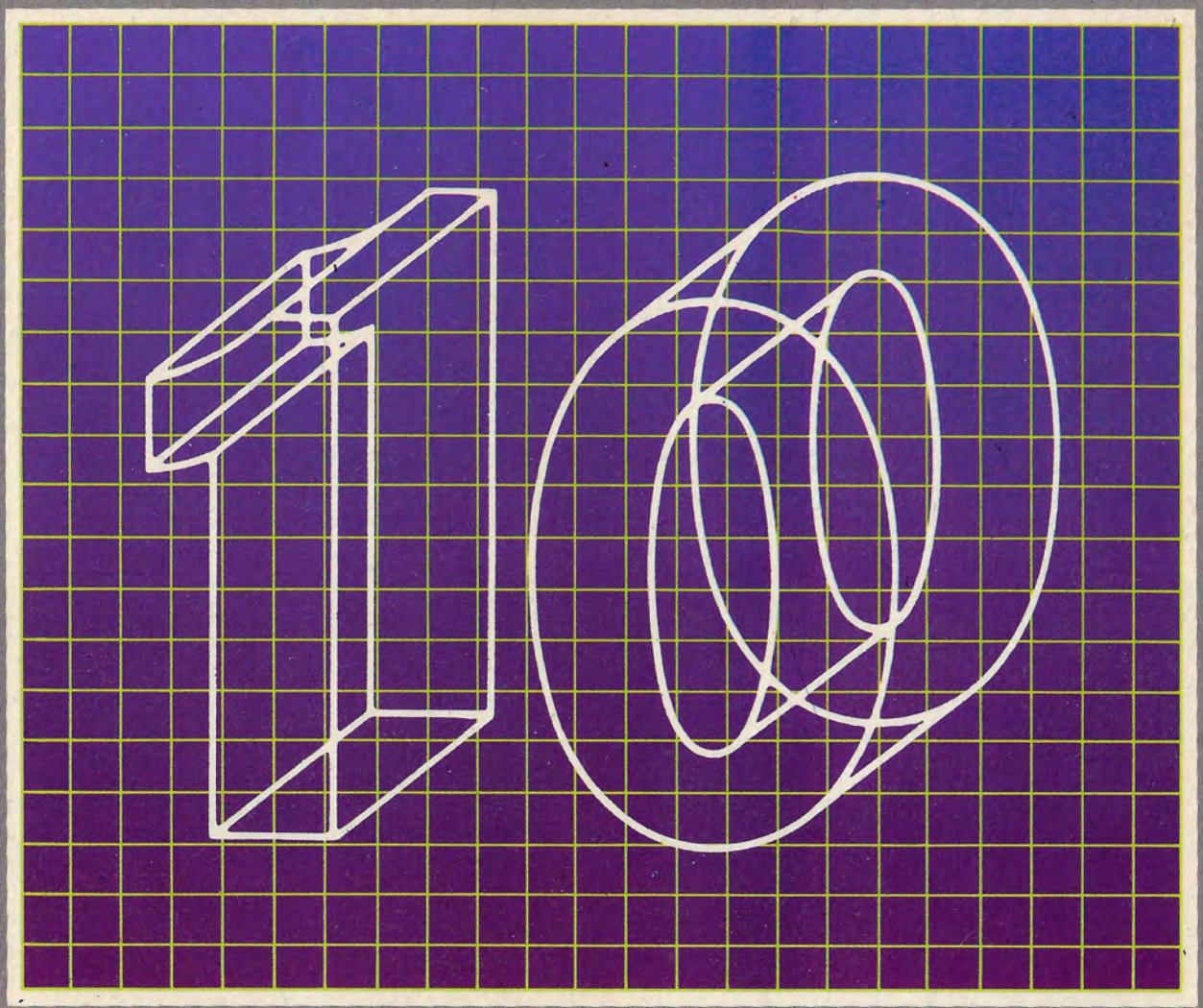
BIBLIOTECA PRACTICA

TRUCOS - SPRITES
BRICOLAGE DEL HARD
AULA ABIERTA
LOGO - TERMINOLOGIA

TALLER DE INFORMATICA

PROGRAMAS VALIDOS PARA AMSTRAD, IBM, SPECTRUM, COMMODORE Y MSX



DISFRUTA TU ORDENADOR, APRENDE Y HAZ COSAS CON EL

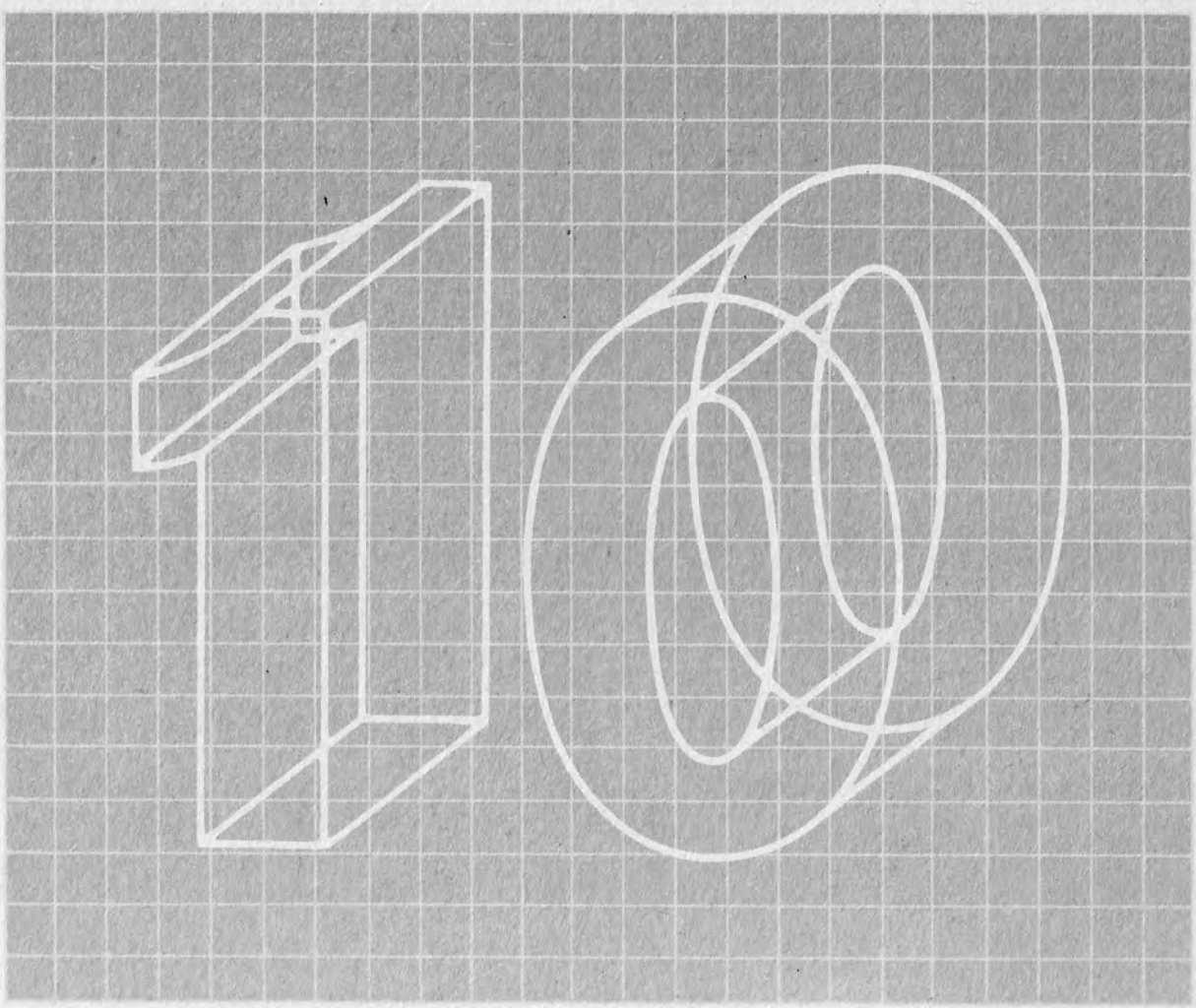
EDICIONES SIGLO CULTURAL

BIBLIOTECA PRACTICA

TALLER DE INFORMATICA

TRUCOS - SPRITES
BRICOLAGE DEL HARD
AULA ABIERTA
LOGO - TERMINOLOGIA

PROGRAMAS VALIDOS PARA AMSTRAD, IBM, SPECTRUM, COMMODORE Y MSX



EDICIONES SIGLO CULTURAL

Una publicación de

EDICIONES SIGLO CULTURAL, S. A.

Director-editor:
RICARDO ESPAÑOL CRESPO.

Gerente:
ANTONIO G. CUERPO.

Directora de producción:
MARIA LUISA SUAREZ PEREZ.

Director de la colección:
JOSE ARTECHE, Ingeniero de Telecomunicación

Diseño:
BRAVO-LOFISH.

Maquetación:
D. S. M.

Dibujos:
JOSE OCHOA

Tomo 10. «Experiencias y prácticas en logo», Carlos Albert, Técnico de Informática; Adoración Llena Jubero, Técnico de Informática. «Manejo de sprites y elementos gráficos», «Trucos y rutinas básicas», Francisco Morales. Técnico de Informática. «Aprende con el ordenador», AULA DE INFORMATICA APLICADA: Soledad Tamariz-Martel, Diplomada en Telecomunicación; Francisco Blanca, Diplomado en Telecomunicación; Alejandro Marcos, Licenciado en Química; Fernando Suero, Diplomado en Telecomunicación. «El Taller del Hardware», Carlos Rey, Ingeniero Industrial. «Pequeña historia de la Informática», «Temas monográficos de vanguardia», «Terminología», «Vocabulario de Informática», Blanca Arbizu, Técnico de Informática.

Ediciones Siglo Cultural, S. A.

Dirección, redacción y administración:
Pedro Teixeira, 8-2.°. Teléf. 810 52 13. 28020 Madrid.

Publicidad:
Gofar Publicidad, S. A. Benito de Castro, 12 bis. 28028 Madrid.

Distribución en España:
COEDIS, S. A. Valencia, 245. Teléf. 215 70 97. 08007 Barcelona.
Delegación en Madrid: Serrano, 165. Teléf. 411 11 48.

Distribución en Ecuador: Muñoz Hnos.

Distribución en Perú: DISELPESA.

Distribución en Chile: Alfa Ltda.

Importador exclusivo Cono Sur:
CADE, S.R.L. Pasaje Sud América, 1532. Teléf.: 21 24 64.
Buenos Aires - 1.290. Argentina.

ISBN del tomo: 84-7688-067-7
ISBN de la obra: 84-7688-047-2

Fotocomposición:
ARTECOMP, S. A. Albarracín, 50. 28037 Madrid.

Imprime:
MATEU CROMO. PINTO (Madrid).

Depósito legal: M-43.185-1986
Printed in Spain - Impreso en España.

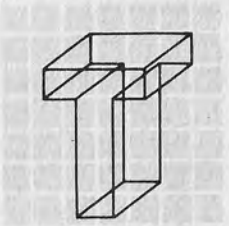
Suscripciones y números atrasados:
Ediciones Siglo Cultural, S. A.
Pedro Teixeira, 8-2.°. Teléf. 810 52 13. 28020 Madrid.

Abril, 1987.

INDICE

EXPERIENCIA Y PRACTICAS EN LOGO

Los operadores aritméticos



ODOS sabemos que con el ordenador podemos efectuar operaciones de cálculo de todo tipo a velocidades considerablemente altas; algunas de estas operaciones las expresamos a través de operadores aritméticos.

Con el LOGO podemos escribir expresiones del tipo:

4+5

en donde relacionamos la constante 4 con la constante 5 a través del operador +.

Este tipo de relaciones forman las expresiones aritméticas. Los datos a relacionar son los operandos; pueden ser constantes numéricas (datos fijos), variables numéricas (nombres que almacenan valores), funciones, o a su vez otras expresiones.

El resultado de estas expresiones puede ser:

— El dato que enviamos a un procedimiento con entrada

```
? CUADRADO 4 + 6
```

10 será el valor que le pasamos al procedimiento CUADRADO.

```
? CIRCULO 10 + :l
```

el dato que pasamos a CIRCULO es 10 más el valor que tenga la variable :L.

— El dato que acompaña a una primitiva

```
? AVANZA 17 + 7
```

la Tortuga avanzará 24 pasos.

— Un dato que queremos que aparezca en pantalla

```
? ESCRIBE 234 + 566
```

aparecerá en la pantalla el número 800.

Entre el operador y los operandos es conveniente dejar siempre un espacio en blanco. Un número negativo lo indicaremos sin dejar espacio entre el signo - y el número (-3).

Los operadores aritméticos son cuatro.

```
+ SUMA
- RESTA
* MULTIPLICACION
/ DIVISION
```

Hay dos formas de escribir las operaciones aritméticas, nosotros ahora sólo vamos a ver una de ellas, la llamada escritura infija, que corresponde a la que estamos acostumbrados a ver y a utilizar.

Esritura infija: los operadores se colocan entre los operandos.

Este tipo de expresiones tienen un orden de prioridad para cuando contienen a varios operadores diferentes.

Cuadro de prioridades.

```
1 /,*
2 -,+
```

Si los operadores de una expresión tienen la misma prioridad, ésta se evaluará de izquierda a derecha.

```
? ESCRIBE 10 + 20 - 40 + 50
40
? ESCRIBE 40 + 5 * 90
490
? ESCRIBE 24 + 5 - 3 * 10
-10
```

Las funciones de las teclas no son iguales en todas las versiones de LOGO.

```
? ESCRIBE 25 / 5 + 40 * 17
685
? ESCRIBE 36 * 5 / 6 + 4
34
```

Si, por ejemplo, en esta última expresión lo que queremos es calcular 36 * 5 y el resultado de dividirlo entre 6 + 50, lo haremos utilizando los paréntesis.

```
? ESCRIBE (36 * 5) / (6 + 4)
18
```

Como has podido comprobar, los paréntesis tienen prioridad a cualquiera de los operadores.

```
ESCRIBE (-3 + 9) / (-5 * 2)
2.4
? ESCRIBE 40 / 10 / (5 * 2)
0.4
```

Por último, te exponemos un ejemplo para que observes y compruebes con tu ordenador la importancia que tiene escribir bien una expresión aritmética.

```
? ESCRIBE 24+5 -3 *10
29
NO SE QUE DEBO HACER CON -30
```

Entiende -3 como un número negativo.

```
? ESCRIBE 24 + 5 -3 * 10
29
NO SE QUE DEBO HACER CON -30
? ESCRIBE 24+5 - 3* 10
-1
? ESCRIBE 24 + 5 + -3 * 10
-1
```

Estas últimas son correctas, y aunque expresadas de distinta forma, dan el mismo resultado.

Una utilidad de los paréntesis es clarificar las expresiones; la operación anterior podemos escribirla así:

```
? ESCRIBE (24+25) + (-3) * 10
```

o

```
? ESCRIBE (24+5) + (-3*10)
```

Practica con las expresiones aritméticas

Los cuatro procedimientos con entrada que a continuación exponemos están relacionados entre sí, ya que todos reciben el mismo dato.

POLIGONOS dibuja un polígono regular de :N lados y de longitud de lado :N * 4

```
? PARA POLIGONOS :N
> REPITE :N [ AV :N 4 GD 360 / : N ]
> FIN
```

CIRCULO dibuja un círculo cuyo tamaño depende del valor de la variable :L

```
? PARA CIRCULO :L
> REPITE 36 [AV :L GD 10 ]
> FIN
```

TRIANGULO dibuja un triángulo equilátero de lado :L * 10

```
? PARA TRIANGULO :L
> REPITE 3 [AV :L * 10 GD 120 ]
> FIN
```

DIBUJO inicializa la pantalla, determina la posición de los dibujos y llama a los procedimientos que los realizan. Es también un procedimiento con entrada, el valor que recibe la variable :L cuando es llamado el procedimiento, es el que luego reciben los otros tres procedimientos llamados.

```
? PARA DIBUJO :L
> PM
> OT
> BP
> GI 90
> SL
> PONY -30
> PONX -30
> BL
> POLIGONOS :L
> SL
> PONY COORY + 15
> BL
> CIRCULO :L
> SL
> PONX COORX + 40
> PONY COORY + 10
> BL
```

No obtenemos el mismo resultado utilizando las mismas teclas dentro y fuera del editor.

```
> TRIANGULO :L
> FIN
```

Observa que utilizamos como operandos funciones como COORX y COORY.

Ejecuta el dibujo dando a :L valores diferentes.

```
? DIBUJO 5
```

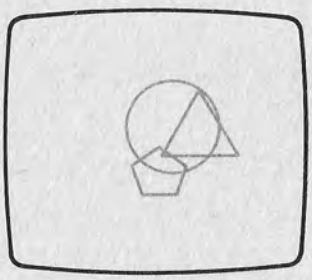


Fig. 1.

```
? DIBUJO 6
? DIBUJO 7
```

Si das a :L valores que estén fuera del intervalor [3 9], el dibujo que te aparecerá en la pantalla seguramente no corresponderá a lo que pretendíamos dibujar.

Para el SPECTRUM haz lo siguiente:

Suprime en el procedimiento DIBUJO la orden PM.

Cambia las siguientes órdenes:

OT	por	ET
BL	por	CL
PONY COORY	por	PONY YCOOR
PONX COORX	por	PONX XCOOR

Este otro ejemplo dibuja una casa, cuya altura depende del valor que le pasemos al procedimiento CASA cuando lo ejecutemos.

El procedimiento no está preparado para que varíe la anchura de la casa, porque o bien tendríamos que hacer cálculos excesivamente largos con la variable que establece la altura, o bien pasarle a CASA otra variable que estableciera la anchura de la casa.

CASA inicializa la pantalla y dibuja la casa. Para dibujar las ventanas llama al procedimiento VENTA.

```
? PARA CASA :L
> OT
> BP
> CENTRO
> BL
> REPITE 2 [ AV :L GD 90 AV 38 GD
  90 ]
> AV :L
> GD 50 AV 25
> GD 80 AV 25
> SL
> PONPOS [ 15 0 ]
> BL
> PONRUMBO 0
> AV :L / 4
> REPITE 18 [AV 1 GD 10 ]
> AV :L / 4
> SL
> PONX 5
> PONY :L - (:L / 3)
> BL
> PONRUMBO 90
> VENTA :L
> SL
> AV 21
> BL
> VENTA :L
> FIN
```

VENTA es un procedimiento con entrada, ya que la altura de las ventanas depende del valor que demos a la altura de la casa.

```
? PARA VENTA :L
> REPITE 2 [AV 7 GI 90 AV :L / 6 GI 90]
> FIN
```

Observa que en estos procedimientos operamos continuamente con operandos variables.

Para que te aparezca bien el dibujo en la pantalla te aconsejamos darle al procedimiento CASA valores comprendidos entre 10 y 70.

Por ejemplo:

```
? CASA 40
```

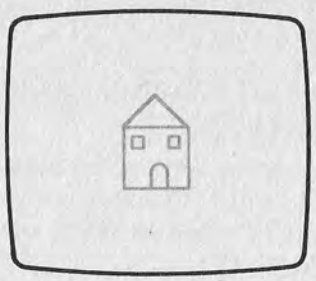


Fig. 2.

Los paréntesis tienen la máxima prioridad.

Para el SPECTRUM haz lo siguiente:

OT por ET
BL por CL

Consigue infinitos dibujos

```
? PARA POLI :N
> BP
> SL
> PONX '(:N * 2)
> BL
> REPITE: N [AV 20 REPITE 5 [ GI 360
  / 5 AV 20 ] GD 360 / :N]
> FIN
```

Ejecútalo tecleando:

```
? POLI 5
```

Este procedimiento dibuja un polígono regular de :N lados: cada lado es un pentágono.

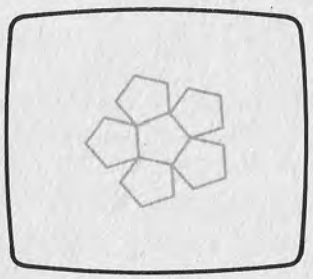


Fig. 3.

```
? PARA POLIGO :N
> BP
> SL
> PONX -(:N * 2)
> BL
> REPITE :N [ AV 20 REPITE 6 [ GI 360
  / 6 AV 20 ] GD 360 / :N ]
> FIN
```

Para ejecutarlo:

```
? POLIGO 6
```

Dibuja un polígono regular de :N lados: cada lado es un hexágono.

Si observas detenidamente los dos procedimientos anteriores descubrirás que podemos conseguir hacer lo mismo y más con un solo procedimiento.

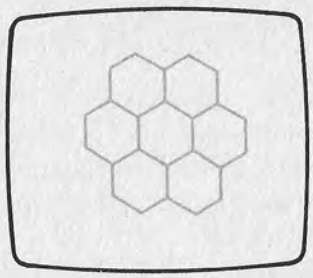


Fig. 4.

```
? PARA POLIGONOS :N
> BP
> SL
> PONX -(:N * 2)
> BL
> REPITE N: [ AV 20 REPITE :N [GI 360
  / :N AV 20 ] GD 360 / :N ]
> FIN
```

Prueba que es cierto ejecutando el procedimiento con valores distintos.

```
? POLIGONOS 5
? POLIGONOS 6
? POLIGONOS 7
```

Con los siguientes procedimientos podemos también dibujar gran cantidad de figuras distintas.

