programa, recuerde que adems de bajar la RAMTOP, le borrar la pantalla y todas las variables que hubie-

ra definido hasta ese momento, adems de restaurarle las posiciones de PRINT y PLOT y el puntero de DATA, de la misma forma que si hubiera ejecutado un RESTORE y un CLS.

Puede utilizar CLEAR sin argumento, que har todo excepto modificar la RAMTOP.

Programando en código máquina

El Spectrum, al igual que la mayoría de los ordenadores, permite llamar desde el Basic a rutinas escritas en código máquina. El código máquina no es realmente un lenguaje de programación (el lenguaje correspondiente es el Assembler) sino el conjunto de números que, almacenados en las posiciones de memoria, le indican al microprocesador las operaciones que debe ir ejecutando.

En lenguajes de alto nivel, como el Basic, cada comando desencadena la ejecución de cientos de instrucciones en código máquina, pero puede haber cosas que no se pueden hacer en Basic, o que se hacen más deprisa en có-

digo máquina, para ello se ha previsto la función **USR**.

USR

Acceso al teclado

USR



ATTR

Definición

La función **USR** con argumento numérico, ejecuta las instrucciones en código máquina correspondientes a los

números almacenados a partir de la dirección apuntada por el argumento, hasta que se encuentre una instrucción **RET** (código 201), momento en el que devuelve como resultado el contenido del par de registros **BC** del microprocesador.

Su estructura general es:

| FUNCIÓN | ARGUMENTO |
|------------|-----------|
| USR | Dirección |

Los programas en código máquina, normalmente, se escriben primero en Assembler, y luego se traducen por medio de un programa que se conoce con el nombre de "Ensamblador".

En este caso, el mismo ensamblador se encarga de introducir el programa en el ordenador. No obstante, si no



La perfección alcanzada en los juegos comerciales sólo es posible con un dominio absoluto del código máquina.

se dispone de ensamblador, también es posible hacer la traducción "a mano". En este caso, será necesario escribir un pequeño programa en Basic que se encargue de introducir el código máquina que, normalmente, se encontrará en sentencias DATA.

Hay dos formas de almacenar un programa en código máquina, una es bajar la RAMTOP (con CLEAR) y almacenar el programa por encima de ésta, con lo que quedará a salvo de borrados accidentales. Este es el sistema más usado, pero en determinados casos, puede ser interesante meter un programa corto en la memoria intermedia de impresora, si bien hay que tener en cuenta que, en este caso, será borrado por cualquier comando que utilice la impresora, o bien por el comando NEW.

Para ilustrar la velocidad y posibilidades del código máquina, hemos desarrollado una pequeña rutina que permite reenumerar las líneas del programa Basic, empezando por la línea 10 y siguiendo de 10 en 10.

En la FIGURA 5 se muestra el listado del programa en lenguaje Assembler, a la izquierda está la traducción a código máquina.

En este caso hemos preferido almacenar el programa en la memoria intermedia de impresora, con el fin de que las direcciones sean las mismas para 16 y 48 K. No obstante, el programa es "reubicable", lo que quiere decir que puede correr en cualquier posición de memoria.

El siguiente programa en Basic se encarga de introducir el código máquina en memoria:

```

10 LET dir=23296
20 FOR n=1 TO 35
30 READ a: POKE dir,a
40 LET dir=dir+1
50 NEXT n
60 DATA 42,83,92,17,10,8,229,2
37,75,75,92,237,65,48,18,225,114
35,115,35,75,35,75,9,35,235,1,1
6,8,9,235,24,229,207,255

```

Cuando lo haya ejecutado, puede salvarlo en cinta tecleando:

```
SAVE "renum." CODE 23296,35
```

Cuando tenga un programa Basic en el que las líneas no estén numeradas de 10 en 10, cargue este reenumerador con:

```
LOAD ""CODE
```

Y cuando lo tenga, teclee:

```
RANDOMIZE USR 23296
```

Deberá obtener el mensaje:

```
0 OK, 0:1
```

que le indicará que todo ha ido correctamente; si ahora hace un listado, verá que las líneas están numeradas de 10 en 10 y empezando por la 10. Tenga en cuenta, no obstante, que los GO TO y GO SUB no habrán sido reenumerados, por lo que deberá hacerlo manualmente.

Si desea que la primera línea sea la 100 y que se numeren de 50 en 50, teclee:

```
POKE 23300,100
POKE 23323,50
```

En general, la dirección 23300 almacena el número de la primera línea (entre 0 y 255) y la dirección 23323 el incremento (también entre 0 y 255).

Si desea que un programa en código máquina se auto-ejecute, deberá utilizar un pequeño cargador en Basic de la forma:

```
10 LOAD ""CODE: RANDOMIZE
USR (dirección)
```

que salvará en cinta con SAVE... LINE 10 antes del programa en código máquina. ■

LOS PERIFERICOS

Un ordenador se compone, básicamente, de una CPU (Unidad Central de Proceso) y de una cierta cantidad de memoria. En el Spectrum, la CPU es el microprocesador Z-80. Este núcleo debe comunicarse con el exterior, para lo cual se sirve de los dispositivos periféricos.

Un periférico es todo dispositivo que se une al ordenador, excepto la CPU y la memoria. Son ejemplos de periféricos, el teclado, la pantalla, el cassette, el joystick, la impresora, el Microdrive, etc.

La CPU se comunica con la memoria a través de los buses de direcciones y de datos, indicando en el bus de control, que quiere acceder a la memoria. Igualmente, para comunicarse con los periféricos utiliza los buses de direcciones y datos, pero esta vez, el bus de control indica que se está accediendo a un periférico. La pantalla es una excepción, ya que la comunicación se hace mediante la ULA que funciona como una segunda "pseudo-CPU" con mayor prioridad.

De la misma forma que cada posición de memoria tiene una dirección, los periféricos tienen también uno o varios números que los definen. Estos números se denominan "ports" (en inglés, "puertos") y cumplen la misma función que las direcciones en la me-



Entre los sistemas de almacenamiento de datos alternativos al cassette se encuentra el wafadrive.



El joystick es especialmente adecuado para los juegos.

moria. Aunque el Z-80 sólo puede direccionar 256 ports de entrada/salida, el Spectrum se las arregla de forma ingeniosa para trabajar con números de port superiores a 255.

Las instrucciones que envían y reciben datos a y desde los ports tienen una sintaxis muy similar a las de la memoria (POKE y PEEK). Vamos a verlas a continuación.

OUT

Acceso al teclado

PEEK MODO E



OUT

+
SIMBOL
SHIFT

Definición

El comando OUT escribe un dato en un port de salida, el número de port se indica mediante la dirección.

Su estructura general es:

| SENTENCIA | ARGUMENTO |
|-----------|-----------------|
| OUT | Dirección, Dato |

Ejemplos:

- OUT 254,16
- OUT d,27
- OUT 25+d,a
- OUT 254,a*8

La dirección ha de estar comprendida entre 0 y 65535, y el dato, ha de estar entre -255 y 255 (un número negativo equivale a 256 menos ese número).

Si alguno de estos números estuviera fuera de este margen, se produciría el error:

B Integer out of range

IN

Acceso al teclado

CODE

MODO E



+

SIMBOL
SHIFT

IN

Definición

La función IN tiene como argumento la dirección de un port, y devuelve como resultado el dato que se encuentra en ese momento en el port.

Su estructura general es:

| SENTENCIA | ARGUMENTO |
|-----------|-----------|
| IN | Dirección |

Ejemplos:

- PRINT IN 254
- LET a=IN 32766
- PRINT 27+IN 223
- LET a=IN b

La dirección puede ser cualquier número entero comprendido entre 0 y 65535. Si estuviera fuera de este margen, se produciría el error:

B Integer out of range

En el Spectrum no se puede utilizar arbitrariamente cualquier port, cada uno tiene su función concreta y existen

números de port que no tienen ningún sentido. Para comprender el funcionamiento de los ports es imprescindible atender a la configuración binaria de los buses de direcciones y datos cuando se llama a cada port.

El bus de direcciones está compuesto por 16 bits, y cada uno maneja un determinado periférico. Los bits se numeran del 0 al 15 empezando por la derecha, y precedidos de una "A" para indicar que se trata del bus de direcciones. Así el bit de más a la derecha se denomina "A0", el siguiente "A1", y así sucesivamente hasta el de más a la izquierda que se denomina "A15".

La configuración binaria que se produce en el bus de direcciones, traducida a decimal, constituye la dirección del port, por ejemplo, la configuración binaria 00000000 01111110 corresponde al port 254 (uno de los más usados en el Spectrum). El bus de datos está compuesto por 8 bits, la configuración binaria de estos bits, traducida a decimal, constituye el dato que se almacena en el port o que se lee del mismo.

Los ocho bits de la derecha del bus de direcciones (A0 a A7), indican a qué periférico se quiere acceder. Esto se indica poniendo este bit a "0" mientras que los demás permanecen a "1", sólo uno de estos bits debe ser "0" a la vez, ya que de lo contrario, se podría crear confusión en la ULA al intentar acceder a varios periféricos simultáneamente.