Utilizamos dos procedimientos. El primero sitúa a la Tortuga en la posición de comienzo y repite 8 veces la llamada a otro procedimiento que dibuja una figura geométrica cuyo número de lados lo determina el valor de la variable LADOS, variando su posición cada vez que termina de dibujar una.

Pasamos el valor de la variable al procedimiento FIGURA, desde el procedimiento GENERAL.

```
? PARA GENERAL :LADO
> BP
> SL
> GI 90 AV 60 GD 90
> BL
> REPITE 8 [ FIGURA :LADOS AV 30
  GD 45 SL AV 30 BL ]
> FIN

? PARA FIGURA :LADO
> REPITE :LADO [AV 30 GD 360/
  :LADO ]
> FIN
```

Una expresión aritmética da siempre como resultado un valor numérico.

Si ejecutamos el procedimiento GENERAL dando distintos valores a la variable LADOS, obtenemos los siguientes dibujos:

? GENERAL 3

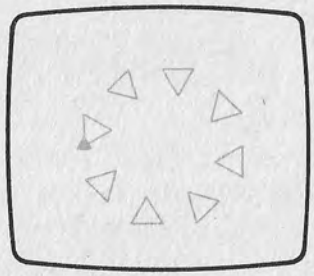


Fig. 5.

? GENERAL 4

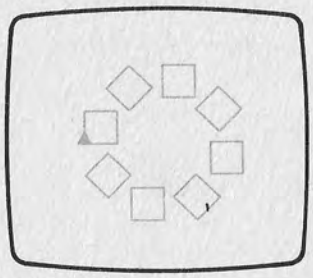


Fig. 6.

.? GENERAL 5

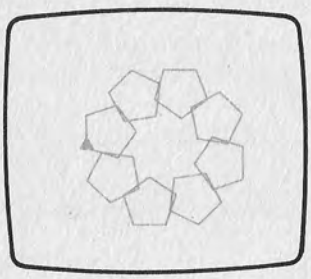


Fig. 7.

? GENERAL 6

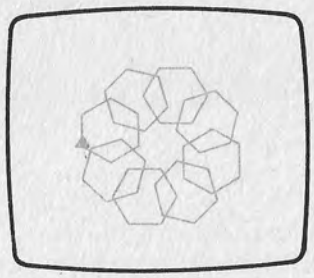


Fig. 8.

? GENERAL 10

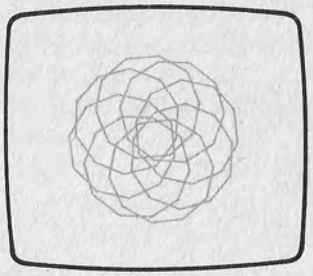


Fig. 9.

Con los siguientes procedimientos, obtenemos el dibujo, sobre unos ejes, de dos pares de rectas que se cortan en un punto.

Primero definimos el procedimiento que dibuja los ejes. Para hacer las escalas tanto horizontal como vertical, definimos otro procedimiento (ESCALAR), con una entrada que determina, según el eje, las veces que hay que repetir las rayas. Este es llamado desde el procedimiento EJES.

```
? PARA EJES
> SL
> PONPOS [ '100 0 ]
> BL
> PONPOS [ 100 0 ]
> SL
> PONPOS [ 0 -60 ]
> BL
> PONPOS [ 0 60 ]
> SL
> PONPOS [ -100 0 ]
> PONRUMBO 90
> ESCALAR 19
> SL
> PONPOS [ 0 -60 ]
> PONRUMBO 0
> ESCALAR 11
> FIN

? PARA ESCALAR :R
> AV 10 GI 90
> RE 2 BL
> REPITE :R [ AV 4 RE 4 GD 90 SL
                AV 10 GI 90 BL ]
> FIN
```

Una vez que tenemos los ejes, definimos otro procedimiento para dibujar las rectas. En

Podemos asignar un valor a una variable desde un procedimiento para que sea utilizado en otro.

este caso, indicamos los puntos extremos de cada una de ellas.

Aparecen dos rectas sobre los ejes que permanecen unos segundos en pantalla. Luego desaparecen y aparecen otras dos

```
? PARA RECTAS
> BP
> EJES
> SL
> PONPOS [ 40 40 ]
> BL
> PONPOS [ -30 -30 ]
> SL
> PONPOS [ 50 -20 ]
> BL
> PONPOS [ -20 50 ]
> ESPERA 200
> BP
> EJES
> SL
> PONPOS [ -60 20 ]
> BL
> PONPOS [ 60 20 ]
> SL
> PONPOS [ -80 -30 ]
> BL
> PONPOS [20 50 ]
> FIN
```

Puedes añadir más rectas con sólo indicar en este procedimiento los nuevos puntos.

Si ejecutas RECTAS:

```
? RECTAS
```

Obtendrás los dos dibujos siguientes:

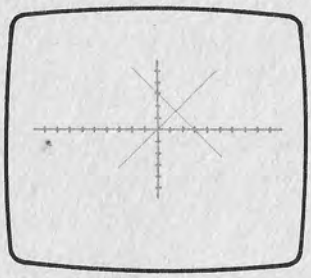


Fig. 10.

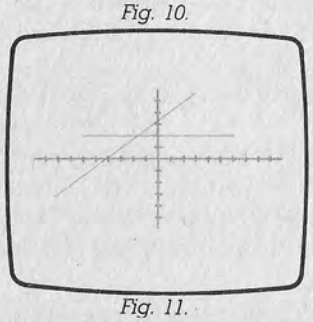


Fig. 11.

Cuadro resumen

— Operadores aritméticos por orden de prioridad.

()	paréntesis.
/, *	división y multiplicación
-, +	resta y suma

Ejercicios

1. *Dibuja un circuito eléctrico.*
2. *Cuadricula la pantalla definiendo un procedimiento que permita variar el tamaño de la cuadrícula tanto en horizontal como en vertical.*
3. *Realiza el siguiente dibujo:*

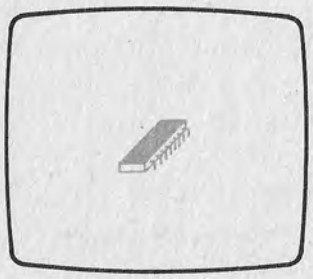


Fig. 12.

4. *El siguiente dibujo lo hemos realizado dibujando, en posiciones aleatorias de la pantalla, 100 cuadrados cuyo tamaño guarda relación con dicha posición.*

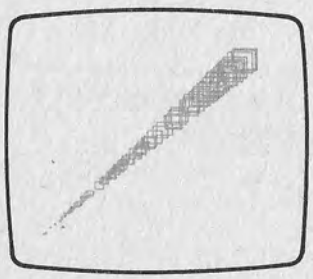


Fig. 13.

Intenta realizarlo.

5. *¿Son correctas las siguientes órdenes?*
   — ESCRIBE 10 + 5 - 6 / (-6) + 40 * 8
   — AVANZA 3 + 6 * - 9
   — RE COORY + 6

Con los paréntesis clarificamos las expresiones.

— ESCRIBE CL + 10
— GD 15 + RUMBO
— REPITE 15 [PONF FONDO + 1 ESPERA 10]

Solución a los ejercicios

1:

Dibujamos un circuito de tamaño y número componentes constante, por eso utilizamos un procedimiento sin entrada.

```
? PARA CIRCUITO
> BP
> SL
> PONX -50
> BL
> GD 35 AV 8
> GI 35 AV 30
> GI 35 AV 48
> GI 90 AV 6 RE 12 AV 6 GD 90
> SL AV 4
> GI 90 BL AV 4 RE 8 AV 4 GD 90
> AV 48 GD 90
> AV 20
> GD 65 AV 4
> REPITE 3 [ GI 130 AV 8 GD 130 AV 8 ]
> GI 130 AV 4 GD 65
> AV 20
> GD 90 AV 40 GI 90
> REPITE 36 [ AV 2 GD 10 ]
> SL GD 90 AV 24
> BL AV 36
> GD 90 AV 26
> FIN
```

Lo ejecutamos:

```
? CIRCUITO
```

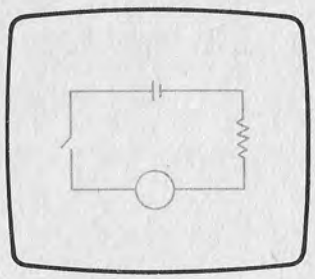


Fig. 14.

2:

Primero definimos un procedimiento que dibuja un marco en la pantalla de 250 puntos en horizontal por 155 en vertical para limitarla. Desde este procedimiento se llama a otros dos que cuadriculan la pantalla, los cuales utilizan una variable para determinar la anchura de la cuadrícula. Esta anchura puede ser diferente en horizontal y en vertical.

```
? PARA CUADRICULA
> BP
> SL
> PONPOS [ -126 -60 ]
> BL
> REPITE 2 [ AV 155 GD 90 AV 250 GD 90 ]
> EH 10
> EV 10
> FIN
```

En este caso la anchura de la cuadrícula está fijada en 10 puntos. Si quieres variarla, cambia los valores que aparecen en EH y EV.

El procedimiento EV dibuja las líneas verticales con una separación igual al valor de la variable ESC.

```
? PARA EH :ESC
> SL
> PONPOS [ -126 -60 ]
> PONRUMBO 0
> BL
> REPITE 250 / :ESC [ AV 155 RE 155 GD 90 AV :ESC GI 9 ]
> FIN
```

El procedimiento EV dibuja las líneas horizontales con una separación igual al valor de la variable ESC1.

```
> PARA EV
> SL
> PONPOS [-126 -60]
> PONRUMBO 90
> BL
> REPITE 155/ :ESC1 [ AV 250 RE 250 GI 90 AV :ESC1 GD 90 ]
> FIN
```

3:

Para dibujar el chip utilizamos dos procedimientos, uno que dibuja el cuerpo y otro

El dato que le enviamos a un procedimiento con entrada, puede ser toda una expresión aritmética.

que dibuja las patillas, y que es llamado desde el primero.

```
? PARA CHIP
> BP
> PONCL 1
> OT
> PONPOS [-10 0]
> BL
> REPITE 2 [AV 8 GD 90 AV 20 GD 90]
> AV 8 GD 45
> AV 50 GD 45
> AV 20 GD 135
> AV 50 GI 45
> AV 8 GI 135
> AV 50 GI 45
> AV 8
> PATILLAS
> FIN
? PARA PATILLAS
> RE 3 GI 135
> SL
> AV 6 GD 45
> REPITE 8 [BL RE 5 GI 90 AV 8
            GD 90 AV 1 GI 90 RE 7
            GI 90 RE 4 SL RE 1 GD 135
            AV 4 GD 45]
> SL GD 90 AV 8
> BL RELLENA
> FIN
> FIN
```

Fijamos el número de patillas en ocho. Para ejecutarlo:

```
? CHIP
```

4:

El procedimiento DIBUJO inicializa la pantalla y llama al procedimiento CUADRADO pasándole a la variable :A de dicho procedimiento un valor aleatorio.

```
? PARA DIBUJO
> BP
> REPITE 100 [ CUADRADO AZAR 180 ]
> FIN
```

CUADRADO dibuja un cuadrado cuyo lado es el valor de la variable :A dividido entre 8 (:A variable de entrada del procedimiento), en las coordenadas indicadas por PONX y PONY.

```
? PARA CUADRADO :A
> SL
> PONX :A - 90
> PONY :A - 90
> BL
> REPITE 4 [ AV :A / 8 GD 90 ]
> FIN
```

5:

— ESCRIBE 10 + 5 - 6 / ('6) + 40 * 8

CORRECTO. El resultado de la operación es 336

— AVANZA 30 + 6 * - 9

INCORRECTO. El signo menos está detrás de un signo de multiplicación y entre él y el nueve hay un espacio; por tanto, la multiplicación no es por -9.

El ordenador mostrará un mensaje de error:

Faltan datos para -

Sería correcto:

```
? AVANZA 30 + 60 * -9
```

— RE COORY + 6

CORRECTO. La Tortuga retrocederá el valor que tenga COORY más 6.

— ESCRIBE CL + 10

CORRECTO. LOGO nos devolverá la suma del código de color del lápiz más 10.

— GD 15 + RUMBO

CORRECTO. La Tortuga girará a la derecha 15 más el rumbo que en este momento lleve.

— REPITE 15 [PONF FONDO + 1 ESPERA 10]

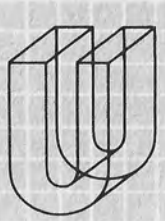
CORRECTO. Con este REPITE conseguimos variar el color de fondo de la pantalla 15 veces.

Un número negativo lo indicamos sin dejar espacio entre el signo - y el número.

Es conveniente dejar un espacio entre el operador y los operandos.

MANEJO DE SPRITES Y ELEMENTOS GRAFICOS

Control de movimiento



NA vez que hemos aprendido cómo mover cualquier tipo de figuras, ya sean sencillas o complejas, por la pantalla, nos vamos a meter en el mundo del control del movimiento.

Pero ¿qué es lo que entendemos por control de movimiento? En casi todos los programas con animación gráfica, el usuario tiene que controlar, mediante el teclado, joystick u otro periférico, el movimiento de algún gráfico por la pantalla. El programa tiene que estar preparado para poder leer de dicho periférico y para convertir las órdenes que le dé el usuario en movimientos de una cierta figura.

Así, pues, vamos a ver cómo podemos hacer que el usuario controle el movimiento de figuras por la pantalla mediante el uso del teclado, del joystick, del ratón o de cualquier otro periférico.

Lo primero que vamos a ver es un programa que nos permitirá mover un asterisco por toda la pantalla utilizando las siguientes teclas:

P - Ir hacia la derecha.
O - Ir hacia la izquierda.
Q - Ir hacia arriba.
A - Ir hacia abajo.

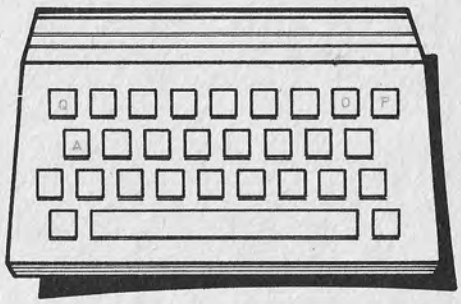


Fig. 1. Las teclas que controlan el movimiento del asterisco están distribuidas de forma que sean cómodas para el usuario.

El programa controla si hemos llegado al borde de la pantalla y se para en caso afirmativo. Dicho programa es el 1.

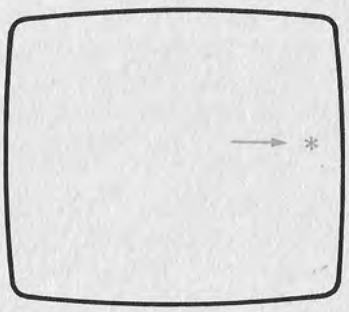


Fig. 2. Si el asterisco se sale de la pantalla, el ordenador dará un mensaje de error. Por ello, en nuestros programas tenemos que prever que esto no ocurra.

```
100 REM ********************************
110 REM * MOVIMIENTO CONTROLADO DE UN *
120 REM * ASTERISCO USANDO LAS TECLAS *
130 REM *                             *
140 REM *         P - DERECHA         *
150 REM *         O - IZQUIERDA       *
160 REM *         Q - ARRIBA          *
170 REM *         A - ABAJO           *
180 REM *                             *
190 REM ********************************
```

```
200 REM
210 CLS
220 LET X1=10
230 LET Y1=10
240 LET X=10
250 LET Y=10
260 LOCATE Y1,X1
270 PRINT " "
280 LOCATE Y,X
290 PRINT "*";
300 A$=INKEY$
310 IF A$<>"P" AND A$<>"O" AND A$<>"Q" AND A$<>"A" THEN GOTO 300
320 LET X1=X
330 LET Y1=Y
340 IF A$="P" THEN LET X=X+1
350 IF A$="O" THEN LET X=X-1
360 IF A$="Q" THEN LET Y=Y-1
370 IF A$="A" THEN LET Y=Y+1
380 IF X>40 THEN LET X=40
390 IF X<1 THEN LET X=1
400 IF Y>24 THEN LET Y=24
410 IF Y<1 THEN LET Y=1
420 GOTO 260
```

Las modificaciones que hay que realizar para utilizar el programa en ordenadores que no sean IBM o compatibles son las que aparecen a continuación.

COMMODORE:

```
210 PRINT CHR$(147)
260 POKE 214,Y1:POKE 211,Y1
280 POKE 214,Y:POKE 211,X
300 GET A$
380 IF X>39 THEN LET X=39
390 IF X<0 THEN LET X=0
410 IF Y<0 THEN LET Y=0
```

AMSTRAD:

```
260 LOCATE X1,Y1
280 LOCATE X,Y
400 IF Y>25 THEN LET Y=25
```

MSX:

```
260 LOCATE X1,Y1
280 LOCATE X,Y
```

SPECTRUM:

```
260 PRINT AT Y1,X1;
280 PRINT AT Y,X;
380 IF X>31 THEN LET X=31
390 IF X<0 THEN LET X=0
400 IF Y>21 THEN LET Y=21
410 IF Y<0 THEN LET Y=0
```

Ahora vamos a ver el funcionamiento del programa línea a línea para poder entender su funcionamiento.

Líneas 100 a 200. Cabecera del programa en líneas REM. En ellas se dice, aparte de la intención del programa, qué teclas hay que utilizar para que la estrella pueda moverse por la pantalla.

Línea 210. Borramos la pantalla.

Línea 220. Asignamos a la variable numérica X1 el valor 10. En esta variable almacenaremos el valor de la columna en la cual se encontraba el asterisco antes de que el usuario pulse algunas de las teclas que permiten el movimiento.

Línea 230. Se asigna a la variable Y1 el valor 10. Esta variable tiene la misma función que Y1, pero se encarga de almacenar la fila.

Línea 240. Almacenamos en X la columna donde se posicionará el asterisco al principio del movimiento y desde el cual empezaremos a moverlo.

Línea 250. Se asigna a la variable Y el valor 10. Esta variable, junto con la X, serán las que nos digan en qué lugar se encuentra el asterisco en un momento determinado o, lo que es igual, dónde vamos a imprimir el asterisco dentro de un momento.

Línea 260. Colocamos el cursor en la posición Y1,X1 de la pantalla (X1,Y1 en el AMSTRAD y MSX) para más tarde imprimir un espacio. La primera vez que se ejecuta esta instrucción, aparentemente no hace nada interesante, pero a partir de la segunda, se encarga de colocar el cursor en la antigua posición donde se encontraba el asterisco para borrarlo.

Línea 270. Imprimimos un espacio en blanco. Este será el que borre el asterisco.

Línea 280. Colocamos el cursor en la posición Y,X de la pantalla (X,Y en el AMSTRAD y MSX) para más adelante imprimir en dicha posición el asterisco.

Línea 290. Imprimimos el famoso asterisco (*).

Línea 300. Miramos si se ha pulsado alguna tecla desde el teclado. Si se ha pulsado alguna, ésta quedará almacenada en la variable alfanumérica A$. Si no se pulsó ninguna tecla, A$ será igual a la cadena vacía (A$="").

Línea 310. En esta línea nos aseguramos de que la tecla que ha pulsado el usuario es una de las permitidas. Por ello, se pregunta que si A$ no es igual a P y, a la vez, no es igual a O y tampoco igual a Q ni a A, entonces volver a mirar si se ha pulsado otra tecla. El programa estará moviéndose entre esta línea y la anterior hasta que el usuario pulse una de las teclas permitidas.

Línea 320. Como ya se ha pulsado una de las teclas que nos van a permitir el movimiento del asterisco, almacenamos en X1 la columna donde se encuentra actualmente.

Línea 330. Almacenamos en Y1 la fila donde se encuentra el asterisco. Esta línea y la anterior son las que nos van a permitir, cuando imprimamos el asterisco en su nueva posición, recordar dónde se encontraba antes para poder borrarlo.

Línea 340. A partir de esta línea empieza el reconocimiento de la tecla pulsada. En esta primera línea se pregunta si el usuario pulsó la tecla P. Esta es la que le dice al programa que mueva el asterisco un lugar hacia la derecha. Por ello, si se pulsó dicha tecla aumentamos el valor de la variable X en uno (LET X=X+1). Con ello aumentamos en uno la columna Y, cuando volvamos a la línea 280, el asterisco se imprimirá un lugar hacia la derecha.

Línea 350. Preguntamos si se ha pulsado la tecla O. En caso afirmativo, es que el usuario quiere moverse un lugar hacia la izquierda. Para movernos hacia la izquierda hay que decrementar la posición del asterisco en uno (LET X=X-1) para imprimirlo en su nueva posición en la línea 280.

Línea 360. En esta línea preguntamos si se ha pulsado la tecla Q. En caso afirmativo (el usuario desea moverse hacia arriba) decrementamos en uno el valor de la variable Y (LET Y=Y-1).

Línea 370. Por fin, si se ha pulsado la tecla A porque el asterisco ha de moverse hacia abajo, incrementamos el valor de la fila (de Y) en uno (LET Y=Y+1).

Una vez que sabemos en qué dirección tiene que moverse el asterisco y que tenemos almacenada dicha posición en las variables X e Y, vamos a comprobar que dicha posición no se encuentra fuera de la pantalla.

Línea 380. Si resulta que el asterisco ha llegado al borde derecho de la pantalla y lo ha sobrepasado, entonces hacemos que X adquiera el valor de la posición más a la derecha de ésta. En el caso del IBM, del AMSTRAD y del MSX esta posición corresponde a la columna 40 (si estamos en el modo de 40 columnas por línea). Si resulta que X es mayor de 40, entonces estamos fuera de la pantalla, por lo que tenemos que hacer que X valga 40. En el SPECTRUM, como sólo tiene 32 columnas y la última está numerada como la 31, entonces tenemos que cambiar esta línea como se ha visto un poco más arriba. En el COMMODORE también ocurre algo parecido, ya que, aunque tiene 40 columnas por línea, éstas están numeradas del 0 al 39.

Línea 390. Comprobamos que el asterisco no va a salirse de la pantalla por la izquierda. En caso de hacerlo, damos a la variables X el valor de la primera columna (0 ó 1, según los ordenadores).

Línea 400. Miramos si el asterisco va a salirse por la parte de abajo de la pantalla. En caso afirmativo, hacemos que se quede en la última línea. Esta línea es: la 24 para COMMODORE, MSX, AMSTRAD y IBM; la 21 para SPECTRUM.

Línea 410. Para terminar, en esta línea miramos si el asterisco va a salirse por la parte superior de la pantalla. En caso afirmativo, hacemos que la variable Y tenga el valor de la primera línea (0 ó 1, según el ordenador).

Línea 420. Hacemos que el control del programa se transfiera a la línea 260. A partir de esta línea colocamos el cursor en la antigua posición del asterisco, lo borraremos (pues todavía se encuentra ahí), colocaremos el cursor en la nueva posición, imprimiremos el asterisco y volveremos a esperar hasta que se pulse una tecla de las permitidas para mover el asterisco de nuevo.

Este programa no realiza ninguna función espectacular, pero ilustra perfectamente cómo hay que realizar la lectura del teclado para hacer que un objeto se mueva por

la pantalla de acuerdo con las órdenes del usuario.

El programa tiene un pequeño fallo. Este consiste en que si pulsamos simultáneamente las teclas Q y P, el asterisco no se desplaza con movimiento diagonal (en este caso sería movimiento diagonal positivo-negativo), sino que se mueve en la dirección de la última tecla pulsada. La mejor manera de arreglar esto es crear otro programa que contemple la pulsación de otras teclas para el movimiento diagonal. El programa 2 puede ser un ejemplo de cómo hacerlo.

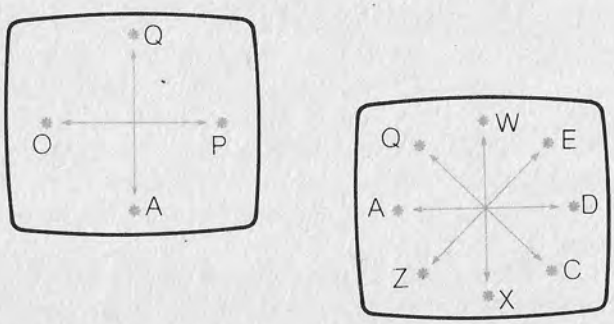


Fig. 3. Con el programa 1 sólo podíamos mover el cursor hacia arriba, abajo, derecha e izquierda, pero con el programa 2 también lo podremos mover en las cuatro diagonales.

```
100 REM ********************************
110 REM * MOVIMIENTO CONTROLADO DE UN *
120 REM * ASTERISCO USANDO LAS TECLAS *
130 REM *                             *
140 REM *         P - DERECHA         *
150 REM *         O - IZQUIERDA       *
160 REM *         Q - ARRIBA          *
170 REM *         A - ABAJO           *
180 REM *         Q - ARRIBA-IZQ.     *
190 REM *         E - ARRIBA-DER.     *
200 REM *         C - ABAJO-DER.      *
210 REM *         Z - ABAJO-IZQ.      *
220 REM *                             *
230 REM ********************************
240 REM
250 CLS
260 LET X1=10
270 LET Y1=10
280 LET X=10
290 LET Y=10
300 LOCATE Y1,X1
310 PRINT " "
320 LOCATE Y,X
330 PRINT "*";
340 A$=INKEY$
350 IF A$<>"D" AND A$<>"A" AND A$<>"W" AND A$<>"X" AND A$<>"Q" AND A$<>"E" AND A
$<>"C" AND A$<>"Z" THEN GOTO 340
360 LET X1=X
370 LET Y1=Y
380 IF A$="D" THEN LET X=X+1
390 IF A$="A" THEN LET X=X-1
400 IF A$="W" THEN LET Y=Y-1
410 IF A$="X" THEN LET Y=Y+1
420 IF A$="Q" THEN LET X=X-1:LET Y=Y-1
430 IF A$="E" THEN LET X=X+1:LET Y=Y-1
440 IF A$="C" THEN LET X=X+1:LET Y=Y+1
450 IF A$="Z" THEN LET X=X-1:Y=Y+1
460 IF X>40 THEN LET X=40
470 IF X<1 THEN LET X=1
480 IF Y>24 THEN LET Y=24
490 IF Y<1 THEN LET Y=1
500 GOTO 300
```

Las modificaciones que hay que hacer para COMMODORE, AMSTRAD, MSX y SPECTRUM son las siguientes:

COMMODORE:

250 PRINT CHR$(147)
300 POKE 214,Y1:POKE 211,X1

```
320 POKE 214,Y:POKE 211,X
340 GET A$
460 IF X>39 THEN LET X=39
470 IF X<0 THEN LET X=0
490 IF Y<0 THEN LET Y=0
```

AMSTRAD:

```
300 LOCATE X1,Y1
320 LOCATE X,Y
480 IF Y>25 THEN LET Y=25
```

MSX:

```
300 LOCATE X1,Y1
320 LOCATE X,Y
```

SPECTRUM:

```
300 PRINT AT Y1,X1;
320 PRINT AT Y,X;
460 IF X>31 THEN LET X=31
470 IF X<0 THEN LET X=0
480 IF Y>21 THEN LET Y=21
490 IF Y<0 THEN LET Y=0
```

Como puede verse en el programa, éste es muy parecido al anterior. Las únicas diferencias que hay entre ambos son:

— En la línea 350 se pregunta si la tecla que ha pulsado el usuario está dentro del siguiente grupo:

D A W X Q E C Z

— Se han aumentado las líneas que diagnostican la dirección en la que se tiene que mover el asterisco. Esto es necesario porque han aumentado el número de teclas que pueden usarse.

— Las teclas que se utilizan para mover el asterisco son distintas que las del programa anterior. Las del primer programa estaban puestas de manera que con una mano controlábamos el movimiento izquierda-derecha y con la otra arriba-abajo. Como ahora tenemos ocho teclas para pulsar, se han puesto todas juntas, pero en la dirección del movimiento que implica cada una. Se puede aprovechar el momento para recomendar al lector que, a la hora de hacer sus propios programas de animación, busque unas teclas que sean cómodas de manejar por el usuario y no poner unas con las cuales sea más fácil hacer el programa. De todas maneras, hablaremos de esto más adelante.

Las teclas que nos permiten mover el asterisco por la pantalla son las siguientes:

W - Movimiento hacia arriba.
X - Movimiento hacia abajo.
D - Movimiento hacia la derecha.
A - Movimiento hacia la izquierda.
Q - Movimiento diagonal arriba-izquierda
E - Movimiento diagonal arriba-derecha
C - Movimiento diagonal abajo-derecha
Z - Movimiento diagonal abajo-izquierda

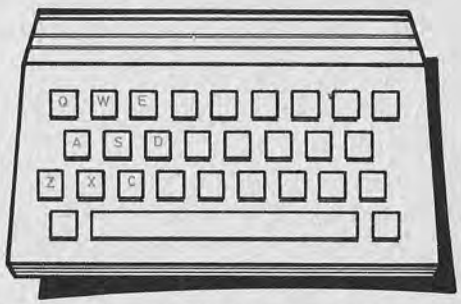


Fig. 4. Así están dispuestas las teclas, que nos permiten mover el asterisco, en el teclado de tu ordenador

Los dos programas que hasta ahora hemos visto, aunque funcionan, no son ningún alarde de programación, pues ni ahorran memoria ni hacen las cosas todo lo rápido que sería necesario. Estos dos programas se hicieron así porque para aumentar la velocidad del programa es necesario utilizar una serie de comandos que ordenadores como el SPECTRUM y el COMMODORE no tienen. A continuación se da un programa que nos permite mover el asterisco por la pantalla, pero realizado de una forma más lógica y más bonita. Aunque se van a incluir las modificaciones para SPECTRUM y COMMODORE, los usuarios de estos ordenadores podrán apreciar que para ellos esta nueva solución no es tan buena.

```
100 REM ********************************
110 REM * MOVIMIENTO CONTROLADO DE UN *
120 REM * ASTERISCO USANDO LAS TECLAS *
130 REM *                             *
140 REM *         P - DERECHA         *
150 REM *         O - IZQUIERDA       *
160 REM *         Q - ARRIBA          *
170 REM *         A - ABAJO           *
180 REM *         Q - ARRIBA-IZQ.     *
190 REM *         E - ARRIBA-DER.     *
200 REM *         C - ABAJO-DER.      *
210 REM *         Z - ABAJO-IZQ.      *
220 REM *                             *
230 REM ********************************
240 REM
250 CLS
```

```
260 LET X1=10
270 LET Y1=10
280 LET X=10
290 LET Y=10
300 REM
310 REM *** PROGRAMA GENERAL ***
320 REM
330 LOCATE Y1,X1
340 PRINT " "
350 LOCATE Y,X
360 PRINT "*";
370 A$=INKEY$
380 LET A=INSTR(" DAWXQECZ",A$)
390 IF A<2 THEN GOTO 370
400 LET X1=X
410 LET Y1=Y
420 ON A-1 GOSUB 480,500,520,540,560,580,600,620
430 GOSUB 670
440 GOTO 330
450 REM
460 REM *** PROCESAMIENTO DE LA TECLA ***
470 REM
480 LET X=X+1
490 RETURN
500 LET X=X-1
510 RETURN
520 LET Y=Y-1
530 RETURN
540 LET Y=Y+1
550 RETURN
560 LET X=X-1:LET Y=Y-1
570 RETURN
580 LET X=X+1:LET Y=Y-1
590 RETURN
600 LET X=X+1:LET Y=Y+1
610 RETURN
620 LET X=X-1:Y=Y+1
630 RETURN
640 REM
650 REM *** COMPROBACION DE POSICION ***
660 REM
670 IF X>40 THEN LET X=40
680 IF X<1 THEN LET X=1
690 IF Y>24 THEN LET Y=24
700 IF Y<1 THEN LET Y=1
710 RETURN
```

Las modificaciones que aparecen a continuación son las que son necesarias realizar para que el programa pueda funcionar en ordenadores distintos del IMB, PC, XT, AT y compatibles. Te volvemos a repetir que, aunque aparecen las modificaciones para SPECTRUM y COMMODORE, este programa no da buenos resultados en ellos.

COMMODORE:

```
250 PRINT CHR$(147)
330 POKE 214,Y1:POKE 211,X1
350 POKE 214,Y:POKE 211,X
370 GET A$
380 LEF B$=" DAWXQECZ":FOR I=1 TO
9:IF MID$(B$,I,1)=A$ THEN LET A=I
385 NEXT I
670 IF X>39 THEN LET X=39
680 IF X<0 THEN LET X=0
700 IF Y<0 THEN LET Y=0
```

AMSTRAD:

```
330 LOCATE X1,Y1
350 LOCATE X,Y
690 IF Y>25 THEN LET Y=25
```

MSX:

```
330 LOCATE X1,Y1
350 LOCATE X,Y
```

SPECTRUM:

```
330 PRINT AT Y1,X1;
350 PRINT AT Y,X;
380 LET B$=" DAWXQECZ":FOR I=1 TO
9:IF B$(I)=A$ THEN LET A=I
385 NEXT I
420 GOSUB 460+20*(A-1)
670 IF X>31 THEN LET X=31
680 IF X<0 THEN LET X=0
690 IF Y<21 THEN LET Y=21
700 IF Y<0 THEN LET Y=0
```

¿Cuáles son las diferencias entre el tercer programa y el cuarto? Estas son:

— En vez de usar una sentencia IF múltiple utilizamos la función INSTR para saber qué tecla ha pulsado el usuario. En el programa, segundo programa de este tomo, preguntábamos, mediante una sentencia IF, si el usuario había pulsado alguna de las teclas permitidas. En este tercer programa, gracias a la instrucción INSTR, podemos saberlo mucho antes. Esta instrucción nos dice si el contenido de la variable A$ se encuentra en la otra cadena alfanumérica que aparece dentro de los paréntesis. El resultado de esta función, que nosotros almacenamos en la variable numérica. A, nos dice en qué posición de la cadena se encuentra A$.

— Otra instrucción que se utiliza es la que está en la línea 420. Esta sirve para transferir el control del programa a una línea, pero dependiendo de un valor. Así, si la variable A tiene valor 2, iremos a la línea 480. Si tiene valor 3, a la 500. Con valor 4, a la 520, y así sucesivamente. Esta instrucción nos daría error si A tuviese un valor superior al número de nú-

meros que se encuentran después de la palabra GOSUB.

— La tercera diferencia es que, como después de usar la función ON ... GOSUB ya sabemos a qué línea nos tenemos que ir, en el grupo de líneas que va desde la 460 hasta 630 no ha sido necesario utilizar ninguna sentencia IF.

Por otra parte, se ha aprovechado para hacer el programa algo más bonito y estructurado mediante la utilización de subrutinas.

Como habrán visto los usuarios del SPECTRUM y del COMMODORE, la razón por la que en estos ordenadores el programa no aumentaría su velocidad es porque carecen de la sentencia INSTR. Como ésta ha habido que simularla mediante un bucle, hace que el programa se vuelva más lento. Por otro lado, el hecho de que el SPECTRUM no tenga la instrucción ON ... GOSUB no supone ningún problema, ya que se puede hacer algo parecido con la ayuda de operaciones matemáticas. Por ejemplo, si en un programa hecho para un ordenador distinto del SPECTRUM nos encontramos con una línea como la siguiente:

ON A GOSUB 100,200,300,400,500,600

y queremos pasar dicho programa al SPECTRUM, tendríamos que cambiar esta línea por:

GOSUB 100*A

Con esto se consigue que si A tiene valor 1, nos vayamos a la línea 100. Si tiene valor 2, a la 200. Si el valor de A es igual a 3, iremos a la 300, y así sucesivamente. También puede pasar que nos encontremos con que los números de línea que se encuentran después de la sentencia GOSUB (o GOTO según los casos) no guarden ninguna relación matemática ni lógica con el valor de la variable que determina el número de línea. Tal puede ser el caso de una línea como la siguiente:

ON A GOSUB 212,343,500,12,400

En este caso lo único que podemos hacer es cambiarla por lo siguiente:

GOTO 212*(A=1)+343*(A2)+500*(A=3)+
+12*(A=4)+400*(A=5)

Esto lo que hace es lo siguiente: Si A es igual a uno, entonces la igualdad (A=1) es cierta, por lo que el ordenador responde como si dicha igualdad fuese un uno (1). Como las demás igualdades de la línea (A=2, A=3, A=4 y A=5) no son ciertas, el ordenador responde a cada una de ellas como si de un cero se tratase, con lo que nos queda la línea algo así como:

GOTO 212*(1)+343*(0)+500*(0)+12*(0)+400*(0)

o lo que es igual:

GOTO 212

Si A valiese cuatro (en vez de uno), todas las igualdades menos (A=4) serían falsas y esa línea sería algo así como:

GOTO 212*(0)+343*(0)+500*(0)+12*(1)+400*(0)

que es lo mismo que:

GOTO 12

Esto último que acabamos de decir no es gratuito, sino que supone una de las modificaciones más fuertes que podemos hacerle aún al programa. Tal modificación consiste en anular el grupo de líneas que controlan si el asterisco ha salido de la pantalla. El programa 4 nos muestra cómo quedaría después de quitar este grupo de líneas y de controlar la posición del asterisco mediante un truco que tiene mucho que ver con lo que acabamos de explicar.

Operación	Resultado	Valor	
A=B	CIERTO	1	SPECTRUM
A=B	FALSO	0	SPECTRUM
A=B	CIERTO	-1	OTROS ORDENADORES
A=B	FALSO	0	OTROS ORDENADORES

Fig. 5.

```
100 REM ********************************
110 REM * MOVIMIENTO CONTROLADO DE UN *
120 REM * ASTERISCO USANDO LAS TECLAS *
130 REM *                             *
140 REM *        D - DERECHA          *
150 REM *        A - IZQUIERDA        *
160 REM *        W - ARRIBA           *
170 REM *        X - ABAJO            *
180 REM *        Q - ARRIBA-IZQ.      *
190 REM *        E - ARRIBA-DER.      *
200 REM *        C - ABAJO-DER.       *
210 REM *        Z - ABAJO-IZQ.       *
220 REM *                             *
230 REM ********************************
```

```
240 REM
250 CLS
260 LET X1=10
270 LET Y1=10
280 LET X=10
290 LET Y=10
300 REM
310 REM *** PROGRAMA GENERAL ***
320 REM
330 LOCATE Y1,X1
340 PRINT " "
350 LOCATE Y,X
360 PRINT "*";
370 A$=INKEY$
380 LET A=INSTR(" DAWXQECZ",A$)
390 IF A<2 THEN GOTO 370
400 LET X1=X
410 LET Y1=Y
420 ON A-1 GOSUB 480,500,520,540,560,580,600,620
430 GOTO 330
440 REM
450 REM *** PROCESAMIENTO DE LA TECLA ***
460 REM ***  Y CONTROL DEL ASTERISCO  ***
470 REM
480 LET X=X+1+(X=40)
490 RETURN
500 LET X=X-1-(X=1)
510 RETURN
520 LET Y=Y-1-(Y=1)
530 RETURN
540 LET Y=Y+1+(Y=24)
550 RETURN
560 LET X=X-1-(X=1):LET Y=Y-1-(Y=1)
570 RETURN
580 LET X=X+1+(X=40):LET Y=Y-1-(Y=1)
590 RETURN
600 LET X=X+1+(X=40):LET Y=Y+1+(Y=24)
610 RETURN
620 LET X=X-1-(X=1):LET Y=Y+1+(Y=24)
630 RETURN
```

Como siempre, las modificaciones que hay que realizar para ordenadores distintos del IBM son las siguientes:

COMMODORE:

```
250 PRINT CHR$(147)
330 POKE 214,Y1:POKE 211,X1
350 POKE 214,Y:POKE 211,X
370 GET A$
380 LEF B$=" DAWXQECZ":FOR I=1 TO
9:IF MID$(B$,I,1)=A$ THEN LET A=I
385 NEXT I
480 LET X=X+1+(X=39)
500 LET X=X-1-(X=0)
520 LET Y=Y-1-(Y=0)
560 LET      X=X-1-(X=0):LET
Y=Y-1-(Y=0)
580 LET      X=X+1+(X=39):LET
Y=Y-1-(Y=0)
600 LET      X=X+1+(X=39):LET
Y=Y+1+(Y=24)
620 LET      X=X-1-(X=0):LET
Y=Y+1+(Y=24)
```

AMSTRAD:

```
330 LOCATE X1,Y1
350 LOCATE X,Y
540 LET Y=Y+1+(Y=25)
600 LET      X=X+1+(X=40):LET
Y=Y+1+(Y=25)
620 LET      X=X-1-(X=1):LET
Y=Y+1+(Y=25)
```

MSX:

```
330 LOCATE X1,Y1
350 LOCATE X,Y
```

SPECTRUM:

```
330 PRINT AT Y1,X1;
350 PRINT AT Y,X;
380 LET B$=" ADWXQECZ":FOR I=1 TO
9:IF B$(I)=A$ THEN LET A=I
385 NEXT I
480 LET X=X+1-(X=31)
500 LET X=X-1+(X=0)
520 LET Y=Y-1+(Y=0)
540 LET Y=Y+1-(Y=21)
560 LET      X=X-1+(X=0):LET
Y=Y-1+(Y=0)
580 LET      X=X+1-(X=31):LET
Y=Y-1+(Y=0)
600 LET      X=X+1-(X=31):LET
Y=Y+1-(Y=21)
620 LET      X=X-1+(X=0):LET
Y=Y+1-(Y=21)
```

Como puede verse en el programa, el control de la posición del asterisco lo estamos realizando al mismo tiempo que incrementamos o decrementamos su fila y/o su columna. ¿Qué es lo que estamos haciendo? Para verlo vamos a coger como ejemplo la línea 540.

```
540 LET Y=Y+1+(Y=24)
```

Más arriba dijimos que las operaciones lógicas como la que vemos entre paréntesis en la línea de arriba, el ordenador las evalúa y mira si son verdaderas o falsas. En el caso de ser falsa, es como si dicha igualdad fuese un cero. Si la igualdad es verdadera, entonces el ordenador le asigna el valor -1 (en el SPEC-

TRUM es valor 1). según esto, si la igualdad de la línea de arriba es cierta, nos quedaría como:

```
540 LET Y=Y+1+(-1)
```

o lo que es igual:

```
540 LET Y=Y
```

ya que +1+(-1) es igual a +1-1 e igual a 0

+1+(-1)=+1-1=0

Con ello se consigue que el valor de la fila no pueda variar. Si la igualdad hubiese sido falsa, entonces se hubiese incrementado en uno el valor de la Y.