Los ocho bits de la izquierda (A8 a A15) deben ser normalmente cero, sólo se utilizan cuando se desea acceder al teclado, en este caso,

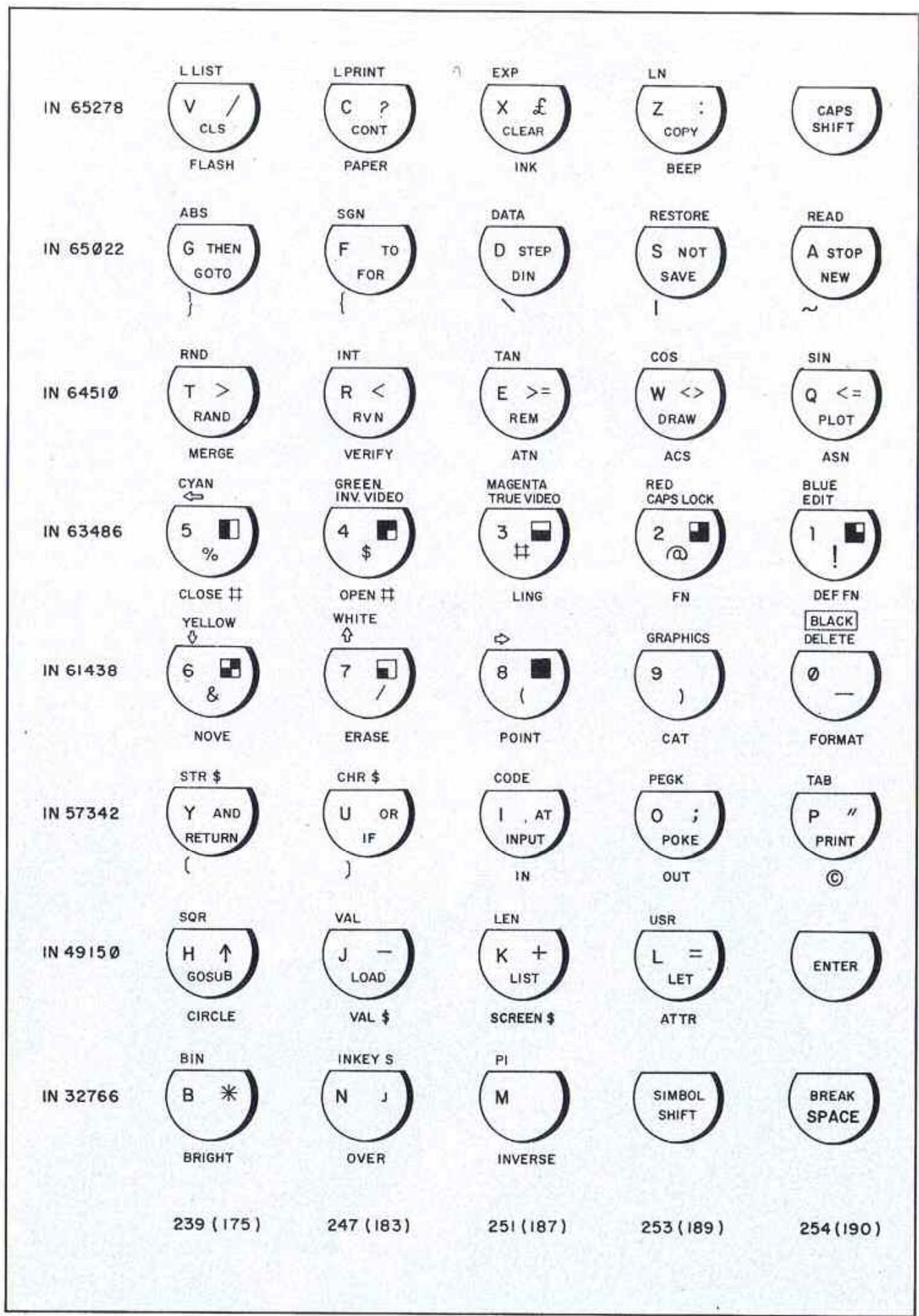


Figura 1. Los ports del teclado en el Spectrum.

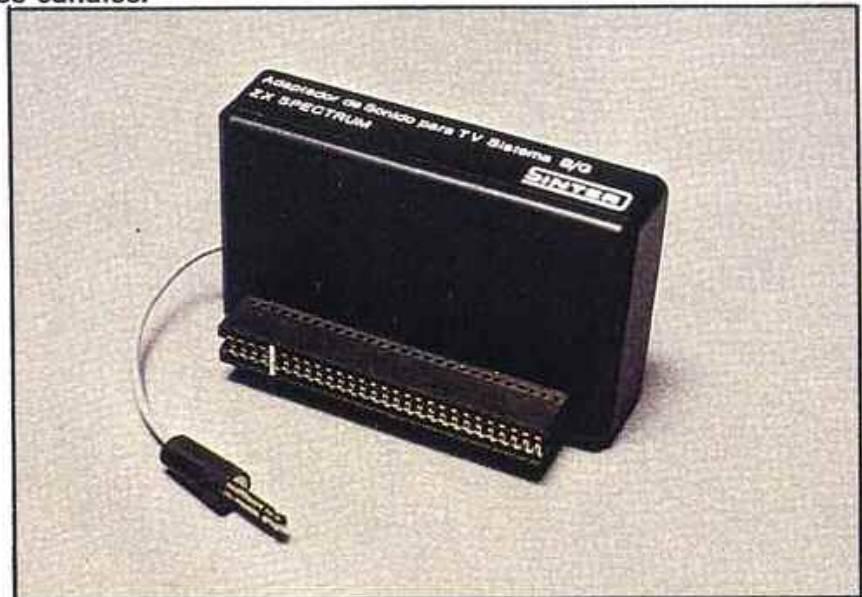


Las posibilidades sonoras del Spectrum quedan notablemente mejoradas con el uso de este sintetizador musical de tres canales.

cada uno indica una semifila (las cinco teclas derechas o izquierdas de una fila horizontal). El bit correspondiente a la fila deberá ser "0" mientras que los demás permanecerán a "1".

En la FIGURA 1 podemos ver las ocho semi-filas del teclado, a la izquierda está el número de port que se utiliza para leer cada semifila. Este dato varía según se trate de un modelo "ISSUE 2" o "ISSUE 3B". Los datos representados entre paréntesis corresponden al "ISSUE 3B" (Spectrum Plus). Si no hay ninguna tecla pulsada, el dato obtenido sería 255 en el "ISSUE 2" y 191 en el "ISSUE 3B".

En la TABLA 1 se ve la configuración binaria del bus de direcciones para cada uno de estos ports. Finalmente, el



La señal de ancho del Spectrum puede mezclarse con la de video para ser reproducida simultáneamente en T.V.

PROGRAMA 1 es un bucle que permite leer las ocho semi-filas devolviendo los datos de cada una en el "array" a (8) y presentándolos ordenados en la pantalla.

| | | |
|-------|------------|----------|
| 65278 | : 11111110 | 11111110 |
| 65022 | : 11111101 | 11111110 |
| 64510 | : 11111011 | 11111110 |
| 63486 | : 11110111 | 11111110 |
| 61438 | : 11101111 | 11111110 |
| 57342 | : 11011111 | 11111110 |
| 49150 | : 10111111 | 11111110 |
| 32766 | : 01111111 | 11111110 |

```

10 DIM a(8): PRINT AT 0,0:
20 FOR n=0 TO 7
30 LET a(n+1)=IN (254+256*(255
-27n))
40 PRINT "a("n+1;")=";a(n+1)
50 NEXT n: GO TO 10

```

Ejecute el PROGRAMA 1 y pulse varias teclas, le servirá para ver qué dato entrega cada una en su port correspondiente.

El port 254 es, sin duda, el más usado en la versión básica del Spectrum, la función IN 254 nos sirve para leer la entrada EAR, la señal está presente en el bit D6 del bus de datos.

Si lo utilizamos como salida, podremos controlar el color del borde con los bits D0, D1 y D2 del bus de datos, ejecute el siguiente programa:

```

10 INPUT "Color del borde ";a
20 OUT 254,a: GO TO 10

```

Tenga en cuenta que el color del borde es temporal, por lo que desaparecerá al pulsar cualquier tecla. Con el siguiente programa podrá conseguir un efecto curioso:

```

10 FOR n=0 TO 7: OUT 254,n
20 NEXT n: GO TO 10

```

El port 254 sirve también para hacer sonar el altavoz interno, la señal deberá estar presente en el bit D4. Por últi-

mo, el bit D3 excita la salida MIC.

Otro port muy importante del Spectrum es el 223 ya que nos sirve para leer un joystick tipo Kempston. Si tiene un interface joystick de este tipo, ejecute el PROGRAMA 2. Se trata de un "Tele-Sketch" con el que podrá dibujar por la pantalla, también puede borrar si mantiene apretado el pulsador de "disparo".

En la versión básica del Spectrum, los bits A5, A6 y A7 del bus de direcciones no se usan, por lo que podrá utilizarlos si decide construir su propio interface. Las direcciones adecuadas serán aquellas que hagan que A0-A4 sean todo "unos", estas direcciones son: 31, 63, 95, 127, 159, 191, 223 y 255.

PROGRAMA 2

```

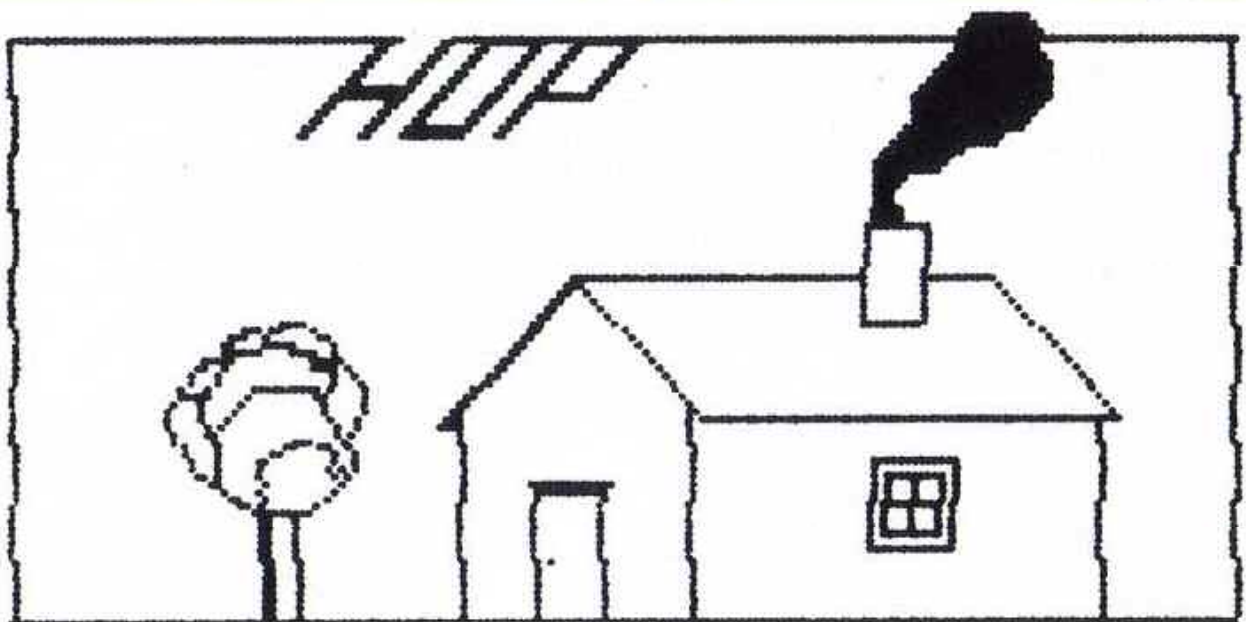
10 REM PROGRAMA 2
20 LET x=128: LET Y=88: LET i=
1
100 PLOT INVERSE i,x,y
110 LET a=IN 223: GO TO 150-30*(
a<>0)
120 LET i=(a>=16): LET a=a-16*(
a>16)

```

```

130 LET x=x+((a=1 OR a=9 OR a=5
) AND x<255)-((a=2 OR a=10 OR a=
6) AND x>0)
140 LET y=y+((a=8 OR a=9 OR a=1
0) AND y<175)-((a=4 OR a=5 OR a=
6) AND y>0)
150 PLOT x,y
160 GO TO 100

```



Joystick Kempston y un poco de paciencia.

VARIABLES DEL SISTEMA

De la misma manera que un programa Basic utiliza una serie de variables, el Sistema Operativo (que de hecho es un programa escrito en código máquina) utiliza las suyas; son lo que se denomina "Variables del Sistema".

Las Variables del Sistema están todas juntas, y ocupan direcciones de memoria fijas. Tienen nombres, pero el ordenador no los reconoce, su única finalidad es servir a efectos nemotécnicos, para recordarnos su función. El verdadero nombre por el que se hace referencia a una variable en concreto es la dirección de la posición de memoria que ocupa. La TABLA 1 es

una lista de todas las variables del sistema ordenadas alfabéticamente, con su dirección en decimal y hexadecimal y el número de bytes que ocupan.

Cuando una variable ocupa más de un byte, el primero contiene el octeto menos significativo y el último, el más significativo, por ejemplo, si el contenido de una variable de dos bytes de longitud fuera "3B4C" (en hexadecimal), el primer byte contendría "4C" y el segundo "3B". Justo al revés de lo que parecería normal, pero éste es el formato que necesita el microprocesador para poder leer los números correctamente.

La mayor parte de las variables ocupan dos bytes. Si desea leer el contenido de una variable cuya dirección es "d", utilice:

```
PRINT PEEK d+256*PEEK (d+1)
```

Y si desea almacenar el número "n" en una variable cuya dirección es "d", utilice:

```
POKE d,n-256*INT (n/256);
POKE d+1, INT (n/256)
```

El PROGRAMA 1 sirve para imprimir el contenido de cualquier variable del sistema, para ello pregunta primero el nombre de la variable, que deberá teclearse tal como aparece en la TABLA 1.

PROGRAMA 1

```
10 REM *****
*          *
*          LECTURA DE          *
*          *
*          VARIABLES DEL        *
*          *
*          SISTEMA              *
*          *
*          *****
20 DEF FN a(d)=PEEK d+256*PEEK
(d+1)
30 DEF FN b(d)=FN a(d)+65536*P
EEK (d+2)
100 REM CARGA DATOS
110 DIM a$(68,6): DIM b$(68,6):
DIM c$(1,6): RESTORE 8000: CLS
120 FOR n=1 TO 68: READ a$(n):
NEXT n
130 FOR n=1 TO 68: READ b$(n):
NEXT n
200 REM PIDE VARIABLE
210 POKE 23658,8
220 INPUT "Que variable desea e
xplorar? ";c$(1)
230 PRINT AT 19,0;"
240 POKE 23658,0
250 IF c$(1)="NINGUN" THEN CLS
: GO TO 9999
300 REM BUSCA VARIABLE
310 FOR n=1 TO 68: IF a$(n)=c$(
1) THEN GO TO 340
```

```
320 NEXT n
330 LET t=0: GO TO 400
340 LET t=VAL b$(n,1): LET d=VA
L b$(n,2 TO )
400 REM DISTRIBUYE A RUTINAS
410 IF NOT t THEN GO TO 450
420 GO TO t*1000
450 PRINT AT 19,0; FLASH 1;"
ESA VARIABLE NO EXISTE
GO TO 200
500 REM IMPRIME TIPOS 1, 2 y 3
510 GO SUB 7000: PRINT "En este
momento vale:";v: GO TO 200
1000 REM VARIABLE TIPO 1
1010 LET v=PEEK d: GO TO 500
2000 REM VARIABLE TIPO 2
2010 LET v=FN a(d): GO TO 500
3000 REM VARIABLE TIPO 3
3010 LET v=FN b(d): GO TO 500
4000 REM VARIABLE TIPO 4
4010 LET v=PEEK d: LET v1=PEEK (
d+1)
4500 REM IMPRIME TIPO 4
4510 GO SUB 7000: PRINT "tiene d
os valores, que son:";"v1";v
"v2";v1: GO TO 200
5000 REM VARIABLE TIPO 5
5500 REM IMPRIME TIPO 5
5510 GO SUB 7000: PRINT "Se usa
para leer el teclado y tiene d
inco valores, que son:"
5520 FOR i=0 TO 4: PRINT "KSTATE
-";i:" ";PEEK (d+i): NEXT i:
```

```

GO TO 200
6000 REM VARIABLE TIPO 6
6500 REM IMPRIME TIPO 6
6510 GO SUB 7000: PRINT "datos de
6 de los canales unidos a cada un
de las corrientes:"
6520 FOR n=0 TO 37 STEP 2: PRINT
FN a(d+n): NEXT n: GO TO 200
7000 REM CABECERA DE IMPRESION
7010 CLS: PRINT "VARIABLE:", c$(
1), "DIRECCION:", d: RETURN
8000 REM LISTA DE VARIABLES
8010 DATA "ATTR P", "ATTR T", "BOR
DCR", "BREG", "CH ADD", "CHANS", "CH
ARS", "COORDS", "CURCHL", "DATADD"
8020 DATA "DEFADD", "DEST", "DF CC",
"DF SZ", "DFCCL", "E LINE", "E PP
C", "ECHO E", "ERR NR", "ERR SP"
8030 DATA "FLAGS", "FLAG$2", "FLAG
X", "FRAMES", "K CUR", "K DATA", "KS
TATE", "LAST K", "LISTSP", "MASK P"
8040 DATA "MASK T", "MEM", "MEMBOT
", "MODE", "NEUPPC", "NMI", "NSPPC",
"NXTLIN", "OLDPPC", "OSPPC"
8050 DATA "P FLAG", "P POSN", "P R
AMT", "PIP", "PPC", "PR CC", "PROG",
"RAMTOP", "RASP", "REPDEL"
8060 DATA "REPPER", "S POSN", "S T
OP", "SCR CT", "SEED", "SPOSNL", "ST
KBOT", "STKEND", "STLEN", "STRMS"
8070 DATA "SUBPPC", "T ADDR", "TUD