En el tomo siguiente veremos más trucos y más formas de controlar el movimiento de figuras por la pantala. De momento te recomiendo, como siempre que vemos algo nuevo, que intentes hacer tus propios programas y que no te conformes con probar los que aquí te vamos dando.

TRUCOS Y RUTINAS BASICAS

ONTINUANDO con el programa que nos estamos fabricando desde hace unos cuantos tomos, vamos a ver en éste otras rutinas que se utilizarán para hacer el programa. Antes de comenzar con ellas volvemos a dar dos programas que, por error, aparecieron con una numeración equivocada en anteriores tomos. Estos dos programas son:

— Rutina de ordenación alfanumérica.
— Rutina de definición de impresión.

La primera de ellas tiene que estar localizada a partir de la línea 8600. La segunda, a partir de la 7000. Este nuevo cambio de números de línea se debe a que anteriormente estaban localizadas en unos números de línea que ya estaban ocupados por otras rutinas. Estas dos rutinas aparecen en este tomo como programas 1 y 2.

```
8600 REM
8601 REM ****************************
8602 REM * ORDENACION ALFANUMERICA *
8603 REM ****************************
8604 REM
8605 CLS
8606 PRINT "O R D E N A R   F I C H E R O"
8607 PRINT "=============================="
8608 PRINT
8609 PRINT "Ordenando fichero por el campo No. ";NC
8610 PRINT
8611 PRINT "ESPERA UN MOMENTO"
8612 LET CC=0
8613 FOR Z=1 TO TT
8614    IF MID$(F$(20),Z,1)=CHR$(254) THEN LET CC=CC+1
8615    IF CC=NC-1 THEN LET Z1=Z:LET Z=TT
8616 NEXT Z
8617 IF CC=NC-1 THEN GOTO 8628
8618 CLS
8619 PRINT "E R R O R"
8620 PRINT "---------"
8621 PRINT
8622 PRINT "Lo siento pero no he encontrado el campo No. ";NC
8623 PRINT
8624 PRINT "Estas seguro de que las fichas de este fichero tienen";NC;"campos?"
8625 X=1:Y=20:GOSUB 8200
8626 CLS
8627 RETURN
8628 FOR Z=1 TO NN
8629    FOR X=1 TO Z
8630       IF RIGHT$(F$(Z),Z1)<RIGHT$(F$(X),Z1) THEN LET A$=F$(Z):LET F$(Z)=F$(X
):LET F$(X)=A$
```

```
8631    NEXT X
8632 NEXT Z
8633 CLS
8634 PRINT "O P E R A C I O N   T E R M I N A D A"
8635 PRINT "======================================="
8636 PRINT
8637 PRINT "El fichero esta ordenado por el campo No. ";NC
8638 X=1:Y=20:GOSUB 8200
8639 CLS
8640 RETURN


     7000 REM
     7001 REM ***************************
     7002 REM * DEFINICION DE IMPRESION *
     7003 REM ***************************
     7004 REM
     7005 CLS
     7006 PRINT "DEFINICION DE LA IMPRESION"
     7007 PRINT "=========================="
     7008 LOCATE 5,1
     7009 PRINT "( CUANTOS CAMPOS QUIERE QUE IMPRIMA ?"
     7010 LOCATE 12,1
     7011 LINE INPUT "==> ";A$
     7012 IF VAL(A$)=0 THEN RETURN
     7013 LET N=VAL(A$)
     7014 DIM N$(N,4)
     7015 LOCATE 5,1
     7016 PRINT "( QUE CAMPOS DESEA QUE IMPRIMA ?      "
     7017 LOCATE 9,1
     7018 PRINT "CAMPOS A IMPRIMIR  = "
     7019 FOR Z=1 TO N
     7020    LOCATE 12,1
     7021    PRINT SPACE$(10)
     7022    LOCATE 12,1
     7023    LINE INPUT "==> ";N$(Z,1)
     7024    IF VAL(N$(Z,1))=0 THEN GOTO 7020
     7025    LOCATE 9,22
     7026    FOR X=1 TO Z
     7027       PRINT N$(X,1);",";
     7028    NEXT X
     7029 NEXT Z
     7030 PRINT CHR$(29);" "
     7031 LOCATE 5,1
     7032 PRINT "( EN QUE ORDEN LOS IMPRIMO ?          "
     7033 PRINT "INTRODUZCA EL NUMERO DEL CAMPO EN EL "
     7034 PRINT "ORDEN EN QUE QUIERE QUE SE IMPRIMA.  "
     7035 LOCATE 10,1
     7036 PRINT "ORDEN DE IMPRESION = "
     7037 FOR Z=1 TO N
     7038    LOCATE 12,1
     7039    PRINT SPACE$(10)
     7040    LOCATE 12,1
     7041    LINE INPUT "==> ";N$(Z,2)
     7042    IF VAL(N$(Z,2))=0 THEN GOTO 7038
     7043    LOCATE 10,22
     7044    FOR X=1 TO Z
     7045       PRINT N$(X,2);",";
     7046    NEXT X
     7047 NEXT Z
     7048 PRINT CHR$(29);" "
     7049 LOCATE 12,1
     7050 PRINT "          ":REM 10 ESPACIOS EN BLANCO
     7051 LET NN=0
```

```
7052 FOR Z=1 TO N
7053    FOR X=1 TO N
7054       IF N$(Z,1)=N$(X,2) THEN LET NN=NN+1:LET X=N+1
7055    NEXT X
7056 NEXT Z
7057 IF NN=N THEN GOTO 7077
7058 CLS
7059 PRINT "CAMPOS A IMPRIMIR =  ";
7060 FOR Z=1 TO N
7061    PRINT N$(Z,1);",";
7062 NEXT Z
7063 PRINT CHR$(29);" "
7064 PRINT
7065 PRINT
7066 PRINT "ORDEN DE IMPRESION = ";
7067 FOR Z=1 TO N
7068    PRINT N$(Z,2);",";
7069 NEXT Z
7070 PRINT CHR$(29);" "
7071 LOCATE 12,1
7072 PRINT "NO COINCIDEN LOS CAMPOS A IMPRIMIR"
7073 PRINT "CON EL ORDEN DE IMPRESION"
7074 LET X=1:LET Y=20
7075 GOSUB 8200
7076 GOTO 7000
7077 LOCATE 15,1
7078 PRINT "( ESTA DE ACUERDO ? (S/N)"
7079 LET A$=INKEY$
7080 IF A$="S" OR A$="s" THEN GOTO 7084
7081 IF A$<>"N" AND A$<>"n" THEN GOTO 7079
7082 ERASE N$
7083 GOTO 7000
7084 FOR Z=5 TO 20
7085    LOCATE Z,1
7086    PRINT SPACE$(40)
7087 NEXT Z
7088 LOCATE 5,1
7089 PRINT "INTRODUZCA AHORA EN QUE COLUMNA"
7090 PRINT "DE LA IMPRESORA SE HA DE IMPRI-"
7091 PRINT "MIR CADA CAMPO."
7092 FOR Z=1 TO N
7093    LOCATE 12,1
7094    PRINT "CAMPO No. ";N$(Z,1);" ";
7095    LINE INPUT "==> ";N$(Z,3)
7096    IF VAL(N$(Z,3))=0 THEN GOTO 7093
7097 NEXT Z
7098 FOR Z=5 TO 17
7099    LOCATE Z,1
7100    PRINT SPACE$(40)
7101 NEXT Z
7102 LOCATE 5,1
7103 FOR Z=1 TO N
7104    PRINT "EL CAMPO ";N$(Z,1);" SE TABULARA EN LA COLUMNA ";N$(Z,3)
7105 NEXT Z
7106 LOCATE 20,1
7107 PRINT "( ESTA DE ACUERDO ? (S/N)"
7108 LET A$=INKEY$
7109 IF A$="S" OR A$="s" THEN GOTO 7112
7110 IF A$<>"N" AND A$<>"n" THEN GOTO 7108
7111 GOTO 7084
7112 FOR Z=5 TO 20
7113    LOCATE Z,1
7114    PRINT SPACE$(40)
7115 NEXT Z
7116 LOCATE 5,1
```

```
7117 PRINT "( CUANTAS LINEAS DEJO ENTRE FICHA"
7118 PRINT "Y FICHA? PULSA -1 PARA CAMBIAR DE"
7119 PRINT "PAGINA."
7120 LOCATE 12,1
7121 LINE INPUT "==> ";N$(1,4)
7122 IF VAL(N$(1,4))=0 THEN GOTO 7120
7123 CLS
7124 PRINT "OPERACION COMPLETADA"
7125 PRINT "====================="
7126 LET X=1:LET Y=20
7127 GOSUB 8200
7128 CLS
7129 RETURN
```

Para que no tengáis que mirar en tomos anteriores y realizar vosotros mismos las modificaciones necesarias para que el programa funcione en ordenadores distintos del IBM, os volvemos a repetir dichas modificaciones con los nuevos números de línea.

Las modificaciones al primer programa son:

COMMODORE:

8605 PRINT CHR$(147)
8618 PRINT CHR$(147)
8626 PRINT CHR$(147)
8633 PRINT CHR$(147)
9639 PRINT CHR$(147)

SPECTRUM:

8614 IF F$(1,Z)=CHR$(254) THEN LET CC=CC+1
8630 IF F$(Z,Z1 TO)<F$(X,Z1 TO) THEN LET A$=F$(Z):LET F$(Z)=F$(X):LET F$(X)=A$

Con respecto al segundo programa, las modificaciones son:

COMMODORE:

7005 PRINT CHR$(147)
7008 POKE 214,5:POKE 211,0
7010 POKE 214,12:POKE 211,0
7015 POKE 214,5:POKE 211,0
7017 POKE 214,9:POKE 211,0
7020 POKE 214,12:POKE 211,0
7021 FOR X=1 TO 10:PRINT " ";:NEXT X
7022 POKE 214,12:POKE 211,0
7025 POKE 214,9:POKE 21,21
7030 PRINT "<CURSOR-IZQ>";" "
7031 POKE 214,5:POKE 211,0
7035 POKE 214,10:POKE 211,0
7038 POKE 214,12:POKE 211,0
7039 FOR X=1 TO 10:PRINT " ";:NEXT X
7040 POKE 214,12:POKE 211,0
7043 POKE 214,10:POKE 211,21
7048 PRINT "<CURSOR-IZQ>";" "
7049 POKE 214,12:POKE 211,0
7058 PRINT CCR$(147)
7063 PRINT "<CURSOR-IZQ>";" "
7070 PRINT "<CURSOR-IZQ>";" "
7071 POKE 214,12:POKE 211,0
7077 POKE 214,15:POKE 211,0
7079 GET A$
7082 REM
7085 POKE 214,Z:POKE 211,0
7086 FOR X=1 TO 40:PRINT " ";:NEXT X
7088 POKE 214,5:POKE 211,0
7093 POKE 214,12:POKE 211,0
7099 POKE 214,Z:POKE 211,0
7100 FOR X=1 TO 40:PRINT " ";:NEXT X
7102 POKE 214,5:POKE 211,0
7106 POKE 214,20:POKE 211,0
7108 GET A$
7113 POKE 214,Z:POKE 211,0
7114 FOR X=1 TO 40:PRINT " ";:NEXT X
7116 POKE 214,5:POKE 211,0
7120 POKE 214,12:POKE 211,0
7123 PRINT CHR$(147)
7128 PRINT CHR$(147)

SPECTRUM:

7008 PRINT AT 5,0;
7008 PRINT AT 5,0;
7010 PRINT AT 12,0;
7011 INPUT LINE "==> ";A$
7014 DIM N$(N,4,3)
7015 PRINT AT 5,0;
7017 PRINT AT 9,0;
7020 PRINT AT 12,0;
7021 FOR X=1 TO 10:PRINT " ";:NEXT X
7022 PRINT AT 12,0;
7023 INPUT LINE "==> ";N$(Z,1)

```
7025 PRINT AT 9,21;
7030 REM
7031 PRINT AT 5,0;
7035 PRINT AT 10,0;
7038 PRINT AT 12,0;
7039 FOR X=1 TO 10:PRINT " ";:NEXT X
7040 PRINT AT 12,0;
7041 INPUT LINE "==> ";N$(Z,2)
7043 PRINT AT 10,21;
7048 REM
7049 PRINT AT 12,0;
7063 REM
7070 REM
7071 PRINT AT 12,0;
7077 PRINT AT 15,0;
7082 REM
7085 PRINT AT Z,0;
7086  FOR X=1 TO 40PRINT " ";:NEXT X
7088 PRINT AT 5,0;
7093 PRINT AT 12,0;
7095 INPUT LINE "==> ";N$(Z,3)
7099 PRINT AT Z,0;
7100  FOR X=1 TO 40:PRINT " ";:NEXT X
7102 PRINT AT 5,0;
7106 PRINT AT 20,0;
7113 PRINT AT Z,0;
7114  FOR X=1 TO 40:PRINT " ";:NEXT X
7116 PRINT AT 5,0;
7120 PRINT AT 12,0;
7121 INPUT LINE "==> ";N$(1,4)
```

AMSTRAD Y MSX:

```
7008 LOCATE 1,5
7010 LOCATE 1,12
7015 LOCATE 1,5
7017 LOCATE 1,9
7020 LOCATE 1,12
7022 LOCATE 1,12
7025 LOCATE 22,9
7031 LOCATE 1,5
7035 LOCATE 1,10
7038 LOCATE 1,12
7040 LOCATE 1,12
7043 LOCATE 22,10
7049 LOCATE 1,12
7071 LOCATE 1,12
7077 LOCATE 1,15
7085 LOCATE 1,Z
7088 LOCATE 1,5
7093 LOCATE 1,12
7099 LOCATE 1,Z
7102 LOCATE 1,5
7106 LOCATE 1,20
7113 LOCATE 1,Z
7116 LOCATE 1,5
7120 LOCATE 1,12
```

El primer programa que vamos a ver en este tomo va a ser la rutina que se encargará de definir la forma del fichero. Esta rutina le preguntará al usuario cuántos campos habrá por registro y cuántos caracteres por campo. Esto es necesario para, a la hora de la entrada de datos y a la hora de imprimir los diferentes registros, saber dónde empezar y dónde terminar. Este programa nos va a permitir configurar nuestro fichero como nosotros queramos o bien como uno que ya hubiésemos configurado. En el caso de que queramos hacer un fichero que sea muy grande y éste no nos quepa de una sola vez en memoria, necesitaremos crear más de un fichero con la misma estructura. Gracias a esta rutina sólo necesitamos definir el primer fichero, pues hay una opción de menú que copia la estructura de un fichero en el nuevo fichero que queramos realizar. Este programa es el 3 y aparece a continuación:

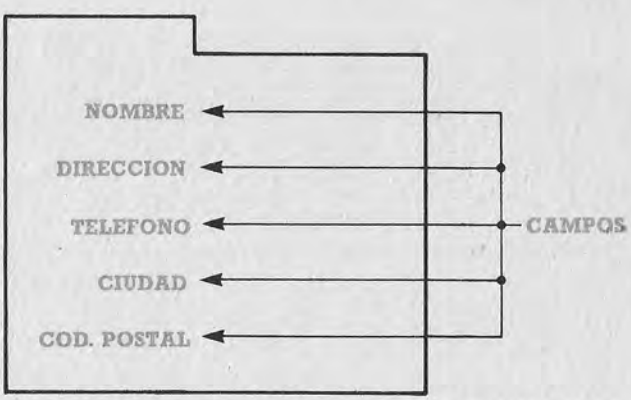


Fig. 1. Con esta rutina configuramos la forma del fichero. Le diremos al ordenador el número de campos y sus nombres.

```
9000 REM
9001 REM ***********************************
9002 REM * DEFINICION DE UN FICHERO NUEVO *
9003 REM ***********************************
9004 REM
9005 CLS
```

```
9006 LET A$="CREACION DE UN FICHERO"
9007 LET N=2
9008 LET A$(1)="UTILIZAR UNO QUE ESTE CREADO"
9009 LET A$(2)="CREAR UNO NUEVO"
9010 GOSUB 8400
9011 IF A$="1" THEN GOTO 9039
9012 PRINT " CREACION DE UN NUEVO FICHERO"
9013 PRINT "-----------------------------"
9014 PRINT:PRINT
9015 PRINT " Por favor, responda a las preguntas quele van a aparecer a continua
cion."
9016 PRINT "========================================"
9017 LOCATE 9,1
9018 INPUT "Cuantos campos tendra cada ficha?";N(1)
9019 IF N(1)<0 OR N(1)>20 THEN GOTO 9017
9020 DIM M$(N(1))
9021 PRINT
9022 FOR Z=1 TO N(1)
9023    LOCATE 16,1
9024    PRINT "
9025    LOCATE 18,1
9026    PRINT "
9027    LOCATE 16,1
9028    PRINT "Cuantos caracteres ocupa el campo ";Z
9029    INPUT N(Z+1)
9030    LOCATE 18,1
9031    PRINT "Como se llama este campo"
9032    INPUT M$(Z)
9033 NEXT Z
9034 CLS
9035 PRINT "OPERACION TERINADA"
9036 LET X=1:LET Y=20
9037 GOSUB 8200
9038 RETURN
9039 REM
9040 REM *** UTILIZACION DE UN FICHERO YA CREADO ***
9041 REM
9042 CLS
9043 PRINT " CREACION DE UN FICHERO"
9044 PRINT "======================="
9045 PRINT
9046 PRINT " Teclea el nombre del fichero que ya tiene una estructura definida y
 del cual laquieres copiar."
9047 PRINT:PRINT:PRINT
9048 INPUT "NOMBRE = ";N$
9049 PRINT:PRINT
9050 PRINT " Asegurate de que esta colocada la cintao disco que contiene dicho f
ichero"
9051 LET X=1:LET Y=20
9052 GOSUB 8200
9053 CLS
9054 PRINT "LEYENDO ... ";N$
9055 OPEN N$ FOR INPUT AS #1
9056 INPUT#1,TT
9057 DIM M$(TT)
9058 FOR Z=1 TO TT
9059    INPUT#1,N(Z+1)
9060 NEXT Z
9061 LET N(1)=TT
9062 FOR Z=1 TO TT
9063    INPUT#1,M$(Z)
9064 NEXT Z
9065 CLS
9066 PRINT "CONFIGURACION LEIDA"
```

```
9067 PRINT:PRINT
9068 PRINT "Cada ficha tiene ";TT;" campos"
9069 PRINT
9070 FOR Z=1 TO TT
9071    PRINT "el campo ";Z;" se llama ";M$(Z)
9072 NEXT Z
9073 LET X=1:LET Y=21
9074 GOSUB 8200
9075 RETURN
```

Las modificaciones que hay que realizar en este programa son muy numerosas debido a que cada ordenador tiene su propia forma de crear los ficheros. Por ejemplo, en el SPECTRUM no se pueden crear ficheros. Por ejemplo, en el SPECTRUM no se pueden crear ficheros, pero se pueden almacenar matrices y vectores. También hay diferencias notables según se esté trabajando con disco o cassette. Aun así, como el programa merece la pena, a continuación aparecen las modificaciones para todos los ordenadores distintos del IBM:

COMMODORE:

```
9005 PRINT CHR$(147)
9017 POKE 214,9:POKE 211,0
9023 POKE 214,16:POKE 211,0
9025 POKE 214,18:POKE 211,0
9027 POKE 214,16:POKE 211,0
9030 POKE 214,18:POKE 211,0
9034 PRINT CHR$(147)
9042 PRINT CHR$(147)
9053 PRINT CHR$(147)
9055 OPEN1,1,0,N$
9065 PRINT CHR$(147)
```

AMSTRAD:

```
9017 LOCATE 1,9
9023 LOCATE 1,16
9025 LOCATE 1,18
9027 LOCATE 1,16
9030 LOCATE 1,18
9055 OPENIN N$
9056 INPUT#9,TT
9059 INPUT#9,N(Z+1)
9063 INPUT#9,M$(Z)
```

MSX:

```
9017 LOCATE 1,9
9023 LOCATE 1,16
9025 LOCATE 1,18
9027 LOCATE 1,16
9030 LOCATE 1,18
9055 OPEN "CAS:"+M$ FOR INPUT AS
#1
```

SPECTRUM:

```
9017 PRINT AT 9,0;
9023 PRINT AT 16,0;
9025 PRINT AT 18,0;
9027 PRINT AT 16,0;
9030 PRINT AT 18,0;
9055 DIM N(21):DIM M$(20,30):LOAD N$
DATA N():LOAD N$
DATA M$()
9056 LET TT=N(1)
9057 REM
9058 REM
9059 REM
9060 REM
9061 REM
9062 REM
9063 REM
9064 REM
```

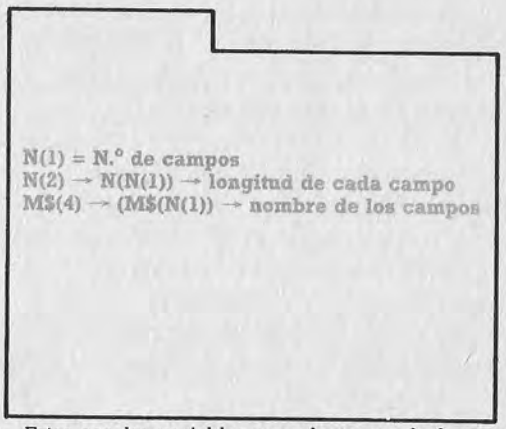


Fig. 2. Estas son las variables que almacenan la forma del fichero.

El funcionamiento del programa línea a línea es el siguiente:

Línea 9005. Borramos pantalla.

Línea 9006. Como hay dos formas de definir el fichero, creando una estructura nueva y utilizando una ya existente, imprimiremos un menú en la pantalla. Almacenamos en la variable alfanumérica A$ la cabecera de dicho menú.

Línea 9007. Asignamos a N el valor 2. Esta variable le dice a la rutina generadora de menús que este menú se compone de dos opciones.

Línea 9008. Asignamos al primer elemento del vector A$ el mensaje que compone la primera opción del menú.

Línea 9009. En el segundo elemento de dicho vector almacenamos la segunda opción del menú.

Línea 9010. Llamamos a la rutina generadora de menús.

Línea 9011. Si el resultado, después de que el usuario ha pulsado la opción que deseaba, es igual a uno (se utilizará la estructura de un fichero que ya existe), nos vamos a la línea 9039. Ya veremos más adelante qué es lo que hacemos a partir de esa línea.

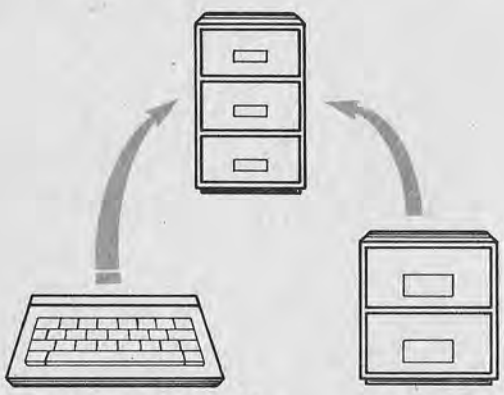


Fig. 3. Podemos definir nuestro nuevo fichero desde el teclado, o bien utilizando la estructura de otro fichero.

Línea 9012. En el caso de que el usuario quiera definir una nueva estructura, la pare del programa que lo realiza se encuentra a partir de esta línea. En ésta imprimimos un mensaje para que el usuario sepa en qué parte del programa se encuentra.

Línea 9013. Subrayamos el mensaje impreso anteriormente.

Línea 9014. Dejamos dos líneas en blanco.

Línea 9015. Se avisa al usuario para que responda las preguntas que a continuación le van a aparecer en pantalla.

Línea 9016. Se deja una separación con asteriscos. Debajo de dicha separación será donde se desarrolle la función principal de la rutina.

Línea 9017. Colocamos el cursor en la primera columna de la línea novena.

Línea 9018. Se le pregunta al usuario cuántos campos tendrá cada ficha del registro que estamos inicializando. El número de campos por registros está restringido a veinte como máximo, pero en muy pocas ocasiones se alcanzará este número.

Línea 9019. Se comprueba que el usuario no ha introducido ni un número negativo ni un número mayor de veinte. En caso de que lo haya hecho se le vuelve a preguntar.

Línea 9020. Se dimensiona un vector que tendrá tantos elementos como campos cada registro. Este vector MS() nos servirá para almacenar, más adelante, los nombres de cada campo.

Línea 9021. Dejamos una línea en blanco.

Línea 9022. Comenzamos un bucle dentro del cual vamos a preguntar la longitud en caracteres de cada campo y el nombre de dicho campo.

Línea 9023. Colocamos el cursor en la primera columna de la fila número 16.

Línea 9024. Imprimimos una serie de espacios en blanco. Estas dos líneas no sirven para nada en la primera vuelta del bucle, pero a partir de la segunda tienen como función la de borrar lo que haya escrito el usuario del programa en la pantalla.

Línea 9025. Colocamos el cursor en la primera columna de la fila 18.

Línea 9026. E imprimimos una serie de espacios en blanco. La función de estas dos líneas es idéntica a la de las dos anteriores.

Línea 9027. Volvemos a colocar el cursor en la línea 16.

Línea 9028. Y le preguntaremos al usuario la longitud del campo número Z en caracteres. Como Z va variando con el bucle, al usuario se le preguntará por un campo distinto cada vez.

Línea 9029. Se realiza un INPUT desde el teclado para recoger el número de caracteres que tendrá dicho campo. El resultado de la entrada se almacenará en el vector N$(), cuya declaración se encuentra en el programa principal.

Línea 9030. Volvemos a colocar el cursor en la línea 18.

Línea 9031. Y le pedimos al usuario que nos diga el nombre de este campo. El nombre del campo es el que aparecerá en la pantalla cuando se recojan los datos de una determinada ficha. Este nombre puede ser algo así como:

— AUTOR.
— NOMBRE DEL GRUPO MUSICAL.
— NUMERO DE TELEFONO.
— ETC.

Esto le dirá al usuario qué dato tiene que introducir en cada momento.

Línea 9032. Se realiza un INPUT desde el teclado y el resultado de la entrada se almacena en el vector M$(), del cual hemos hablado un poco más arriba.

Línea 9033. Aquí termina el bucle.

Línea 9034. Borramos la pantalla.

Línea 9035. Se le avisa al usuario de que la operación está terminada.

Línea 9036. Se asigna a la variable X el valor uno y a la variable Y el valor veinte. Estas dos variables serán las que le digan a la rutina de pulsa una tecla dónde tiene que colocar el mensaje.

Línea 9037. Llamamos a la rutina de pulsa una tecla.

Línea 9038. Una vez que el usuario ha pulsado cualquier tecla, se devuelve el control al programa principal.

La serie de líneas que aparecen a continuación son las encargadas de leer la estructura de un fichero que ya esté creado para crear otro nuevo.

Línea 9042. Borramos la pantalla.

Línea 9043. Imprimimos la ya usual cabecera para avisar al usuario de en qué parte del programa se encuentra.

Línea 9044. Subrayamos el mensaje.

Línea 9045. Dejamos una línea en blanco.

Línea 9046. Se le pregunta al usuario por el nombre del fichero del cual vamos a copiar la estructura.

Línea 9047. Dejamos tres líneas en blanco.

Línea 9048. Y recogemos del teclado, mediante una sentencia INPUT, lo que haya escrito el usuario en el ordenador. El resultado lo almacenamos en la variable alfanumérica N$.

Línea 9049. Dejamos dos líneas en blanco.

Línea 9050. Y le decimos al usuario que se asegure de que la cinta (o disco, según el ordenador) que ha preparado es la correcta. Este mensaje aparece porque el programa no está preparado para detectar errores.

Línea 9051. Se asignan los valores uno y veinte a las variables X e Y. Estas dos variables le dicen a la rutina de pulsa una tecla dónde tiene que imprimir el mensaje.

Línea 9052. Llamamos a la rutina de pulsar una tecla.

Línea 9053. Borramos la pantalla.

Línea 9054. Imprimimos un mensaje que le dice al usuario que el ordenador está leyendo el fichero que él introdujo como modelo de estructura.

Línea 9055. Abrimos el fichero. En el SPECTRUM, a la vez que abrimos el fichero, lo leemos.

Línea 9056. Leemos del fichero el número de campos que tiene cada registro.

Línea 9057. Dimensionamos la tabla de los nombres de los campos con el número de campos que acabamos de leer.

Línea 9058. Comienza un bucle desde uno hasta el número máximo de campos por registro dentro del cual vamos a leer la longitud de cada campo en caracteres.

Línea 9059. Leemos una a una la longitud de cada campo. El resultado de la lectura queda almacenada en el vector numérico N().

Línea 9060. Aquí termina el bucle.

Línea 9061. Asignamos al primer elemento del vector N() el número de campos por registro.

Línea 9062. Comenzamos otro bucle, dentro del cual vamos a leer del fichero el nombre de cada campo.

Línea 9063. Leemos mediante un input el nombre de cada campo y lo almacenamos en el vector M().

Línea 9064. Aquí termina el bucle.

Línea 9065. Borramos la pantalla.

Línea 9066. Sacamos un mensaje en la pantalla que le dice al usuario cuántos campos tiene cada ficha. Este mensaje y el siguiente está puesto por si el usuario se ha equivocado de fichero y desea leer la configuración de otro.

Línea 9067. Dejamos una línea en blanco.

Línea 9070. Comenzamos un bucle dentro del cal se le va a mostrar al usuario el nombre de todos los campos que componen un registro.

Línea 9071. Se imprime el número y el nombre del registro.

Línea 9072. Aquí termina el bucle.

Línea 9073. Se asignan valores a las variables X e Y para imprimir el mensaje:

PULSA UNA TECLA

Línea 9074. Llamamos a la rutina que imprime dicho mensaje.

Línea 9075. Y retornamos al programa principal.

Grabación de memorias

E las memorias descritas en el tomo anterior hay que destacar por su interés práctico aquellas que pueden ser grabadas y mantener su contenido indefinidamente, aun después de retirada la alimentación. Son las memorias llamadas comúnmente de sólo lectura o ROM (Read Only Memory). Dentro de este tipo de memorias las que pueden ser grabadas por el usuario son las más interesantes, pues permite realizar cambios en el sistema, para adaptarlo a nuestras necesidades.

Vamos a describir el proceso de grabación de los diferentes tipos de ROM grabables por el usuario y mostraremos varios diseños adaptables a nuestros ordenadores personales. Como uno de los tipos de memoria son borrables mediante rayos ultravioleta, indicaremos el modo de realizar un borrador que permita poner la memoria en las condiciones iniciales, si es necesario repetir el proceso.

Memorias de fusible o PROMS

Para determinadas aplicaciones que requieren tiempos de acceso muy bajos las memorias de sólo lectura de fusibles resultan apropiadas. Permiten realizar algunos tipos de funciones simples y de decodificación, sustituidas actualmente en muchos casos por circuitos PAL que incluyen, además, funciones complejas. Sin embargo, es intersante conocer el proceso de grabación y un ejemplo de la circuitería necesaria.

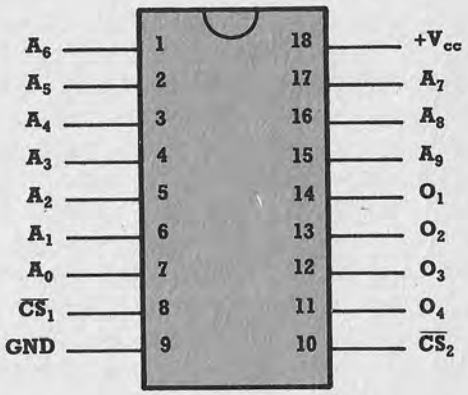


Fig. 1. Memoria PROM de fusibles PB506.

La grabación se efectúa mediante pulsos de corriente. Comienza verificando el nivel de tensión que se obtiene para una corriente que se inyecta en cada bit. Las entradas de decodificación del circuito deberán estar desactivadas, es decir, a nivel 1. La intensidad que se inyecta es de 20 miliamperios, a través del pin de salida. Si la tensión es menor que el nivel de referencia de 7 voltios, la celda está en nivel 1 y es programable. Se inyecta entonces una corriente de 200 miliamperios y se mantiene durante 7,5 microsegundos. Después se vuelve a verificar el nivel de tensión para una intensidad de 20 miliamperios. El ciclo se repite un número variable de ciclos hasta que la tensión de salida sobrepasa el nivel de referencia en la verificación, que corresponderá a que el fusible ha quedado prácticamente roto. En ese caso el nivel de tensión subirá hasta el límite dado por el generador de corriente. Después de la verificación se recomienda dar otros cuatro pulsos de grabación.

Fig. 2. Señales de grabación.

El circuito de grabación es de cierta dificultad, pues requiere la lectura de niveles analógicos después de cada pulso de grabación. Los pulsos de grabación, sin embargo, pueden generarse fácilmente mediante una fuente de intensidad constante, a partir de una fuente estabilizada y de un transistor, pues es sabido que un transistor se comporta como fuente de intensidad constante, proporcional a la intensidad de base. Por supuesto, que en vacío o con una resistencia de carga muy alta la corriente estará limitada por la tensión del generador, no comportándose realmente como fuente de intensidad constante.

La memoria es utilizada como cualquiera de las de sólo lectura, activando las entradas de dirección y las de selección del chip. El tiempo de acceso es de 70 nanosegundos máximo.

Memorias borrables con rayos ultravioleta o EPROMS

Las más comunes entre las memorias de sólo lectura grabables por el usuario son las de borrado por rayos ultravioleta. Tienen la particularidad de almacenar la información en una celda aislada y que actúa sobre la puerta de un transistor MOS. El proceso de grabado consiste en la introducción de cargas eléctricas en la celdilla mediante pulsos de tensión, suficientemente alta para el proceso, pero sin llegar a dañar el resto de la circuitería. Para activar solamente una celda es necesario direccionarla de la misma manera que se hace para leerla, pero con las entradas de control en condición de escritura y la entrada de grabación al nivel correspondiente al tipo a grabar. Por tanto, la secuencia necesaria para la grabación será:

— Establecer los niveles de reposo.
— Poner en las entradas de dirección los bits correspondientes a la dirección a grabar.
— Poner la entrada de selección en el nivel para grabar.
— Poner la entrada de grabación en el nivel activo DURANTE EL TIEMPO ESPECIFICADO.
— Verificar la grabación activando la señal de control de salida y realizando un ciclo de lectura.
— Desactivar la entrada de selección.
— Desactivar el nivel de grabación.

Necesitamos, pues, para el grabador los siguientes elementos:

— Señales para poder programar la dirección deseada de la memoria.
— Señales de control para producir los ciclos de grabación, lectura y verificación.
— Tensiones programables para las entradas de grabación.
— Módulos de personalización para los diferentes tipos de EPROM, pues las patillas de cada tipo de memoria presentan algunas diferencias.
— Conexión al ordenador personal.

El diagrama de bloques necesario es el de la figura 3:

Vamos a describir por separado la realización de cada bloque y más adelante mostraremos el circuito completo.

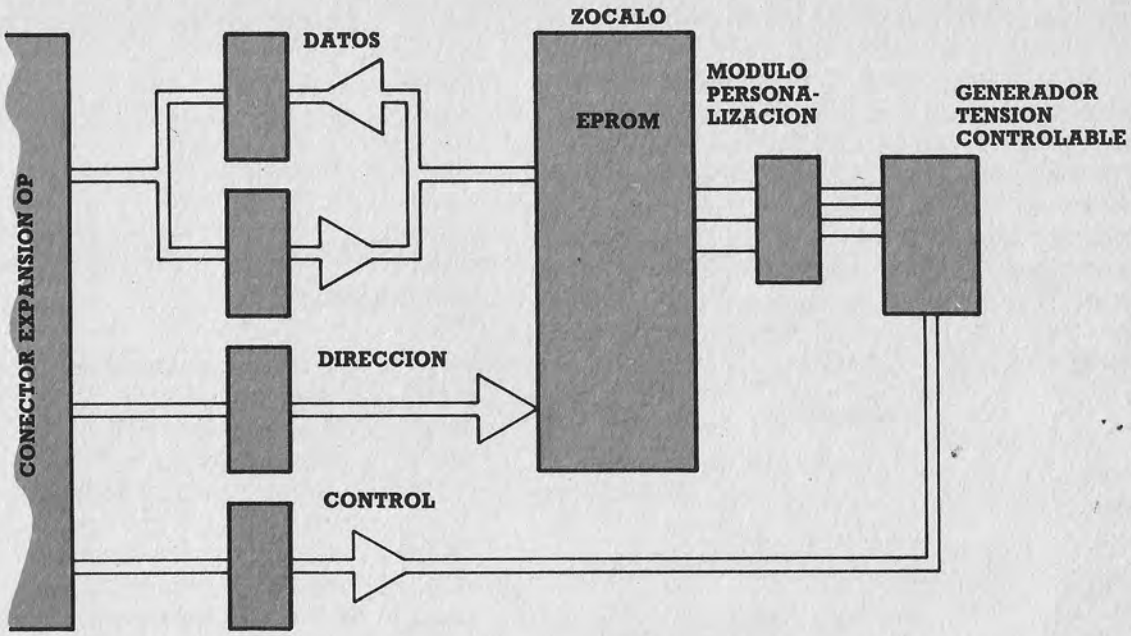


Fig. 3. Diagrama de bloques del grabador de EPROMS.

Señales para programación de la dirección

Se necesita poder direccionar hasta 64K, es decir, 16 bits. Una solución posible sería mediante puertos de salida conectados al bus, pero considerando que lo normal es realizar la lectura y la grabación de forma secuencial, se puede realizar el mismo trabajo generando la dirección mediante un contador que se inicializa a 0, al principio de la operación y se hace avanzar mediante pulsos para conseguir la dirección deseada. El número de chips necesario es el mismo en los dos casos, con la diferencia que solamente es necesario direccionar un puerto.

El circuito puede realizarse mediante dos chips de contadores de 8 etapas cada uno. El contador utilizado es de tipo asíncrono y se consiguen los 16 bits encadenando la salida del primero con la entrada del segundo. Las entradas necesarias son el reloj y la puesta a cero. Estas entradas estarán conectadas a sendas salidas de un puerto.

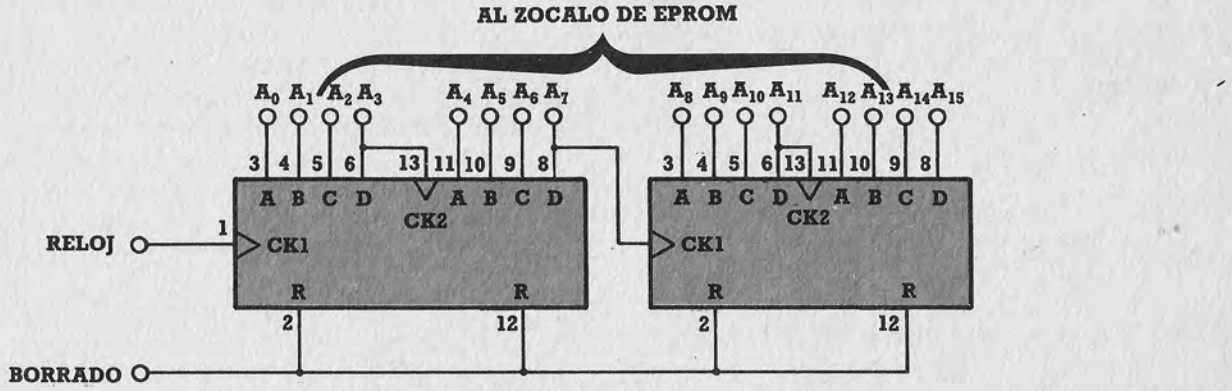


Fig. 4. Contador de direcciones.

Decodificación de dirección de la tarjeta

Utilizando la tarjeta de ampliación de puertos de entrada/salida tenemos ya direccionados lo puertos. Si hacemos una tarjeta específica para grabación deberemos incluir la decodificación necesaria para un puerto de

entrada y dos de salida. Se muestra el circuito de decodificación para Spectrum, con la denominación que ya hemos empleado en la descripción de la tarjeta de ampliación de puertos de entrada/salida. Pueden utilizarse los mismos esquemas que para dicha tarjeta para ubicar los puertos del grabador en la zona que más convenga. Es recomendable que esta tarjeta sea autónoma, pues realmente no es necesario utilizarla simultáneamente con las tarjetas de otros proyectos, pero es, en definitiva, cada uno quien decide cómo la utiliza.

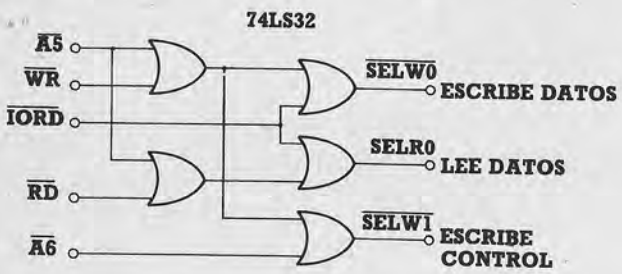


Fig. 5. Decodificación de dirección.