```

```

ATA", "TVFLAG", "UDG", "VARS", "WORK
SP", "X PTR"
8500 REM DIRECCIONES Y TIPOS
8510 DATA "123693", "123695", "123
624", "123655", "223645", "223631",
"223606", "423677", "223633", "2236
39"
8520 DATA "223563", "223629", "223
684", "123659", "223686", "223641",
"223625", "223682", "123610", "2236
13"
8530 DATA "123611", "123658", "123
665", "323672", "223643", "123565",
"523552", "123560", "223615", "1236
94"
8540 DATA "123696", "223656", "223
698", "123617", "223618", "223726",
"123620", "223637", "223662", "1236
64"
8550 DATA "123697", "123679", "223
732", "123609", "223621", "123680",
"223635", "223730", "123608", "1236
61"
8560 DATA "123562", "423688", "223
660", "123692", "223670", "423690",
"223651", "223653", "223666", "6235
68"
8570 DATA "123623", "223668", "223
566", "123612", "223675", "223627",
"223649", "223647"

```

| Bytes | dirección | | Contenido | |
|----------|-----------|-------|-----------|--|
| | Nombre | dec. | | hexa. |
| ATTR P | 1 | 23693 | 5C8D | Atributos permanentes en curso. |
| ATTR T | 1 | 23695 | 5C8F | Atributos temporales en curso. |
| BORDCR | 1 | 23624 | 5C48 | Atributos de la parte inferior de la pantalla. |
| BREG | 1 | 23655 | 5C67 | Registro "B" del calculador. |
| CH ADD . | 2 | 23645 | 5C5D | Dirección del siguiente carácter que ha de ser interpretado por el Intérprete de Basic. |
| CHANS | 2 | 23631 | 5C4F | Dirección del área de información para canales. |
| CHARS | 2 | 23606 | 5C36 | Dirección del juego de caracteres, menos 256. |
| COORDS | 2 | 23677 | 5C7D | El primer byte indica la coordenada "x" del último punto PLOTeadado, y el segundo byte, la coordenada "y". |
| CURCHL | 2 | 23633 | 5C51 | Dirección del canal en curso. |
| DATADD | 2 | 23639 | 5C57 | Puntero de DATAs. |
| DEFADD | 2 | 23563 | 5C0B | Dirección del argumento de una función definida por el usuario, si se está valorando alguna, en otro caso, vale 0. |
| DEST | 2 | 23629 | 5C4D | Dirección de la variable en una asignación. |
| DF CC | 2 | 23684 | 5C84 | Dirección de la posición de PRINT en el archivo de pantalla. |
| DF SZ | 1 | 23659 | 5C6B | Número de líneas, incluida una en blanco, de la parte inferior de la pantalla. |
| DFCCL | 2 | 23686 | 5C86 | Como "DF CC", pero para la parte inferior de la pantalla. |
| E LINE | 2 | 23641 | 5C59 | Dirección del comando que está siendo tecleado. |
| E PPC | 2 | 23625 | 5C49 | Número de la línea a la que apunta el cursor del programa ">". |
| ECHO E | 2 | 23682 | 5C82 | Número de las 33 columnas, y de las 24 líneas del final del buffer de entrada. |
| ERR NR | 1 | 23610 | 5C3A | Código del informe, menos 1. |
| ERR SP | 2 | 23613 | 5C3D | Dirección del elemento de la pila de máquina que se usa como retorno en caso de error. |
| FLAGS | 1 | 23611 | 5C3B | Indicadores de control del Sistema. |
| FLAG\$ 2 | 1 | 23658 | 5C6A | Cursor "L" o "C". "L" = 0, "C" = 8. |
| FLAGX | 1 | 23665 | 5C71 | Indicadores de control del Sistema. |
| FRAMES | 3 | 23672 | 5C78 | Reloj en tiempo real, se incrementa cada 50 milisegundos. |
| K CUR | 2 | 23643 | 5C58 | Dirección del cursor. |

| Bytes | | dirección | | Contenido |
|--------|----|-----------|-------|---|
| Nombre | | dec. | hexa. | |
| K DATA | 1 | 23565 | 5C0D | Segundo byte de los controles de color introducidos por el teclado. |
| KSTATE | 8 | 23552 | 5C00 | Ocho variables intermedias, usadas para leer el teclado. |
| LAST K | 1 | 23560 | 5C08 | Código de la última tecla pulsada. |
| LISTSP | 2 | 23615 | 5C3F | Dirección de retorno tras un listado automático. |
| MASK P | 1 | 23694 | 5C8E | Máscara para colores transparentes. |
| MASK T | 1 | 23696 | 5C90 | Como "MASK P", pero temporal. |
| MEM | 2 | 23656 | 5C68 | Dirección del área usada como memoria del calculador. |
| MEMBOT | 2 | 23698 | 5C92 | Base de la memoria del calculador. |
| MODE | 1 | 23617 | 5C41 | Modo de cursor "K"=0, "E" =1, "G"=2 |
| NEWPPC | 2 | 23618 | 5C42 | Número de línea a la que hay que saltar. |
| NMI | 2 | 23728 | 5CB0 | Vector de interrupción no enmascarable (anulada para facilitar la protección del software comercial). |
| NSPPC | 1 | 23620 | 5C44 | Número de sentencia dentro de una línea, a la que hay que saltar. |
| NXTLIN | 2 | 23637 | 5C55 | Dirección de la siguiente línea de programa. |
| OLDPPC | 2 | 23662 | 5C6E | Línea a la que salta CONTINUE. |
| OSPPC | 1 | 23664 | 5C70 | Número de sentencia, dentro de la línea a la que salta CONTINUE. |
| P FLAG | 1 | 23697 | 5C91 | Indicadores para la impresión. |
| P POSN | 1 | 23679 | 5C7F | Número de las 33 columnas de la posición de la impresora. |
| P RAMT | 2 | 23732 | 5CB4 | Dirección del último byte de RAM física. |
| PIP | 1 | 23609 | 5C39 | Duración del tono emitido al pulsar una tecla. |
| PPC | 2 | 23621 | 5C45 | Contador del programa en Basic (número de línea). |
| PR CC | 1 | 23680 | 5C80 | Byte menos significativo de la dirección de la siguiente posición de LPRINT. |
| PROG | 2 | 23635 | 5C53 | Dirección de inicio del programa Basic. |
| RAMTOP | 2 | 23730 | 5CB2 | Dirección del último byte del área de memoria ocupada por el sistema Basic. |
| RASP | 1 | 23608 | 5C38 | Duración de la señal emitida al llenarse la memoria. |
| REPDEL | 1 | 23561 | 5C09 | Tiempo (en 1/50 de segundo) que ha de estar pulsada una tecla para que comience a repetirse. |
| REPPER | 1 | 23562 | 5C0A | Tiempo (en 1/50 de segundo) entre sucesivas repeticiones de una tecla. |
| S POSN | 2 | 23688 | 5C88 | El primer byte contiene el número de 33 columnas de la posición de PRINT, el segundo contiene el número de 24 líneas. |
| S TOP | 2 | 23660 | 5C6C | Número de la línea superior en un listado automático. |
| SCR CT | 1 | 23692 | 5C8C | Contador de (Scroll), es siempre 1 más que el número de líneas que se han de subir antes de preguntar (scroll?). |
| SEED | 2 | 23670 | 5C76 | Origen de las operaciones para generar un número pseudo-aleatorio. |
| SOSNL | 2 | 23690 | 5CBA | Como (S POSN), pero para la parte inferior de la pantalla. |
| STKBOT | 2 | 23651 | 5C63 | Dirección del fondo de la pila del calculador. |
| STKEND | 2 | 23653 | 5C65 | Dirección de la parte superior de la pila del calculador, e inicio del área de reserva. |
| STRLEN | 2 | 23666 | 5C72 | Longitud de la cadena de destino en una asignación. |
| STRMS | 38 | 23568 | 5C10 | Canales unidos a las corrientes abiertas. |
| SUBPPC | 1 | 23623 | 5C47 | Contador de programa Basic (número de sentencia dentro de la línea). |
| T ADDR | 2 | 23668 | 5C74 | Dirección del siguiente elemento de la tabla sintáctica. |
| TVDATA | 2 | 23566 | 5C0E | Bytes de color y controles AT y TAB que van al televisor. |
| TVFLAG | 1 | 23612 | 5C3C | Indicadores asociados con el televisor. |
| UDG | 2 | 23675 | 5C7B | Dirección del área de gráficos definidos por el usuario. |
| VARS | 2 | 23627 | 5C4B | Dirección del área de variables. |
| WORKSP | 2 | 23649 | 5C61 | Dirección del área de trabajo. |
| X PTR | 2 | 23647 | 5C5F | Dirección del carácter que sigue al signo (?). |

EL JUEGO DE CARACTERES

Cada uno de los signos, letras o números que puede imprimir el Spectrum se corresponde con un número comprendido entre 0 y 255, a este número se le denomina *código*.