Conexión al bus de datos

La combinación de un puerto de cada tipo se utiliza como bus bidireccional para conexión al bus de datos de la EPROM. Se empleará un circuito de 8 bits de salida con biestables y 8 bits de entrada con amplificadores de tres estados. La dirección es la misma para los dos, con la diferencia que uno se activa con el pulso de escritura y el otro con el pulso de lectura.

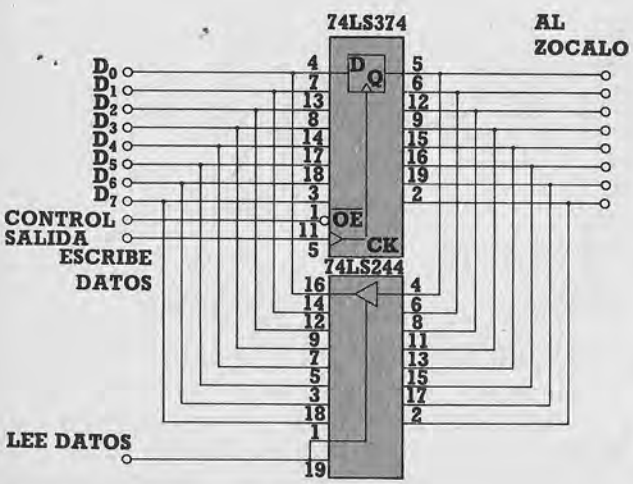


Fig. 6. Conexión al bus de datos.

La salida de los biestables hacia el bus de datos de la EPROM es controlable desde programa, pues las líneas son bidireccionales y ha de poderse dejar en tercer estado.

Generación de tensiones de grabación

Para la grabación son necesarias varias tensiones según el tipo de componente. En la entrada de grabación (Vpp) es necesario poder programar las tensiones 0, 12, 21 y 25 voltios. Podemos generar una tensión superior a la alimentación del equipo mediante un conversor DC-DC, que incluye un oscilador y después con un regulador controlable seleccionar la tensión adecuada. Para cada tipo de EPROM solamente es necesario conmutar entre 0 voltios y la tensión de grabación. La forma apropiada para realizar la selección es mediante puentes, que, además, es necesario tener en cuenta para poder llevar las tensiones a las patillas adecuadas a cada modelo. La solución es lenta pero segura y por lo demás no presenta mucho inconveniente el tener que sustituir el módulo de personalización para cada modelo.

Como la corriente consumida por los circuitos lógicos es alta, se recomienda utilizar una fuente separada que permita independizar la generación de impulsos de los circuitos del ordenador personal. Disponiendo de señal alterna para la generación de 5 voltios, podemos generar también la tensión de grabación mediante multiplicadores de tensión. La corriente de grabación es muy baja, del orden de 30 miliamperios, por lo que resultará sencilla la generación de la tensión necesaria.

El regulador que estabiliza la tensión de salida a partir de la tensión filtrada es de tipo LM317, que presenta la particularidad de poder programarse la tensión mediante el valor de la resistencia a tierra desde el terminal común. La tensión de salida resulta ser:

$$Vo = 1{,}25\ (1 + R2/R1) + Iadj\ R2.$$

Donde R1 es la resistencia entre las patillas de salida y común y R2 la resistencia desde la patilla común y masa. Por ser la intensidad Iadj, que sale por el terminal común, inferior a 100 microamperios, puede despreciarse el segundo término.

Para poder seleccionar las tensiones que en los diferentes casos es necesario dis-

poner se conectan en serie con las resistencias de programación transistores controlados por programa, que se hacen conducir o mantener al corte mediante la salida de un puerto. Este sencillo procedimiento puede emplearse con transistores corrientes siempre que las intensidades sean bajas y no se necesite mucha exactitud en las tensiones programadas. Si se necesitara programar valores muy exactos deberíamos utilizar transistores FET o MOS, que presentan una relación de resistencia abierto/cerrado muy grande e independiente de la tensión aplicada. En cualquier caso, es recomendable ajustar los valores de las tensiones, mediante la puesta en paralelo con la resistencia a masa de una del valor necesario para conseguir la tensión justa.

Para los ordenadores personales que disponen de suficiente corriente en el conector de expansión se puede prescindir de la generación de la tensión de alimentación de 5 voltios, pero sigue siendo necesario generar la tensión de grabación. Mediante la tensión de 12 voltios y un oscilador obtenemos tensión alterna, que podemos duplicar para conseguir la tensión necesaria. La regulación se hará igualmente mediante las resistencias conmutadas con transistores.

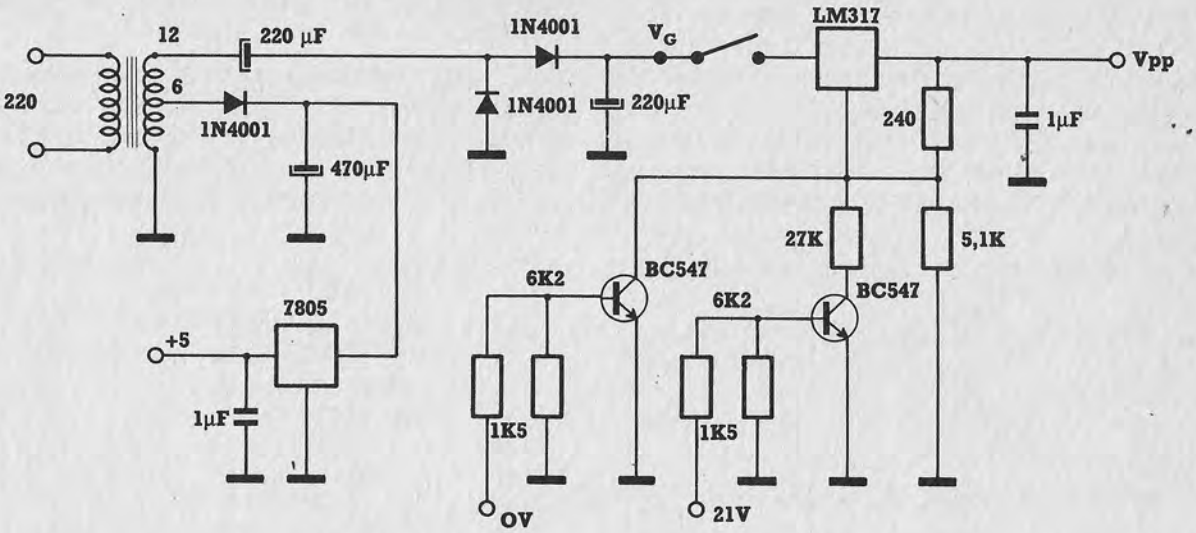


Fig. 7. Tensión de grabación con doblador de tensión.

Módulos de personalización

Para poder programar diferentes tipos de EPROMS es necesario dotar al equipo de grabación de la posibilidad de asignar las señales apropiadas a cada tipo. Montaremos un zócalo con las señales que pueden variar en cada tipo y prepararemos un soporte especial de componentes con los puentes necesarios para que se correspondan con las patillas. Si

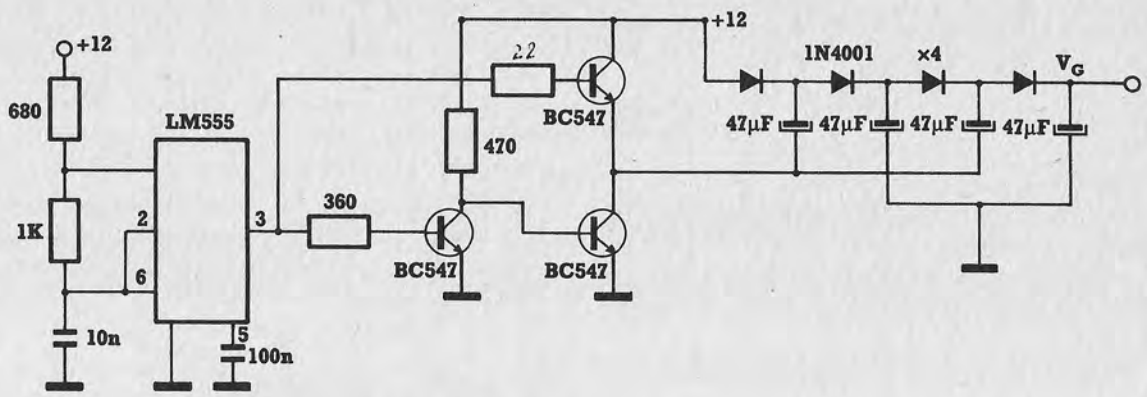


Fig. 8. Tensión de grabación con oscilador.

solamente deseamos grabar un tipo de EPROM podríamos, por supuesto, cablear la tarjeta para ese determinado circuito.

Las señales que resultan diferentes para las EPROMS son:

• Vcc. Es siempre 5 voltios, pero entrando por la patilla 28 o 26, según que la EPROM sea de más de 64K. o menos.

• Vpp. Es la tensión de programación, que puede ser necesario conmutar con valores de 0, 12, 21 y 25 voltios. Para controlar la salida (-OE), podríamos necesitar la tensión de 5 voltios, pero solamente es necesario para las 2732 y 27512, en cuyo caso puede hacerse la lectura como si de verificación se tratase, es decir, bajo la tensión de programación.

Estas señales se programarán desde el puerto de control, escribiendo cada vez el octeto completo, modificando el bit que se necesite.

Los pines de la izquierda significan la señal lógica representada y el bit del puerto de control o del contador de direcciones que la produce. La señal A12 no es necesario pasarla por el zócalo de personalización por ser utilizada en la misma patilla en todas la EPROMS de 64K. en adelante.

Los módulos de personalización se montarán sobre soporte de componentes y se etiquetarán de acuerdo con el tipo de EPROM para el que se deben utilizar, teniendo especial cuidado en marcar la orientación como

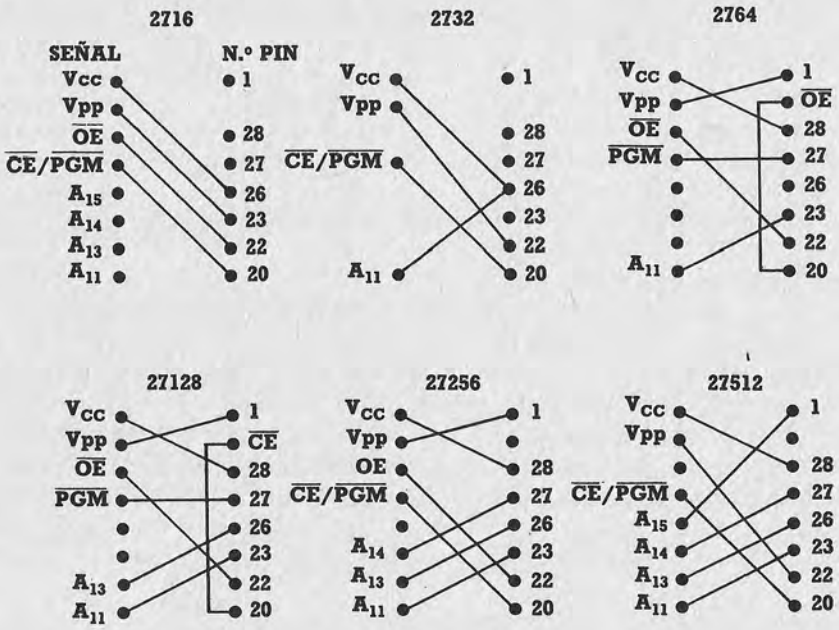


Fig 9. Módulos de personalización.

deben colocarse, pues una inversión del orden de pines sería catastrófica tanto para la EPROM como seguramente para el grabador y el ordenador.

• -OE. Control del amplificador de tres estados de la salida.

• -CE/-PGM. Control de selección de la EPROM y pulsos de programación de nivel lógico.

• A15. Solamente necesario para la 27512.

• A14. Necesario para 27512 y 27256.

• A13. Necesario para 27512, 27256 y 27128.

• A11. Necesario de la 2732 en adelante.

• -CE. Necesario por separado de -PGM en 2764 y 27128.

Señales de control

Las señales necesarias para producir el ciclo de grabación o lectura se generan por programa desde un puerto de salida. Todas las señales tienen unas especificaciones muy holgadas, excepto el pulso de grabación, que tiene un mínimo, por debajo del cual no se ga-

rantiza la permanencia de lo grabado y un máximo, por encima del cual pueden producirse daños irreversibles en la celdilla.

El pulso de grabación podemos realizarlo por programa, necesitando entonces recurrir a una rutina en lenguaje de máquina, o disponer de un oscilador monoestable con la duración ajustable y que sea activado por programa.

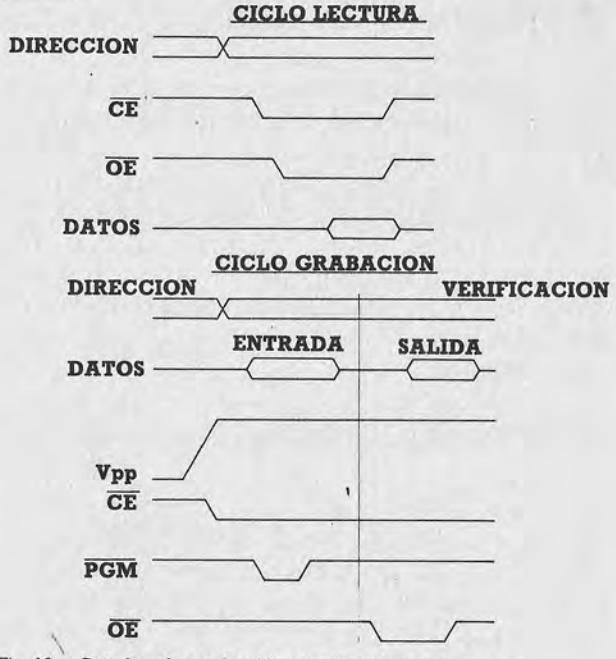


Fig 10. Señales de grabación, verificación y lectura.

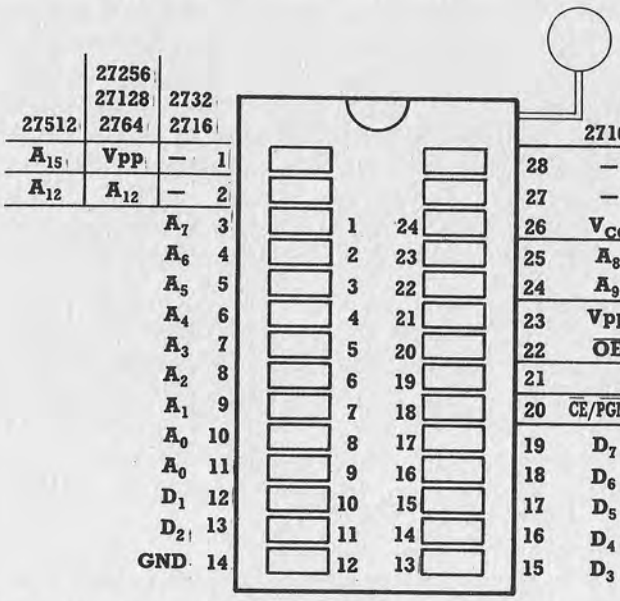
denominados de "fuerza de inserción nula", que permiten la colocación de la EPROM sin dañar las patillas y dejarla sujeta mediante una palanca. Estos zócalos son un poco caros, pero recomendables si se va a grabar con frecuencia. Para poder grabar las EPROMS 2764 y superiores es necesario que el zócalo tenga 28 patillas. Para la grabación de 2716 y 2732 deberemos tener especial cuidado en colocarlas ocupando las posiciones desplazadas, dejando las patillas 1, 2, 27 y 28 del zócalo sin utilizar.

En la figura 11 se muestra la asignación de pines de las distintas EPROMS, detallando especialmente las que son diferentes. En el exterior del zócalo se indica el número de pin, para la numeración correspondiente a la configuración de 28 patillas, mientras que en el interior se indica para la configuración de 24 patillas.

Puede verse que hay solamente 8 patillas cuya asignación de función difiere entre los diferentes tipos. Es necesario tener en cuenta también las tensiones de programación, que se ajustarán mediante otro zócalo de personalización.

Control de grabación

Para aumentar la seguridad de que no se "regraba" una EPROM que ya lo esté y que



	2716	2732	2764	27128	27256	27512
28	—	—	V'_{CC}	V_{CC}	V_{CC}	V_{CC}
27	—	—	$\overline{PGM}$	$\overline{PGM}$	A_{14}	A_{14}
26	V_{CC}	V_{CC}	—	A_{13}	A_{13}	A_{13}
25	A_8					
24	A_9					
23	Vpp	A_{11}	A_{11}	A_{11}	A_{11}	A_{11}
22	$\overline{OE}$	$\overline{OE}$/Vpp	$\overline{OE}$	$\overline{OE}$	$\overline{OE}$	$\overline{OE}$/Vpp
21						
20	$\overline{CE}$/PGM	$\overline{CE}$/$\overline{PGM}$	$\overline{CE}$	$\overline{CE}$	$\overline{CE}$/$\overline{PGM}$	$\overline{CE}$/$\overline{PGM}$
19	D_7					
18	D_6					
17	D_5					
16	D_4					
15	D_3					

Fig. 11. Zócalo para la EPROM.

Zócalo para la EPROM

Para mantener la EPROM que se graba es conveniente disponer de un zócalo de los deseamos copiar, es conveniente disponer un conmutador que impida la generación de tensiones de grabación si solamente se desea leer o verificar. Además, colocaremos un dio-

do LED que se ilumine cuando la tensión de grabación esté presente en el zócalo, para recordarnos que no debemos manipular indebidamente la EPROM.

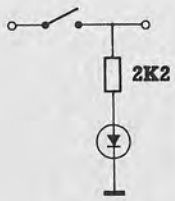


Fig 12. Control de grabación.

Registro de control

Para el funcionamiento de la tarjeta hay que disponer de varias señales de control, como ya se ha indicado al describir cada bloque del grabador. Todas las señales pueden generarse mediante un puerto de salida. El registro se escribe cada vez con la información necesaria para activar cada acción. El programa deberá tener en cuenta el estado en que se encuentra cada bit para no alterarlo indebidamente.

DENOMINACION	ACCION
OE B	CONTROL SALIDA A EPROM
Vpp = 0	PARA PERMITIR GENERA 0 VOLTIOS
Vpp = 21/25	PARA GENERAR TENSION DE GRABACION
PGM	PULSOS LOGICOS DE GRABACION
OE	PARA LECTURA DE LA EPROM
CE	SELECCION DE LA EPROM
BORRADO	PONE CONTADORES A 0
RELOJ	AVANZA DIRECCION

Fig. 13. Denominación de las señales de control.

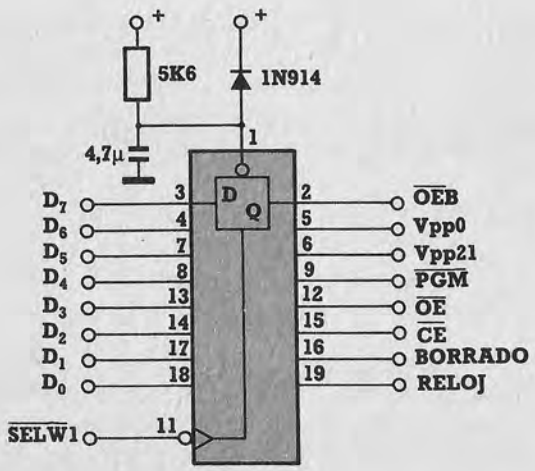


Fig. 14. Registro de control.

El registro de control se inicializa a cero mediante un circuito temporizador formado por un condensador y una resistencia. El diodo colocado en paralelo con la resistencia, acelera la descarga del condensador al apagar el equipo. Se emplea precisamente el circuito 74LS273 por disponer de la entrada de reset.

Programa de grabación

La circuitería descrita por sí sola no es capaz de ayudarnos a grabar las memorias EPROM, es necesario un programa que dote de capacidad lógica al circuito montado. Por ser un programa largo y con muchas posibilidades de adaptación a diversas aplicaciones se describirá detalladamente en el próximo tomo. También se incluirá un esquema completo englobando todos los bloques descritos en estas páginas.

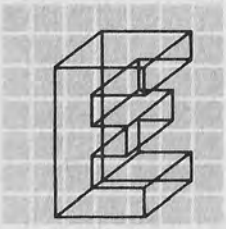
Memorias borrables eléctricamente

La complejidad del proceso de grabación y borrado de las EPROMS ha forzado la disponibilidad de circuitos que puedan ser grabados de forma más simple y que puedan ser alterados sin necesidad de sacarlos del circuito. Las memorias borrables eléctricamente (EEPROM = Electrically Erasable PROM) han venido a solucionar este problema.

Su comportamiento en lectura es exactamente igual al de las memorias EPROM. Para la grabación y el borrado hay que distinguir dos tipos: las que utilizan tensiones de 5 voltios únicamente o las que requieren tensiones mayores. Las primeras poseen, además, la circuitería interna necesaria para mantener el pulso de grabación el tiempo necesario.

NATURALEZA Y TECNOLOGIA

Estudio del rozamiento como fenómeno físico



EMPEZAREMOS por considerar los bloques de la figura 1. La polea se considera sin masa.

Los bloques A y B interaccionan a través de la tensión de la cuerda. Se trata, por tanto, de una fuerza interna al sistema. Pero el rozamiento del bloque A con el plano es una fuerza externa, además, de los pesos. Por consiguiente, ya no es constante la velocidad del centro de gravedad.

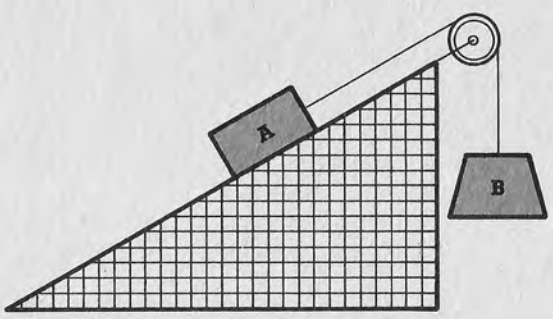


Fig. 1. Se simulará el movimiento de dos bloques unidos por una cuerda que pasa por una polea.

El movimiento de este sistema dependerá de:

— Las masas A y B, m_A y m_B;

— El coeficiente dinámico de rozamiento, K;

— El ángulo del plano, &.

La fuerza de rozamiento entre el bloque A y el plano depende de los materiales. La tabla 1 da algunos coeficientes de rozamiento. El rozamiento siempre se opone al movimiento del bloque, y vale $m_A \times g \times k \times \cos$ &.

Planteando las ecuaciones de Newton para cada bloque, o bien una ecuación global, podemos llegar a la siguiente sistematización:

Coeficientes de Rozamiento	
Madera sobre piedra	0,4
Madera sobre madera	0,5-0,2
Madera sobre metal	0,6-0,2
Hierro sobre piedra	0,7-0,3
Metal sobre metal (seco)	0,25-0,15
Superficies metálicas engrasadas	0,2-0,1
Corcho sobre asfalto	0,7

Condición	Movimiento de B	Aceleración
$m_B > m_A (\text{sen } \alpha + k \cos \alpha)$	Baja	$\frac{g(m_B - m_A \text{ sen } \alpha - m_A k \cos \alpha)}{m_A + m_B}$
$m_A(\text{sen } \alpha - k \cos \alpha) < m_B < m_A(\text{sen } \alpha + k \cos \alpha)$	No hay movimiento	0
$m_B > m_A (\text{sen } \alpha - k \cos \alpha)$	Sube	$\frac{g(m_A \text{ sen } \alpha - m_B - m_A k \cos \alpha)}{m_A + m_B}$

Tomaremos g = 9,81 en el Sistema Internacional como la aceleración de la gravedad. A la vista del cuadro anterior, *a priori* no podemos decir hacia dónde se moverá el sistema. Pero basta introducir los datos deseados y el ordenador trabaja para nosotros.

Una vez introducidos los cuatro parámetros de los que depende el sistema, se visualiza en pantalla la aceleración y el movimiento resultante de los dos bloques prismáticos.

Tenemos un primer ejemplo en la figura 2, con los datos en unidades del Sistema Internacional. En este caso el bloque B sube. Pero en la figura 3 la masa de B es suficiente para vencer el rozamiento y el peso tangencial de A:B baja y A sube, acelerados a 3,55 m./s.^{2}.

¿Cómo se saca un rendimiento óptimo al programa? Dejando invariables tres datos y variando el restante, para ver su influencia. Es lo que se ha hecho en la figura 4. Si siguiéramos aumentando el coeficiente k, llegaría un momento en que el rozamiento impide el movimiento, cosa de la que avisa el programa.

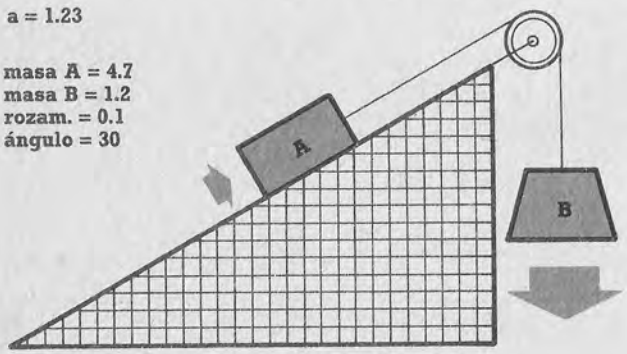


Fig. 2. El bloque B sube acelerado.

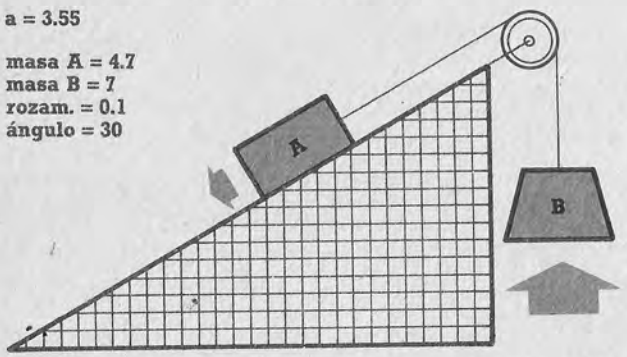


Fig. 3. Al ser el bloque B suficientemente pesado, en esta figura baja acelerado.

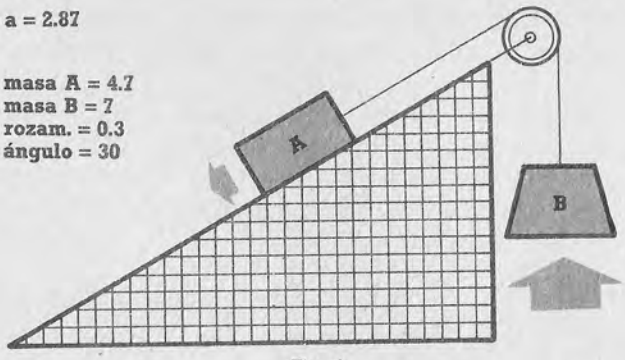


Fig. 4.

A este respecto se nos presenta una pequeña incongruencia: el coeficiente k es dinámico y, sin embargo, es posible que no haya movimiento. Se elimina tal incongruencia si tenemos en cuenta que el coeficiente estático, o sea, sin movimiento, es mayor que el dinámico. Por tanto, si la fuerza de rozamiento es capaz de impedir el movimiento con un coeficiente pequeño, tanto más con uno grande.

Como caso particular tenemos la figura 5, con un ángulo de plano igual a 0 grados. En este caso, el bloque prismático A se dibuja como un paralelepípedo, pudiendo solaparse el dibujo al principio del movimiento. Si en este caso haces 0, el coeficiente de rozamiento, se da el sorprendente hecho de que B, por ligero que sea, es capaz de arrastrar a A, por pesado que sea. Es lo que predice la teoría, en ausencia de rozamiento.

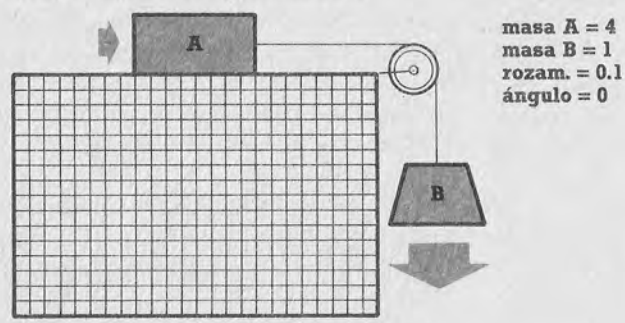


Fig. 5.

```
1  REM***********
2  REM**SPECTRUM**
3  REM***********
4  REM************************
5  REM**ROZAMIENTO EN PLANOS**
6  REM************************
10 INPUT "angulo del plano (0-60)";A
20 IF A<0 OR A>60 THEN GOTO 10
30 BORDER 6 :CLS
40 LET B=A*PI/180:LET C=TAN B:GOSUB 400
50 LET K=120:LET P=14:LET S=1
57 REM ******************
58 REM   bucle principal
59 REM ******************
```

```
60 FOR T=0 TO 60:IF K<1 OR K>140 THEN END
70 PLOT 150-K*COS B,140-K*SIN B:GOSUB 500
80 PLOT 159+15*COS B,140+15*SIB B:DRAW 0,-P
90 DRAW 3,0:DRAW 8,-8:DRAW -22,0:DRAW 8,8:DRAW 3,0
100 IF T=0 THEN GOSUB 600
110 IF T=1 THEN GOSUB 300
115 IF T=1 AND S=-1 THEN PRINT AT 7,17;"A"; AT 20,24;"B"
120 PRINT AT 0,10;"a=";INT (G*100)/100
125 LET H=G
130 IF H>2.1 THEN LET H=H-1:GOTO 130
140 LET L=H*T*T/2:LET K=K-L*S:LET P=P+L*S
150 PAUSE 40:NEXT T:STOP
297 REM ******************
298 REM impresion de variables
299 REM ********************
300 PRINT AT 3,25;"masa b=";AT 8,25;N AT 11,25;"rozam.="; AT 12,25;R; AT 15,25;"
angulo=";AT  16,25;A
310 PAUSE 60:RETURN
400 PLOT 150,140:DRAW 0,-140
410 FOR X=150 TO 0 STEP -1
420 LET Y=C*(X-150)+140:IF Y<0 THEN GOTO 440
430 PLOT X,Y:NEXT  X
440 PLOT 150,140:DRAW 15*COS B,15*SIN B
450 CIRCLE 150+15*COS B,140+15*SIN B,9:RETURN
497 REM ************************************
498 REM dibujo prisma
499 REM ************************************
500 DRAW -10*SIN B,INT (10*COS B)
510 IF A=0 THEN DRAW 0,-1
520 IF T=0 THEN DRAW (K+15)*COS B,(K+15)*SIN B:DRAW -(K+15)*COS B,-(K+15)*SIN B
525 IF T=1 AND S=-1 THEN DRAW 27*COS B,INT(27*SIN B):DRAW -27*COS B,-INT(27*SIN
B)
530 IF S=-1 THEN DRAW (1)*COS B,(1)*SIN B:DRAW -(1)*COS B,-(1)*SIN B
540 IF A=0 THEN DRAW 0,1
550 DRAW -10,0
560 DRAW 0,A/15-10*(1+TAN B*(1+SIN B)):RETURN
597 REM ***************************
598 REM   entrada de datos
599 REM ***************************
600 PAUSE 20:PRINT AT 5+K*SIN B/8,20-K*COS B/8;"a";AT 6,24;"b"
602 IF T=1 THEN RETURN
605 INPUT "masa A(Kg) ";M
610 INPUT "masa B(Kg) ";N
615 INPUT "coef.rozam.";R
625 IF N>M*SIN B+M*R*COS B THEN LET G=9.810001*(N-M*SIN B-M*R*COS B)/(M+N):LET S
=1:RETURN
630 IF N<M*SIN B-M*R*COS B THEN LET G=9.810001*(M*SIN B-N-M*R*COS B)/(M+N):LET S
=-1:LET K=12:LET P=135:CLS:GOSUB 400:RETURN
640 PRINT AT 10,13;FLASH 1;"el rozamiento impide el movimiento":END
700 REM ************************************
710 REM **VARIACIONES PARA I.B.M.-AMSTRAD**
715 REM ************************************
720 REM   sustituir plot por pset , draw por line - step ,border por screen 1,pa
use por un bucle vacio (for t=1 to 10000:next i),print at por locate fila,column
a
730 REM en las funciones trigonometricas los parametros deben ir entre parentesi
s,el flash por color 16,1,0
740 REM las variaciones de graficos en commodore se hacen segun mapa de memoria
con peek y poke
```

Comentario del programa

Líneas 10 y 20, introducción del ángulo (máximo de 60 grados).

Línea 30, pone la pantalla en modo gráfico.

En línea 40, conversión de grados a radianes.

Líneas 57 a 150, dibujo del plano, polea, etcétera.

Líneas 297 a 310, impresión de variables en pantalla.

Líneas 400 a 450, dibujo del círculo.

Líneas 497 a 560, dibujo del prisma.

Líneas 597 a 640, introducción de datos.

MATEMATICAS

Raíces reales de una ecuación f(x) = 0 por bipartición

Si una función continua en un intervalo toma valores de distinto signo en sus extremos, la función se anula en algún punto inter-

medio (teorema de Bolzano). Vamos a hallar por bipartición del intervalo ese punto (al menos uno) en que se anula.

En general, lo que se calcula es un valor aproximado, con un error menor que un número E fijado previamente.

Supuesto que se verifica la condición expuesta en el intervalo (X1, X2), el procedimiento es sencillo.

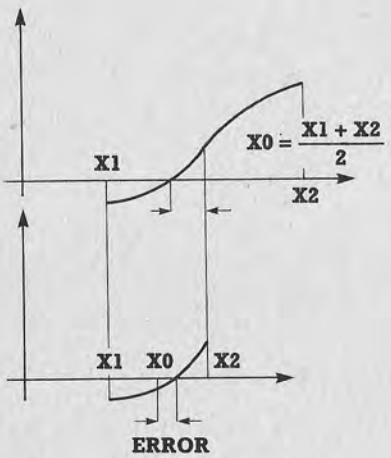


Fig. 6.

En la figura se representa una función Y = F(X),siendo F(X1) negativo y F(X2) positivo. Tomamos el punto medio del intervalo (X1, X2) y vemos el valor de la función en ese punto F(X0). Si es cero, hemos encontrado una raíz y el problema termina. Si F(X0) no es cero, entonces, de entre los dos segmentos en que se ha dividido el intervalo, elegimos aquél en que la función toma valores de signo contrario en sus extremos. Así sucesivamente, hasta que la longitud del intervalo sea menor que 2E. Tomando ahora X0 como valor aproximado de la raíz, cometemos un error menor que E.

No es seguro que en algunas de las sucesivas biparticiones podamos dar exactamente con la raíz :f[(X1 + X2)/2] = 0. Lo que sí es seguro es que cada vez nos acercamos más a ella; hasta tener la raíz acotada en un intervalo [X1-X2] muy pequeño. Tomando entonces su punto intermedio como raíz, cometemos un error menor que (X1-X2)/2.

Apoyándonos en estas ideas haremos un programa que nos permita hallar raíces para una ecuación dada. Una de las múltiples soluciones podría ser:

```
1 REM**I.B.M-AMSTRAD-MSX**
2 REM**********************
5  REM *********************************
10  REM calcular raices por biparticion
15 REM *********************************
20 CLS
30 FOR I=1 TO 6
40 PRINT :NEXT I
50 PRINT"pulse LIST 130 y ponga la funcion"
70 PRINT"pulse RETURN y a continuacion RUN 90"
80 STOP
90 CLS
100 INPUT "introduzca extremos del intervalo";X1,X2
110 INPUT "error minimo.....";E
120 FOR I=1 TO 10:PRINT:NEXT I
130 DEF FNY(X)= 3X+2
140 LET Y1=FNY(X1):LET Y2=FNY(X2)
150 IF FNY(X1)=0 THEN PRINT "hay una raiz en";X1:GOTO 250
160 IF  FNY(X2)=0 THEN PRINT"hay una raiz en";X2:GOTO 250
170 GOSUB 300
180 LET X0=(X1+X2)/2
190 LET Y0=FNY(X0)
200 IF Y0=0 THEN PRINT"hay una raiz en";X0:GOTO 250
210 IF ABS(X2-X1)<2*E THEN PRINT X0;"es raiz con error menor que";E:GOTO 250
220 IF SGN(FNY(X1))=SGN(Y0) THEN GOTO 240
230 LET X2=X0:GOTO 180
240 LET X1=X0:GOTO 180
250 END
300 REM Subrutina para controlar el cambio de signo
310 IF SGN(Y1)<>SGN(Y2) THEN GOTO 400
320 LET L=X2-X1
330 FOR I=X1 TO 1000
340 LET C=X1+L*RND
350 IF SGN(Y1)<>SGN(FNY(C)) THEN GOTO 390
360 NEXT I
370 PRINT"no se encuentra cambio de signo"
380 PRINT"es posible que no haya raices":GOTO 250
390 LET X2=C
400 RETURN
```

Comentario del programa

El programa está dividido en las siguientes partes:

1. Presentación, instrucciones y entrada de datos (líneas 10-110).

2. Se comprueba si alguno de los extremos es la raíz de la ecuación. Líneas 150-160.

En caso de que ninguno de los extremos sea raíz:

3. Estudio del cambio de signo (subrutina 300).

Si los extremos cumplen la condición del teorema de Bolzano, nos salimos de la subrutina y el programa continúa. En caso contrario, hay que buscar un punto C en el interior del intervalo [X1, X2] tal que el intervalo [X1, C] sea válido.

Para ello, se podría dividir [X1, X2] en un número elevado de partes, 100 ó 1.000 y ver los valores de la función en los puntos intermedios. Pero este procedimiento es menos rentable que obtener números aleatorios comprendidos entre X1 y X2 .En efecto, en la figura, la función cambia de signo entre las divisiones 850-950. Habría que comprobar como mínimo los 850 primeros números para observar un cambio de signo. En cambio, la probabilidad de obtener un número aleatorio en ese intervalo es 1/10.

4. Por último, si hemos obtenido el intervalo adecuado, se busca la raíz en ese intervalo (líneas 180-250).

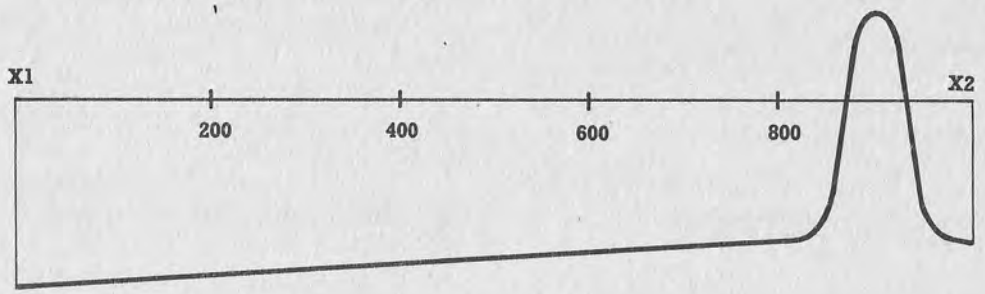


Fig. 7.

SOCIEDAD

Evolución de la población

La población de una ciudad ha evolucionado de la siguiente forma:

1935	1238400	1960	2362057
1940	1401017	1965	2716366
1945	1580880	1970	3123821
1950	1786055	1975	3592394
1955	2053963	1980	4237525

El objeto de nuestro programa es representar, mediante un diagrama de barras, esta evolución.

El programa sería el siguiente:

```
10 REM ****************************
20 REM *EVOLUCION DE LA POBLACION*
30 REM ****************************
32 CLS
35 PRINT "REPRESENTACION DE LA EVOLUCION"
40 PRINT "DE LA POBLACION DE UNA CIUDAD"
50 PRINT "MEDIANTE DIAGRAMA DE BARRAS"
60 PRINT:PRINT
70 PRINT "PARA VER EL DIAGRAMA PULSE"
80 PRINT "RETURN"
90 INPUT A$
110 CLS
120 LET M=0
130 FOR I=1 TO 10
140 READ  A(I),P(I)
150 IF P(I)>M THEN LET M=P(I)
160 NEXT I
170 REM **REPRESENTACION DEL DIAGRAMA**
180 FOR I=1 TO 10
190 PRINT:PRINT:PRINT A(I);
200 LET C(I)=INT(P(I)*34/M+.5)
210 FOR J=1 TO  C(I)
220 PRINT"*";
230 NEXT J
240 NEXT I
250 DATA 1935,1238400,1940,1401107
260 DATA 1945,1580880,1950,1786055
270 DATA 1955,2053963,1960,2362057
280 DATA 1965,2716366,1970,3123821
290 DATA 1975,3592394,1980,4237525
300 END
```

Comentario del programa

El año y la población están recogidos en las variables con subíndice A(I) y P(I), respectivamente, donde 1 < = I < = 10.

Para elegir la escala, asociamos al valor máximo, que designamos por M, las 34 casillas; luego a un número N de habitantes le hacemos corresponder:

N*34/M

Como geralmente saldrá un número fraccionario es necesario redondear las unidades mediante la sentencia:

S = INT(N*34/M + 0.5)

El número de casillas asociado a la población P(I) se guardará en la variable C(I). En el ciclo 130-160 leemos los años y las poblaciones correspondientes, y a la vez determinamos el valor máximo de la población, mediante la instrucción de la línea 150, valor que guardamos en la variable M.

En la línea 190 se imprime el año A(I) y en la 200 se hace el cambio de escala.

A continuación imprimimos la barra correspondiente a la poblacion I con las instrucciones de las lineas 210-230.