Existen varios sistemas normalizados de codificación de caracteres, el más utilizado en informática se denomina ASCII, y es el que utiliza el Spectrum, con ligeras variaciones.

EL ASCII

La palabra ASCII está compuesta por las siglas de «American Standard Code for Information Interchange» (Código Normalizado Americano para Intercambio de Información).

El ASCII se elaboró en los Estados Unidos a finales de la década de los 60. En principio se pensó para su uso en teletipos, estos aparatos transmiten 7 bits de código más uno de «paridad», por lo que el ASCII utiliza códigos comprendidos entre 0 y 127. La decodificación en los teletipos se hacía de forma mecánica, por lo cual este proceso tuvo que ser simplificado al máximo. El ASCII utiliza los cinco bits de menor peso para designar un carácter determinado, y los dos siguientes, para indicar si se trata de

| | 001..... | 010..... | 011..... |
|----------|----------|----------|----------|
| ...00000 | espacio | @ | £ |
| ...00001 | ! | A | a |
| ...00010 | " | B | b |
| ...00011 | # | C | c |
| ...00100 | \$ | D | d |
| ...00101 | % | E | e |
| ...00110 | & | F | f |
| ...00111 | ' | G | g |
| ...01000 | (| H | h |
| ...01001 |) | I | i |
| ...01010 | * | J | j |
| ...01011 | + | K | k |
| ...01100 | , | L | l |
| ...01101 | - | M | m |
| ...01110 | . | N | n |
| ...01111 | / | O | o |
| ...10000 | 0 | P | p |
| ...10001 | 1 | Q | q |
| ...10010 | 2 | R | r |
| ...10011 | 3 | S | s |
| ...10100 | 4 | T | t |
| ...10101 | 5 | U | u |
| ...10110 | 6 | V | v |
| ...10111 | 7 | W | w |
| ...11000 | 8 | X | x |
| ...11001 | 9 | Y | y |
| ...11010 | : | Z | z |
| ...11011 | ; | [| { |
| ...11100 | < | \ | |

un código no imprimible (de control), un número o signo, una mayúscula o una minúscula.

Los 32 primeros códigos eran de control, y no producían la impresión de ningún carácter. Los 96 restantes constituyen propiamente, el juego de caracteres del ASCII, el código 127 se usa con frecuencia, para indicar el borrado del último carácter impreso. En la FIGURA 1 se pueden ver todos los caracteres del ASCII, ordenados según la configuración binaria de su código.

El juego de caracteres del Spectrum

El Spectrum utiliza una variante del ASCII, los 32 primeros códigos son también de control, si bien cumplen funciones bastante diferentes a las asignadas por el ASCII. Los 96 siguientes son idénticos a los caracteres ASCII, salvo el código 127 que se utiliza para el signo de «Copyright».

Por otro lado, el Spectrum no necesita comprobación de paridad, por lo que los ocho bits están disponibles. Ello permite duplicar el número de códigos utilizables.

Los 37 códigos siguientes al 127, se han utilizado para caracteres gráficos, de los cuales, los primeros 16 están definidos, y los 21 restantes son definibles por el usuario (los famosos UDG).

Los últimos 91 códigos han sido asignados a los «Tokens» que utiliza el Spectrum. La función CODE aplicada sobre una cadena, da como resultado el código del primer carácter que la compone. Su

| | | | |
|----------|---|---|---|
| ...11101 | = |] |) |
| ...11110 | > | ^ | ~ |
| ...11111 | ? | - | ⋮ |

Figura 1. Configuración binaria del ASCII.

| Código | | Funcion |
|--------|------|---|
| Dec. | Hexa | |
| 0 | 00 | No Utilizado |
| 1 | 01 | No utilizado |
| 2 | 02 | No utilizado |
| 3 | 03 | No utilizado |
| 4 | 04 | No utilizado |
| 5 | 05 | No utilizado |
| 6 | 06 | CAPS LOCK y efecto de "coma" en impresion. |
| 7 | 07 | EDIT |
| 8 | 08 | Cursor Izquierda |
| 9 | 09 | Cursor Derecha |
| 10 | 0A | Cursor Abajo |
| 11 | 0B | Cursor Arriba |
| 12 | 0C | DELETE (Borrado) |
| 13 | 0D | ENTER (Nueva linea) |
| 14 | 0E | CAPS SHIFT + SIMBOL SHIFT y "número" dentro de un programa. |
| 15 | 0F | No Utilizado |
| 16 | 10 | INK (Control de tinta). |
| 17 | 11 | PAPER (Control de papel). |
| 18 | 12 | FLASH (Control de parpadeo). |
| 19 | 13 | BRIGHT (Control de brillo) |
| 20 | 14 | INVERSE (Control de inversion de video). |
| 21 | 15 | OVER (Control de sobreimpresion). |
| 22 | 16 | AT (Control de posicionamiento). |
| 23 | 17 | TAB (Control de tabulacion) |
| 24 | 18 | No Utilizado |
| 25 | 19 | No Utilizado |
| 26 | 1A | No Utilizado |
| 27 | 1B | No Utilizado |
| 28 | 1C | No Utilizado |
| 29 | 1D | No Utilizado |
| 30 | 1E | No Utilizado |
| 31 | 1F | No Utilizado |

Figura 2. Códigos de Control.

inversa, CHR\$ aplicada sobre un número, da como resultado el carácter que tiene ese número como código.

La FIGURA 2, muestra la lista de códigos de control utilizados en el Spectrum. Las FIGURAS 3, 4 y 5 muestran las

tres partes de juego de caracteres. La FIGURA 6 corresponde a los caracteres gráficos y por último, la FIGURA 7 muestra la lista de «Tokens»

con sus códigos.

En todos los casos, los códigos han sido representados en decimal como en hexa. ■

| Código | | Caracter |
|--------|------|----------|
| Dec. | Hexa | |
| 32 | 20 | Espacio |
| 33 | 21 | ! |
| 34 | 22 | " |
| 35 | 23 | # |
| 36 | 24 | \$ |
| 37 | 25 | % |
| 38 | 26 | & |
| 39 | 27 | ' |
| 40 | 28 | (|
| 41 | 29 |) |
| 42 | 2A | * |
| 43 | 2B | + |
| 44 | 2C | , |
| 45 | 2D | - |
| 46 | 2E | . |
| 47 | 2F | / |
| 48 | 30 | 0 |
| 49 | 31 | 1 |
| 50 | 32 | 2 |
| 51 | 33 | 3 |
| 52 | 34 | 4 |
| 53 | 35 | 5 |
| 54 | 36 | 6 |
| 55 | 37 | 7 |
| 56 | 38 | 8 |
| 57 | 39 | 9 |
| 58 | 3A | : |
| 59 | 3B | ; |
| 60 | 3C | < |
| 61 | 3D | = |
| 62 | 3E | > |
| 63 | 3F | ? |

Figura 3. Signos y números.

| Codigo | | Caracter |
|--------|------|----------|
| Dec. | Hexa | |
| 64 | 40 | @ |
| 65 | 41 | A |
| 66 | 42 | B |
| 67 | 43 | C |
| 68 | 44 | D |
| 69 | 45 | E |
| 70 | 46 | F |
| 71 | 47 | G |
| 72 | 48 | H |
| 73 | 49 | I |
| 74 | 4A | J |
| 75 | 4B | K |
| 76 | 4C | L |
| 77 | 4D | M |
| 78 | 4E | N |
| 79 | 4F | O |
| 80 | 50 | P |
| 81 | 51 | Q |
| 82 | 52 | R |
| 83 | 53 | S |
| 84 | 54 | T |
| 85 | 55 | U |
| 86 | 56 | V |
| 87 | 57 | W |
| 88 | 58 | X |
| 89 | 59 | Y |
| 90 | 5A | Z |
| 91 | 5B | [|
| 92 | 5C | \ |
| 93 | 5D |] |
| 94 | 5E | ↑ |
| 95 | 5F | - |

Figura 4. Letras mayúsculas.

| Codigo | | Caracter |
|--------|------|----------|
| Dec. | Hexa | |
| 96 | 60 | £ |
| 97 | 61 | a |
| 98 | 62 | b |
| 99 | 63 | c |
| 100 | 64 | d |
| 101 | 65 | e |
| 102 | 66 | f |
| 103 | 67 | g |
| 104 | 68 | h |
| 105 | 69 | i |
| 106 | 6A | j |
| 107 | 6B | k |
| 108 | 6C | l |
| 109 | 6D | m |
| 110 | 6E | n |
| 111 | 6F | o |
| 112 | 70 | p |
| 113 | 71 | q |
| 114 | 72 | r |
| 115 | 73 | s |
| 116 | 74 | t |
| 117 | 75 | u |
| 118 | 76 | v |
| 119 | 77 | w |
| 120 | 78 | x |
| 121 | 79 | y |
| 122 | 7A | z |
| 123 | 7B | { |
| 124 | 7C | } |
| 125 | 7D | ~ |
| 126 | 7E | ~ |
| 127 | 7F | (c) |

Figura 5. Letras minúsculas.

| Codigo | | Caracter |
|--------|------|----------|
| Dec. | Hexa | |
| 128 | 80 | |
| 129 | 81 | . |
| 130 | 82 | : |
| 131 | 83 | = |
| 132 | 84 | . |
| 133 | 85 | |
| 134 | 86 | % |
| 135 | 87 | ^ |
| 136 | 88 | . |
| 137 | 89 | ^ |
| 138 | 8A | |
| 139 | 8B | ^ |
| 140 | 8C | = |
| 141 | 8D | ^ |
| 142 | 8E | ^ |
| 143 | 8F | ^ |
| 144 | 90 | UDG "A" |
| 145 | 91 | UDG "B" |
| 146 | 92 | UDG "C" |
| 147 | 93 | UDG "D" |
| 148 | 94 | UDG "E" |
| 149 | 95 | UDG "F" |
| 150 | 96 | UDG "G" |
| 151 | 97 | UDG "H" |
| 152 | 98 | UDG "I" |
| 153 | 99 | UDG "J" |
| 154 | 9A | UDG "K" |
| 155 | 9B | UDG "L" |
| 156 | 9C | UDG "M" |
| 157 | 9D | UDG "N" |
| 158 | 9E | UDG "O" |
| 159 | 9F | UDG "P" |
| 160 | A0 | UDG "Q" |
| 161 | A1 | UDG "R" |
| 162 | A2 | UDG "S" |
| 163 | A3 | UDG "T" |
| 164 | A4 | UDG "U" |

Figura 6. Caracteres gráficos.

| Código | | Token |
|--------|------|----------|
| Dec. | Hexa | |
| 165 | A5 | RND |
| 166 | A6 | INKEY\$ |
| 167 | A7 | PI |
| 168 | A8 | FN |
| 169 | A9 | POINT |
| 170 | AA | SCREEN\$ |
| 171 | AB | ATTR |
| 172 | AC | AT |
| 173 | AD | TAB |
| 174 | AE | VAL\$ |
| 175 | AF | CODE |
| 176 | B0 | VAL |
| 177 | B1 | LEN |
| 178 | B2 | SIN |
| 179 | B3 | COS |
| 180 | B4 | TAN |
| 181 | B5 | ASN |
| 182 | B6 | ACS |
| 183 | B7 | ATN |
| 184 | B8 | LN |
| 185 | B9 | EXP |
| 186 | BA | INT |
| 187 | BB | SQR |
| 188 | BC | SGN |
| 189 | BD | ABS |
| 190 | BE | PEEK |
| 191 | BF | IN |
| 192 | C0 | USR |
| 193 | C1 | STR\$ |
| 194 | C2 | CHR\$ |
| 195 | C3 | NOT |
| 196 | C4 | BIN |
| 197 | C5 | OR |
| 198 | C6 | AND |
| 199 | C7 | <= |
| 200 | C8 | >= |
| 201 | C9 | <> |
| 202 | CA | LINE |
| 203 | CB | THEN |
| 204 | CC | TO |
| 205 | CD | STEP |
| 206 | CE | DEF FN |
| 207 | CF | CAT |
| 208 | D0 | FORMAT |
| 209 | D1 | MOVE |
| 210 | D2 | ERASE |

Figura 7.
Códigos de «Tokens».

| Código | | Token |
|--------|------|-----------|
| Dec. | Hexa | |
| 211 | D3 | OPEN # |
| 212 | D4 | CLOSE # |
| 213 | D5 | MERGE |
| 214 | D6 | VERIFY |
| 215 | D7 | BEEP |
| 216 | D8 | CIRCLE |
| 217 | D9 | INK |
| 218 | DA | PAPER |
| 219 | DB | FLASH |
| 220 | DC | BRIGHT |
| 221 | DD | INVERSE |
| 222 | DE | OVER |
| 223 | DF | OUT |
| 224 | E0 | LPRINT |
| 225 | E1 | LLIST |
| 226 | E2 | STOP |
| 227 | E3 | READ |
| 228 | E4 | DATA |
| 229 | E5 | RESTORE |
| 230 | E6 | NEW |
| 231 | E7 | BORDER |
| 232 | E8 | CONTINUE |
| 233 | E9 | DIM |
| 234 | EA | REM |
| 235 | EB | FOR |
| 236 | EC | GO TO |
| 237 | ED | GO SUB |
| 238 | EE | INPUT |
| 239 | EF | LOAD |
| 240 | F0 | LIST |
| 241 | F1 | LET |
| 242 | F2 | PAUSE |
| 243 | F3 | NEXT |
| 244 | F4 | POKE |
| 245 | F5 | PRINT |
| 246 | F6 | PLOT |
| 247 | F7 | RUN |
| 248 | F8 | SAVE |
| 249 | F9 | RANDOMIZE |
| 250 | FA | IF |
| 251 | FB | CLS |
| 252 | FC | DRAW |
| 253 | FD | CLEAR |
| 254 | FE | RETURN |
| 255 | FF | COPY |

INDICE

INTRODUCCION

El lenguaje del Spectrum _____ 2

Parte I

Capítulo 1

TECLADO DEL SPECTRUM

Acceso al teclado _____ 3

Modo **K** _____ 3

Modo **L** _____ 6

Modo **C** _____ 6

Modo **E** _____ 6

Modo **G** _____ 7

Capítulo 2

CONFECCION DE PROGRAMAS

Edición de programas _____ 9

Corrección de errores _____ 11

Ejercicio _____ 14

Capítulo 3

EL SPECTRUM PLUS

Teclado del «ZX Spectrum +» _____ 14

Modos **L** **C** _____ 15

Modo **E** _____ 15

Modo **G** _____ 16

Edición de programas _____ 16

Capítulo 4

ALMACENAMIENTO DE PROGRAMAS

Introducción _____ 17

Verificación _____ 19

Recuperación de programas _____ 20

Protección de programas _____ 21

Conservación de cintas _____ 23

Oscilación _____ 24

Ajuste _____ 24

Capítulo 5

CONSTANTES Y VARIABLES

Introducción _____ 25

Constantes numéricas _____ 26

Notación entera _____ 26

Notación decimal _____ 27

Notación exponencial _____ 27

Notación binaria _____ 30

Decimal-Binario _____ 30

Binario-Decimal _____ 31

Ejercicio _____ 31

Constantes alfanuméricas _____ 31

Variables numéricas _____ 32

Variables alfanuméricas _____ 32

Capítulo 6

OPERADORES

| | |
|-------------------------|----|
| Introducción | 33 |
| Operadores aritméticos | 33 |
| Expresiones aritméticas | 33 |
| Cálculo de expresiones | 33 |
| Operadores de relación | 34 |
| Operadores lógicos | 34 |
| Función «AND» | 34 |
| Función «OR» | 35 |
| Función «NOT» | 35 |
| Ejercicio | 36 |

Capítulo 7

CODIGO ASCII

| | |
|------------------------|----|
| Introducción | 37 |
| Manejo de la tabla | 37 |
| Organización del ASCII | 38 |
| Transmisión del ASCII | 40 |

Capítulo 8

OPERACIONES CON CADENAS

| | |
|--------------------------|----|
| Concatenación de cadenas | 42 |
| Subcadenas | 43 |
| Fragmentación | 43 |
| Fragmentación específica | 43 |
| Asignación de subcadenas | 45 |
| Comparación de cadenas | 45 |
| Ordenación de cadenas | 47 |
| Prioridades | 48 |

Capítulo 9

ELABORACION DE PROGRAMAS

320 MICROBASIC

| | |
|------------------------|----|
| Introducción | 50 |
| Análisis | 50 |
| Síntesis | 51 |
| Representación gráfica | 51 |
| Programa | 52 |

Parte II

Capítulo 10

EL JUEGO DE SENTENCIAS

| | |
|------------------------------|----|
| Clasificación | 60 |
| Comandos de control | 60 |
| Comandos de programación | 60 |
| Comandos de entrada/salida | 60 |
| Manejo de cadenas | 61 |
| Funciones aritméticas | 61 |
| Funciones lógicas | 61 |
| Comandos de dibujo | 61 |
| Comandos de control de color | 62 |
| Sonido | 62 |
| Manejo de impresora | 62 |
| Interface 1 | 63 |
| Manejo Microdrive | 63 |
| Auxiliares | 64 |
| Programa «COLOREAR» | 64 |
| Programa «GRAFICAS» | 64 |
| Programa «GDU» | 66 |
| Programa «DIBUJANDO» | 66 |
| Programa «BIPBIP» | 68 |

Capítulo 11

COMANDOS BASICOS

| | |
|------------------------|----|
| REM | 69 |
| Funciones de Video | 70 |
| LET | 70 |
| Representación gráfica | 71 |
| PRINT | 71 |
| Desplazamientos | 72 |
| Formatos | 72 |

| | | | |
|-------------------------------|----|-----------------------|-----|
| TAB _____ | 73 | Introducción _____ | 108 |
| AT _____ | 73 | DO WHILE _____ | 108 |
| Canales de comunicación _____ | 75 | REPEAT UNTIL _____ | 108 |
| INPUT _____ | 77 | Diferencias _____ | 108 |
| INPUT TAB y AT _____ | 79 | FOR/NEXT _____ | 109 |
| INPUT LINE _____ | 80 | STEP _____ | 111 |
| Otra aplicación _____ | 80 | Bucles anidados _____ | 112 |
| Programas de repaso _____ | 81 | Errores _____ | 113 |
| Programa «GRANJA» _____ | 81 | Programas _____ | 113 |
| Programa «EDUCACION» _____ | 83 | | |
| Programa «INTERES» _____ | 84 | | |
| Programa «GRADOS» _____ | 86 | | |
| Programa «FICHA» _____ | 86 | | |

Capítulo 12

COMANDOS DE CONTROL

| | |
|----------------------------|----|
| RUN _____ | 89 |
| BREAK _____ | 90 |
| STOP _____ | 91 |
| Ruptura del «INPUT» _____ | 92 |
| CONTINUE _____ | 94 |
| Informes de pantalla _____ | 96 |
| NEW _____ | 97 |
| CLS _____ | 97 |
| LIST _____ | 98 |
| LIST y EDIT _____ | 98 |

Capítulo 13

SALTOS INCONDICIONALES Y CONDICIONALES

| | |
|-------------------------------------|-----|
| Introducción _____ | 99 |
| GO TO _____ | 99 |
| IF... THEN... _____ | 101 |
| Evaluación de las condiciones _____ | 103 |
| Programa _____ | 107 |

Capítulo 14

BUCLES

SUBROUTINAS

| | |
|---|-----|
| Introducción _____ | 122 |
| GO SUB _____ | 122 |
| RETURN _____ | 122 |
| Utilización de «GOSUB» y «RETURN» _____ | 122 |
| Tipos de subrutinas _____ | 124 |
| Subrutinas anidadas _____ | 125 |
| Error _____ | 126 |
| Programas _____ | 127 |

Capítulo 16

DATOS DE UN PROGRAMA

| | |
|--------------------------------------|-----|
| Introducción _____ | 134 |
| READ _____ | 134 |
| DATA _____ | 134 |
| Utilización de «READ» y «DATA» _____ | 134 |
| RESTORE _____ | 135 |
| Errores _____ | 137 |
| Programas _____ | 140 |

Capítulo 17

LECTURA DEL TECLADO Y TEMPORIZACIONES

| | |
|-----------------|-----|
| INKEY\$ _____ | 142 |
| PAUSE _____ | 142 |
| Programas _____ | 144 |

Capítulo 18

FUNCIONES

| | |
|--------------------------------|-----|
| Introducción | 148 |
| Funciones numéricas | 148 |
| ABS | 148 |
| INT | 148 |
| SGN | 149 |
| SQR | 149 |
| BIN | 150 |
| PI | 150 |
| El radián | 151 |
| SIN | 151 |
| COS | 152 |
| TAN | 152 |
| ASN | 152 |
| ACS | 152 |
| ATN | 152 |
| Aplicación de la trigonometría | 153 |
| Función exponencial | 154 |
| EXP | 154 |
| Función logarítmica | 155 |
| LN | 155 |
| Definición de funciones | 159 |
| DEF FN | 159 |
| FN | 159 |
| Errores | 160 |

Capítulo 19

FUNCION ALEATORIA

| | |
|----------------------|-----|
| Introducción | 161 |
| RND | 161 |
| Programa «BARQUITOS» | 164 |
| RANDOMIZE | 164 |
| Programa «TABLA» | 168 |

Capítulo 20

FUNCIONES DE CADENA

| | |
|--------------|-----|
| Introducción | 169 |
|--------------|-----|

| | |
|-------------------------------|-----|
| LEN | 170 |
| STR\$ | 171 |
| VAL | 171 |
| VAL\$ | 172 |
| Conversiones de código | 172 |
| CHR\$ | 172 |
| CODE | 174 |
| Funciones definidas de cadena | 175 |
| Errores | 175 |
| Programa | 177 |

Capítulo 21

MATRICES

| | |
|----------------------------|-----|
| Introducción | 179 |
| Dimensionado de matrices | 179 |
| DIM | 180 |
| Matrices numéricas | 180 |
| Asignación y visualización | 182 |
| Manejo de tablas | 185 |
| Matrices de cadena | 195 |
| Asignación | 196 |
| Fragmentación | 196 |
| Errores | 199 |
| Grabación de datos | 200 |
| Programa | 202 |

Capítulo 22

DEPURACION DE PROGRAMAS

| | |
|----------------------|-----|
| Introducción | 206 |
| Errores | 206 |
| Depuración | 207 |
| STOP y CONTINUE | 209 |
| Programa «Depurador» | 209 |
| Ejercicio | 210 |

Capítulo 23

COLOR

| | |
|--------------|-----|
| Introducción | 211 |
|--------------|-----|

| | |
|---|-----|
| Teoría del color _____ | 212 |
| Síntesis aditiva _____ | 213 |
| Zonas de pantalla _____ | 214 |
| BORDER _____ | 214 |
| PAPER _____ | 215 |
| INK _____ | 215 |
| Atributos permanentes y temporales _____ | 216 |
| Resolución del color _____ | 217 |
| Transparencia y contraste _____ | 217 |
| Simulación de colores _____ | 218 |
| Control de impresión _____ | 219 |
| BRIGHT _____ | 219 |
| INVERSE _____ | 220 |
| FLASH _____ | 221 |
| Atributos de pantalla _____ | 222 |
| ATTR _____ | 222 |
| Caracteres de control _____ | 222 |
| Acceso directo _____ | 223 |
| Errores _____ | 224 |

Capítulo 24

GRAFICOS

| | |
|-----------------------------------|-----|
| Introducción _____ | 226 |
| Tipos de gráficos _____ | 226 |
| Bloques de color _____ | 226 |
| Gráficos predefinidos _____ | 226 |
| Pantalla en alta resolución _____ | 228 |
| PLOT _____ | 228 |
| DRAW _____ | 229 |
| Arcos de circunferencia _____ | 232 |
| Programa especial _____ | 234 |
| CIRCLE _____ | 234 |
| Técnicas avanzadas _____ | 235 |
| OVER _____ | 239 |
| SCREEN\$ _____ | 242 |
| Almacenamiento de pantallas _____ | 242 |
| POINT _____ | 242 |
| Programa _____ | 243 |

Capítulo 25

GRAFICOS DEFINIDOS

| | |
|-----------------------------------|-----|
| Introducción _____ | 246 |
| ¿Cómo se almacenan? _____ | 246 |
| Definición de «GDU» _____ | 249 |
| Utilización de los «GDU» _____ | 250 |
| Programas de aplicación _____ | 254 |
| Grabación de los «GDU» _____ | 254 |
| Lectura de los «GDU» _____ | 254 |
| Programa generador de «GDU» _____ | 254 |
| Programa _____ | 255 |

Capítulo 26

SONIDO

| | |
|------------------------------|-----|
| Introducción _____ | 258 |
| BEEP _____ | 258 |
| Nociones musicales _____ | 261 |
| Tono _____ | 261 |
| Duración _____ | 263 |
| Compás _____ | 264 |
| Variables relacionadas _____ | 264 |
| Grabación de sonidos _____ | 264 |
| Periféricos _____ | 266 |
| Software Musical _____ | 266 |
| Efectos sonoros _____ | 266 |

Capítulo 27

SENTENCIAS DE GRABACION Y CARGA

| | |
|-----------------------------|-----|
| Introducción _____ | 267 |
| SAVE _____ | 267 |
| Programa LISTADOR _____ | 268 |
| Grabación de matrices _____ | 269 |
| Programa DIRECTORIO _____ | 270 |
| VERIFY _____ | 271 |
| LOAD _____ | 272 |
| Programa EDIT/DIR _____ | 273 |
| MERGE _____ | 273 |
| Comodidad de uso _____ | 274 |
| Búsqueda de programas _____ | 274 |
| Referencias _____ | 274 |
| Programas _____ | 274 |

Capítulo 28

GESTION DE IMPRESORA

| | |
|----------------------------|-----|
| Introducción | 276 |
| LPRINT | 276 |
| Programa TEST DE IMPRESORA | 277 |
| LLIST | 277 |
| COPY | 278 |
| Ejemplo de COPY | 278 |
| Otras impresoras | 278 |
| Tipos de impresoras | 279 |
| Elección de una impresora | 280 |
| Juego de caracteres | 281 |

Capítulo 29

INTERFACE 1

| | |
|---------------------------------------|-----|
| Introducción | 283 |
| Canales y corrientes | 283 |
| Asignación de canales y corrientes | 283 |
| OPEN # | 283 |
| Desactivación de canales y corrientes | 284 |
| CLOSE # | 284 |
| El Microdrive | 284 |
| FORMAT | 284 |
| Programa FICHERO | 285 |
| CAT | 286 |
| Grabación y carga | 287 |
| Borrado de programas | 287 |
| ERASE | 288 |
| Ejecución automática | 288 |
| Protección de ficheros | 288 |
| Ficheros de datos | 288 |
| Grabación de datos | 289 |
| Apertura de fichero | 289 |
| Cierre de ficheros | 289 |
| Lectura y ampliación de ficheros | 290 |
| Red de área local | 292 |
| Interface RS-232 | 293 |

Capítulo 30

LA MEMORIA

| | |
|---|-----|
| Introducción | 295 |
| POKE | 295 |
| PEEK | 296 |
| Tipos de memoria | 296 |
| Bit y Byte | 297 |
| Memoria ROM | 297 |
| La Memoria RAM | 298 |
| Almacenamiento de programas | 301 |
| Variables | 301 |
| Variable numerica cuyo nombre es una sola letra | 301 |
| Variable numérica cuyo nombre son varias letras | 301 |
| Variable de cadena de caracteres | 301 |
| Variable de control de bucle FOR-NEXT | 301 |
| Matriz de numeros | 302 |
| Matriz de caracteres | 302 |
| Borrado de variables | 302 |
| CLEAR | 302 |
| Cuando se llena la memoria | 303 |
| Programando en código máquina | 305 |
| USR | 305 |

Capítulo 31

LOS PERIFERICOS

| | |
|---------------------------|-----|
| Introducción | 307 |
| OUT | 308 |
| IN | 308 |
| Ports del teclado | 309 |
| Programa SCAN DEL TECLADO | 311 |
| Programa SKETCH | 311 |

Capítulo 32

VARIABLES DEL SISTEMA

| | |
|--------------------------------|-----|
| Introducción | 312 |
| Programa LECTOR DE VARIABLES | 312 |
| Tabla de variables del Sistema | 313 |

SENTENCIAS Y FUNCIONES

A

| | |
|------------|-----|
| ABS _____ | 148 |
| ACS _____ | 152 |
| AND _____ | 34 |
| ASN _____ | 152 |
| AT _____ | 73 |
| ATN _____ | 152 |
| ATTR _____ | 222 |

B

| | |
|--------------|-----|
| BEEP _____ | 258 |
| BIN _____ | 150 |
| BORDER _____ | 214 |
| BRIGHT _____ | 219 |

C

| | |
|----------------------------|-----|
| CAT _____ | 286 |
| CHR\$ _____ | 172 |
| CIRCLE _____ | 234 |
| CLEAR _____ | 302 |
| CLOSE # _____ | 284 |
| CLS _____ | 97 |
| CODE _____ | 174 |
| CONTINUE (CONT) _____ 94 y | 209 |
| COPY _____ | 278 |
| COS _____ | 152 |

D

| | |
|--------------|-----|
| DATA _____ | 134 |
| DEF FN _____ | 159 |
| DIM _____ | 180 |
| DRAW _____ | 229 |

E

| | |
|-------------|-----|
| ERASE _____ | 288 |
| EXP _____ | 154 |

F

| | |
|--------------|-----|
| FLASH _____ | 221 |
| FN _____ | 159 |
| FOR _____ | 109 |
| FORMAT _____ | 284 |

G

| | |
|-------------|-----|
| GOSUB _____ | 122 |
| GO TO _____ | 99 |

I

| | |
|---------------|-----|
| IF _____ | 101 |
| IN _____ | 308 |
| INK _____ | 215 |
| INKEY\$ _____ | 142 |
| INPUT _____ | 77 |
| INT _____ | 148 |
| INVERSE _____ | 220 |

L

| | |
|-------------|-----|
| LEN _____ | 170 |
| LET _____ | 70 |
| LINE _____ | 80 |
| LIST _____ | 98 |
| LLIST _____ | 277 |
| LN _____ | 155 |
| LOAD _____ | 272 |

LPRINT _____ 276

M

MERGE _____ 273

MOVE _____ 292

N

NEW _____ 97

NEST _____ 109

NOT _____ 35

O

OPEN # _____ 283

OR _____ 35

OUT _____ 308

OVER _____ 239

P

PAPER _____ 215

PAUSE _____ 142

PEEK _____ 296

PI _____ 150

PLOT _____ 228

POINT _____ 242

POKE _____ 295

PRINT _____ 71

R

RANDOMIZE (RAND) _____ 164

READ _____ 134

REM _____ 69

RESTORE _____ 135

RETURN _____ 122

RND _____ 161

RUN _____ 89

S

SAVE _____ 267

SCREEN\$ _____ 242

SGN _____ 149

SIN _____ 151

SQR _____ 149

STEP _____ 111

STOP _____ 91 y 209

STR\$ _____ 171

T

TAB _____ 73

TAN _____ 152

THEN _____ 101

TO _____ 110

U

USR _____ 305

V

VAL _____ 171

VAL\$ _____ 172

VERIFY _____ 271

PROGRAMAS «MICROBASIC»

| | | | |
|--------------------------|-----------------|----------------------------|----------------|
| EJER1 _____ | 10, 11, 15 y 16 | CARTAS _____ | 189, 190 y 191 |
| CODEBIN _____ | 25, 26 y 27 | 11 ERRORES _____ | 209 |
| ASCII _____ | 37 | SIN ERRORES _____ | 210 |
| ASCII/DECIMAL _____ | 39 | ATRIBUTOS _____ | 224 |
| CHR\$ _____ | 40 | DEPURACION _____ | 225 |
| CONCATENACION _____ | 42 | CARTA COLOR _____ | 225 |
| FRAGMENTACION _____ | 44 y 45 | COLORES 1 _____ | 225 |
| COMPARACION _____ | 47 | COLORES 2 _____ | 225 y 226 |
| ORDENA _____ | 48 y 49 | DIBUJO PEZ _____ | 227 y 228 |
| LISTIN _____ | 51, 52, 53 y 59 | MICROHOBBY _____ | 230 |
| COLOREAR _____ | 61 | ABSTRACTO _____ | 232 |
| GRAFICAS _____ | 62 | GRAFICO _____ | 233 |
| G.D.U. _____ | 64 | RECTAS _____ | 234 |
| DIBUJANDO _____ | 65 | MAPA _____ | 237, 238 y 239 |
| BIPBIP _____ | 66 | OVER _____ | 239 |
| GRANJA _____ | 74 y 75 | LABERINTO _____ | 240 |
| EDUCACION 2 GRADO _____ | 75 y 76 | ALFABETO ESPAÑOL _____ | 246 |
| INTERES SIMPLE _____ | 77 | NOTAS GRAFICAS _____ | 247 |
| GRADOS _____ | 78 y 79 | LECTURA «GDU» _____ | 250 |
| FICHA _____ | 79 y 80 | PALITROQUE _____ | 251, 252 y 253 |
| CALCULADORA _____ | 92 y 93 | NAVIDAD _____ | 259 |
| iiiNEW!!! _____ | 93 y 94 | DIATONICA _____ | 259 |
| AGENDA _____ | 102 | CROMATICA _____ | 260 |
| AREAS Y PERIMETROS _____ | 105 y 106 | DOS CRUCES _____ | 260 y 261 |
| ESTADISTICA _____ | 114 y 115 | EFECTO 1 _____ | 263 |
| HISTOGRAMAS _____ | 116 | EFECTO 2 _____ | 263 |
| ADIVINO _____ | 126 | EFECTO 3 _____ | 263 |
| LONGITUD _____ | 130 | EFECTO 4 _____ | 263 |
| GEOGRAFIA 1 _____ | 138 | NAVIDAD (TREMOLLO) _____ | 264 |
| GEOGRAFIA 2 _____ | 139 y 140 | LISTADOR _____ | 268 |
| MAQUINA _____ | 145 | DIRECTORIO _____ | 270 y 271 |
| MOVIMIENTO _____ | 157 | EDIT/DIR _____ | 273 |
| iiiAGUA!!! _____ | 165 | IMPRESORA (TEST) _____ | 277 |
| LA TABLA _____ | 166 | FICHERO _____ | 285 y 286 |
| BUSQUEDA _____ | 169 | RENUMERADOR _____ | 304 y 306 |
| INSERTAR _____ | 169 | SCAN DE TECLADO _____ | 311 |
| ANULAR _____ | 171 | SKETCK _____ | 311 |
| INPUT _____ | 174 | LECTURA DE VARIABLES _____ | 312 |
| MANEJO DE TABLAS _____ | 179 y 180 | | |



MICROBASIC

Aunque el Basic fue diseñado como un lenguaje de alto nivel universal para todos los microordenadores, la salida al mercado de los distintos modelos fue diversificando su estructura y prestaciones hasta tal punto que actualmente puede decirse que no hay dos máquinas que permitan ser programadas utilizando un Basic idéntico para ambas.

Sinclair no constituye una excepción y este manual está especializado en el Basic utilizado en sus microordenadores Spectrum y Spectrum+.

Profusamente ilustrado con esquemas, fotos y figuras a todo color, todas las explicaciones están acompañadas de abundantes ejemplos prácticos.

Con toda probabilidad, se trata del libro más completo que se ha publicado sobre el Basic Sinclair, aclarando importantes lagunas existentes en este terreno.

Respecto de la programación en Basic, constituye una completa guía de utilización de cada uno de los comandos disponibles y, quizás el aspecto más destacable, su grado de dificultad es progresivo, haciéndolo especialmente indicado para todas aquellas personas que sin tener conocimientos específicos en esta materia desean aventurarse en el apasionante mundo de la programación.