PARA LOS MAS JOVENES

A la caza del submarino (juego)

En un casillero de 8 × 8 el ordenador sitúa al azar un submarino, fuera del alcance de nuestra vista. Vamos lanzando cargas de profundidad en diversas casillas. Tras cada carga lanzada, el ordenador contesta con la distancia a la que se encuentra el submarino, medida en casillas horizontales y verticales. Si la distancia es cero, contesta "HUNDIDO" y muestra la posición del submarino (*).

En el programa propuesto hemos establecido una subrutina (300-399) para dibujar el cuadro de batalla. Para ello, hemos introducido en tres variables de cadena los siguientes motivos:

M1$ = "12345678"
M2$="**********" (10 asteriscos)
M3$="* *" (8 espacios con un asterisco a cada lado).

La impresión, en líneas sucesivas, de los motivos M1$, M2$, M3$ (ocho veces) y M2$, poniendo delante de cada M3$ el número correspondiente, produce el recuadro de la figura.

Con la función TAB se ajustan los motivos y se centra el dibujo.

En las líneas 110, 120 y 130 se cargan en la variable de índice D$(I) los caracteres *, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, que van a marcar la distancia D a la que se encuentra el submarino. Como para dicha marca sólo disponemos de un espacio, las distancias 10, 11, 12, 13 y 14 son representadas por las letras A, B, C, D, y E.

Comienza el juego a partir de la línea 140, que remite a la subrutina 300 de representación gráfica.

En la línea 150 el ordenador elige al azar las coordenadas del submarino.

Las líneas 160, 170 y 180 sirven para situar el cursor en la zona inferior de la pantalla, y preguntar allí las coordinadas de la zona del disparo.

Desde la 190 a la 220 se calcula la distancia D, y se imprime en la casilla correspondiente. En la 230, si la distancia no es cero, se repite el disparo, y si es cero se imprime "HUNDIDO", borrando previamente con espacios la línea en la que hemos estado poniendo las coordenadas. Y el juego finaliza.

```
1  REM**I.B.M.-AMSTRAD-MSX**
2  REM**********************
10 REM *A LA CAZA DEL SUBMARINO *
15 REM **************************
20 GOSUB 300
30 PRINT"ESTO ES LA CAZA DEL SUBMARINO"
40 PRINT"SE LANZA UNA CARGA DE PROFUNDIDAD"
50 PRINT"ESCRIBIENDO EL NUMERO DE LINEA Y EL DE COLUMNA SEPARADOS POR COMA"
60 PRINT:PRINT"EN LA ZONA CORRESPONDIENTE APARECE"
70 PRINT"LA DISTANCIA A LA QUE  SE HALLA EL"
80 PRINT"SUBMARINO MEDIDA EN CUADRADITOS HORIZONTALES Y VERTICALES"
90 PRINT:PRINT"PARA EMPEZAR EL JUEGO PULSA RETURN"
100 INPUT A$
110 DIM D$(14)
120 FOR I=0 TO 14:READ D$(I):NEXT I
```

```
130 DATA *,1,2,3,4,5,6,7,8,9,A,B,C,D,E
140 GOSUB 300
145 RANDOMIZE
147 LOCATE 12,1:PRINT"
150 LET X=INT(8*RND(8)+1):Y=INT(8*RND(8)+1)
160 LOCATE 1,1
170 FOR I=1 TO 20:PRINT:NEXT I
180  INPUT "LINEA,COLUMNA";L,C
190 LET D=ABS(X-L)+ABS(Y-C)
200 LOCATE 1,1
210 FOR I=1 TO L+1:PRINT:NEXT
220 PRINT TAB(C+12);D$(D)
230 IF D<>0 THEN LOCATE 21,1:PRINT"                                    ":GOTO 160
240 LOCATE 1,1
250 FOR I=1 TO 20:PRINT:NEXT I
260 PRINT"                          HUNDIDO"
270 END
300 REM***SUBRUTINA DE REPRSENTACON GRAFICA"
310 M1$="12345678"
320 M2$="**********"
330 M3$="*        *"
340 CLS
350 PRINT TAB(13);M1$
360 PRINT TAB(12);M2$
370 FOR I=1 TO 8
380 PRINT TAB(9);I;TAB(12);M3$
390 NEXT I
395 PRINT TAB(12);M2$
399 RETURN
```

Nota

Modificaciones a los programas para Spectrum y Commodore

SPECTRUM

Donde figure::

"LOCATE X, Y" sustituir por "PRINT AT X, Y" siendo X, Y los números de fila y columna respectivos.

"END" sustituir por "GOTO 9999".

COMMODORE

Donde figure:

"CLS" sustituir por "PRINT CHR$(143)".

AMSTRAD

Donde figure:

"LOCATE X, Y sustituir por "LOCATE Y, X"

PEQUEÑA HISTORIA DE LA INFORMATICA

El valle del Silicio (II)

IEMPRE que se piense en el valle del Silicio debe recordarse a Frederick Terman. El fue, si no su creador, al menos su impulsor más convencido.

Se preguntará el lector en qué colaboró este científico, que fue, desde luego, un magnífico profesor de Universidad, pero que no destacó de forma estelar como muchos otros, ni evidentemente obtuvo el Premio Nobel en ninguna disciplina.

La principal cualidad de Terman fue probablemente su magnífica entrega a la docencia. Se puede ser un buen profesor, y enseñar con claridad a los alumnos, pero también, por el contrario, se puede ser otro tipo de profesor vocacional. En este tipo de profesores, su misión no consiste únicamente en enseñar bien, sino en formar a los alumnos, ejercer un cierto papel de padre protector y canalizador de esfuerzos. Y todo esto debe hacerse sin esperar una recompensa especial, porque no se trata de un trabajo, sino que es una vocación (léase diversión). Eso fue Terman. Y a esas cualidades vocacionales se unió un espíritu pragmático que no concebía separaciones entre los estudios, la preparación para un trabajo y la realización de dicho trabajo. Para él, la Universidad no podía estar desligada de la empresa. Una y otra se compenetraban. La investigación en la Universidad debía cubrir necesidades y objetivos empresariales, y a su vez, los empresarios debían nutrirse del enorme potencial humano de la Universidad, aprovechando los hallazgos realizados por los investigadores.

A nuestro entender, ese fue el logro de Terman: La creación de una amalgama empresa-Universidad, que junto con una inquebrantable voluntad le hacía ganar las batallas, porque los Malos Hados llegaban a aburrirse de ponerle cortapisas. El valle de Santa Clara se convirtió en el valle del Silicio debido a muchas pequeñas circunstancias favorables, pero la principal probablemente sea la indomable voluntad, vitalidad y falta de desánimo de Frederick Terman.

Este valle ha acumulado entre su hermosa vegetación la mayor concentración de sabiduría humana jamás conseguida. En una hipotética línea que uniera Los Angeles, San Francisco y San Diego, se encuentra el 60 % de los científicos de peso a nivel mundial. Dos de cada tres premios Nobel viven en la zona. Sus Universidades (Stanford, Berkeley y Caltecch) se encuentran entre las más prestigiosas del mundo. La renta "per cápita" es la más alta de los Estados Unidos, y la población es, en su mayor parte, bastante joven (la mitad tiene menos de cuarenta años). La creatividad potencial de sus habitantes hace que puedan ofrecer sus ideas a una empresa, o financiador, pudiendo hacerse ricos en menos de cinco años si la idea es suficientemente original. En suma, es el valle de la creatividad.

Naturalmente, Terman no ha sido el artífice único de todo lo obtenido, pero sí le podemos atribuir, sin por ello exagerar, que ha sido el creador del embrión.

Terman nace en 1900 en un pequeño pueblo de Indiana, y es educado por su padre, un eminente psicólogo, siguiendo unas teorías de educación natural de Rousseau. Y, fiel a una de ellas, no fue a la escuela hasta que tuvo diez años. Su padre era un investigador bastante innovador (a él se deben los tests de inteli-

gencia Stanford-Binet). El Terman adolescente vivió en un ambiente universitario libre y algo provinciano. La familia se había trasladado a Stanford, donde su padre era profesor, y fue en esta Universidad donde Frederick decidió estudiar Ciencias Químicas. Como cualquier muchacho cuya familia dispusiera de medios económicos suficientes, Terman marchó a seguir estudios al Este, al MIT (Massachusetts Institute of Technology). Allí (por el momento todo le iba bien) fue alumno predilecto de uno de los científicos más considerados del momento, Vannevar Bush, que le ofreció un puesto de profesor en el propio Instituto. Sin embargo, la vida de Terman iba a complicarse enormemente. En 1924 vuelve a la casa de sus padres para pasar el verano, y a partir de ese momento su salud va a sufrir tales quebrantos que casi estuvo a punto de morir. Sólo su enorme fuerza de voluntad, o testarudez, lo sacaron de la situación. Primero tuvo tuberculosis, y, además, en un grado avanzado, estando muy afectado. Por si no era suficiente, estuvo a punto de morir de peritonitis y tuvo problemas en los ojos. Todas estas calamidades le obligaron a pasar más de un año en cama. El tiempo que tuvo que estar postrado no fue en absoluto desperdiciado por el joven químico. Lo aprovechó para leer y experimentar en radio, tema de vanguardia en aquellos años. Al final de su enfermedad, su padre le consiguió que le ofrecieran unas clases de Ingeniería de la Electricidad en Stanford. Eran dos horas de clase diarias, y dada su situación física, Terman debía permanecer el resto del día en la cama.

Para el año 1937 ya era director del Departamento de Ingeniería de la Electricidad de Stanford, y estaba totalmente integrado en su Universidad. Su trabajo le apasionaba y colaboraba y experimentaba con sus alumnos en el laboratorio, e incluso en su casa, donde tenía montado un pequeño taller. Ya hemos hablado de sus relaciones con Hewlett y Packard (dos de sus mejores alumnos). Terman comenzó a sufrir entonces a su alrededor el fenómeno de la fuga de cerebros. Todo muchacho que quisiera "hacer carrera" debía marcharse al Este, donde estaban las grandes empresas (Bell, etc.), que ofrecían más oportunidades de futuro. Terman era un entusiasta de su valle, de la Universidad, del calor y naturalidad de sus habitantes, e instaba a sus alumnos a ayudar a crear un clima tal que se fueran creando puestos de trabajo en la propia zona. Para lograrlo, él creía en una interrelación continua, un flujo bidirección de información y colaboración entre empresa y Universidad.

¿Sabía usted que...

¿Sabía usted que en San Francisco el Consulado de la URSS tiene cerca de doscientos diplomáticos, casi tantos como la propia Embajada en Washington?

La actividad de muchos de ellos es simplemente espionaje industrial. Es posible que haya prosoviéticos infiltrados en las compañías, pero no se trata de llegar a esos extremos. Para la mayoría, su trabajo consiste en suscribirse a revistas, frecuentar librerías técnicas, apuntarse a seminarios y otros actos académicos y, en general, entrar en contacto con nuevos investigadores pertenecientes al equipo de algún investigador importante. Si llegan a tratar con alguna frecuencia con un investigador interesante, le ofrecerán revistas o libros, y también algunos artículos sobre sus trabajos. Es el momento de preguntar por las investigaciones del otro, que amablemente accederá (naturalmente, si su naturaleza no es de absoluto secreto).

Sólo con las revistas y libros que aparecen ya se pueden elaborar *dossiers* interesantes. Como las innovaciones en electrónica e informática son tan rápidas, ciertas informaciones novedosas ya no lo son tanto para los fabricantes y, por tanto, constituyen material publicable.

A raíz de la Segunda Guerra Mundial, su antiguo profesor del MIT (Massachusetts Institute of Technology), reclutado por el gobierno de Roosevelt para tareas de investigación de la Defensa, le llama para dirigir un laboratorio de investigación de radio, también ligado a la Defensa. En un año Terman organizó a unos 800 jóvenes que trabajaban en absoluto secreto. Las investigaciones sobre radar fueron desde luego muy rentables, y salvaron muchas vidas humanas.

Pasada la guerra, Terman vuelve a Stanford con muchas ideas. Ha colaborado con otros científicos del Este y ha aprendido cómo se canalizan los recursos obtenidos para investigación en otras universidades, entre ellas Harvard. La idea es patrocinar pequeños trabajos de investigación, en los que participen uno o dos profesores y varios alumnos becados, frente a los grandes proyectos, que suponen a veces un despilfarro económico. Ter-

man vuelve a Stanford a poner en práctica todo lo aprendido; sacando dinero de donde no hay (los recursos de esta Universidad son cortos frente a otras que han colaborado más efectivamente en la defensa del país).

Otra vez afincado en Stanford, Terman fue el mejor "relaciones públicas" de la zona. Estaba muy relacionado con los empresarios que se iban asentando (muchos de ellos eran sus alumnos) y además intentaba ayudarles en sus problemas. (Por ejemplo, tomó la decisión, criticada por muchos, de alquilar parte de los terrenos pertenecientes a la Universidad a algunas empresas que comenzaban entonces a trabajar, entre ellas Hewlett Packard. Todas estas relaciones crearon una enorme confianza y, por tanto, muchos intercambios entre las empresas y la Universidad. Los donativos económicos que ofrecían las firmas (a cambio de exenciones fiscales) iban subiendo en progresión geométrica año tras año, y además con las rentas de las empresas instaladas podían cubrir los gastos de la escuela.

Berkeley es otra de las Universidades próxima al valle del Silicio. En ella trabajaba Ernest Lawrence creando las bases de lo que sería más tarde el ciclotrón. Este acelerador de electrones, que se utilizaría para bombardear y abrir por primera vez el núcleo de un átomo, le valió al profesor Lawrence (Ernest) el Premio Nobel en 1939. Las aspiraciones del equipo de Lawrence eran explorar la materia (desde luego no parece probable que se le ocurriera aprovechar la fisión nuclear con fines belicistas). Sin embargo, el profesor Oppenheimer, uno de los padres de la bomba atómica, perteneció al equipo de Lawrence.

(Olvidemos la bomba atómica y recordemos otra aplicación del ciclotrón más altruista y tonificante para el espíritu): John Lawrence, hermano de Ernest Lawrence, utilizó un sistema basado en el ciclotrón para eliminación de células cancerosas.

¿A qué se debió que California resurgiera tan espectacularmente después de la Segunda Guerra Mundial? Es muy posible que su renacer fuera causado precisamente por la guerra. No debemos olvidar el desastre de Pearl Harbor, que dio lugar a la intensa actividad bélica en el Pacífico. Como California era eminentemente una zona agrícola, no disponía apenas de industria de apoyo de armamento, y fue preciso crearla (naturalmente, mediante modernas instalaciones). Además, la industria aeronáutica también estaba asentada en la zona, probablemente porque el actual magnate de la aviación, Joe Douglas, comenzó su actividad pidiendo dinero a otro pionero de la región, el "Terman de Los Angeles", Joe Chandler. De este "dictador", Douglas obtuvo además del dinero una lista de otros posibles "donantes". Algo más al norte en la región se habían ido creando astilleros, que iban creciendo año tras año en el número de toneladas construidas. Las necesidades bélicas del país favorecieron muchísimo la economía de la zona.

En Stanford se reúnen ciento setenta de los más poderosos hombres de negocios de los Estados Unidos, y algunos de otros países. Se trata de la reunión de la "Business Roundtable", verdadero grupo de presión norteamericano.

¿Sabía usted que...

¿Sabía usted que las grandes compañías de Electrónica e Informática no sólo han sido creadas por grandes nombres, sino que han "producido" hombres notables?

Uno, especialmente conocido, al margen de opiniones sobre su persona, creó su popularidad y sedimentó su personalidad como hombre público en la gran General Electric. Nos referimos al presidente de los Estados Unidos, Ronald Reagan.

Todo el mundo sabe que el presidente Ronald Reagan fue actor en su juventud. Lo que no saben muchos es cómo se interesó por la política y cómo pasó a tomar parte activa en ella. Ronald Reagan lleva ya casi veinte años en puestos políticos de enorme responsabilidad. En 1967 juró su cargo como gobernador de California. Estaba recogiendo su fruto. La semilla la había echado a la tierra durante los ocho años en que trabajó para la General Electric. Sus cualidades de orador hicieron que esta firma lo contratara para recorrer las distintas factorías, como relaciones públicas. Su misión consistía en conocer los problemas de los empleados de las distintas factorías y, al mismo tiempo, hacerles llegar mensajes de la dirección. Sus cualidades de orador y las amistades de personajes influyentes que había hecho durante su etapa en el cine hicieron que fuera apoyado por muchos de ellos, avanzando hasta llegar a ser gobernador de California. Conservador y convencido de sus ideas sin fisura alguna, Reagan ofrecía la imagen de alguien firme que sabe lo que quiere. De nuevo esa imagen, sus dotes y sus amigos influyentes le han aupado hasta la posición más alta en su país.

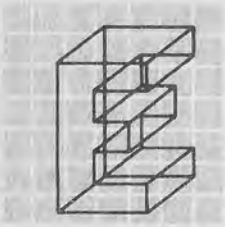
Esta reunión demuestra la importancia a nivel empresarial que va tomando California.

Pero pasemos a otra época más moderna, a una California rica y en expansión; donde conviven ejecutivos, hippies y gentes de todo tipo; y estudiemos a uno de ellos: Alan Shugart. Es otro de los hombres clave del valle. Actualmente, además de dirigir Amstrad, es el fundador de Seagate. Estudió Física e Ingeniería en Redlands (California), y una vez finalizados sus estudios comenzó a trabajar para IBM, en Santa Mónica, como encargado del servicio de mantenimiento de una gama de máquinas. De ahí pasó a San José, a un laboratorio de investigación y desarrollo, también de IBM. Fue en este laboratorio donde Alan se formó realmente. Estaban desarrollando la memoria RAM. Trabajó también en otros tipos de memoria, discos, etc. Su decisión de abandonar la compañía se debió a un traslado a Nueva York, donde no se encontraba en absoluto identificado. Como la zona que rodea San Francisco tiene una enorme densidad de empresas de electrónica e informática, le fue fácil encontrar trabajo en Memorex, como vicepresidente, encargándose del desarrollo de nuevos productos. En 1973 se decidió a crear su propia empresa junto con otros amigos, la Shugart Associates. El cometido de la empresa era fabricar drives de discos flexibles. Al cabo de dos años existieron discrepancias entre el equipo que se ocupaba de la financiación y el equipo técnico, y como resultado de todo ello, Alan abandonó la empresa. Estaba muy afectado y se dedicó a un trabajo-hobby que desde luego no parece muy ligado a la industria informática: la pesca del salmón. Pero realmente ese hobby era también un modo de vida, y Alan Shugart pasó estrecheces hasta que en 1979 uno de sus antiguos socios, que también abandonó la firma, le propuso formar otra compañía para fabricar dispositivos de almacenamiento de datos de mucha mayor capacidad, pero que fueran adaptables para miniordenadores. La idea era buena, ya que normalmente esas máquinas utilizan varios drives de discos flexibles. Y se pusieron en marcha. Fundaron la "Seagate" (Shugart no podía ni quería utilizar su propio nombre, porque la Shugart Associates seguía existiendo, y encontró que "Seagate" sonaba parecido). Realmente, la Seagate no está exactamente en el valle del Silicio, pero sí consideramos que Alan Shugart es un hombre típico de dicho valle. Más tarde fundó Amstrad, que estos días es uno de los ordenadores más populares del mercado.

En el valle viven muchos otros hombres dignos de ser mencionados. Algunos, como Alan Kay (vicepresidente de Atari), viven en Malibú y tienen su empresa a 700 kilómetros de distancia, Alan resulve los posibles problemas que se pueden presentar a través de terminales de datos, y con visitas semanales. Otros también han sido "fundadores" más de una vez, vendiendo sus antiguas empresas para comenzar a "dar cuerpo" a sus ideas en otra empresa nueva. Otros se han arruinado y han vuelto a surgir del desastre. Todos ellos constituyen un ejemplo vivo de lo que la voluntad y la preparación pueden obtener.

Seguiremos hablando del valle del Silicio; de los financieros que viven en él, de la empresa Apple, etc.

Programación orientada al objeto



ENTRE las numerosas innovaciones que se están produciendo en el campo de la Informática, una de las más prometedoras es la modificación conceptual de las técnicas de programación que se está realizando con la introducción de la programación orientada al objeto.

Esta modificación consiste en cambiar el esquema básico de enfoque de los problemas, de tal modo que se piense no en un conjunto de «procedimientos» (bloques de texto de lenguaje de programación que realiza ciertas funciones) que «manejan» o «procesan» datos (unos datos estáticos: variables o no). Con esta nueva concepción, el conjunto de datos (llamados objetos) son «dinámicos», es decir, realizan funciones sobre sí mismos, cuando se les envían ciertos «mensajes» (u órdenes de actuación).

Para realizar este tipo de programación se pueden utilizar los lenguajes convencionales (o algunos de ellos) ligeramente modificados (en ciertos casos, la modificación es mayor). Sin embargo, es más usual utilizar alguno de los lenguajes específicamente desarrollados a tal fin. Estos lenguajes orientados al objeto se suelen llamar no-procedurales, en contraposición a los lenguajes de programación convencionales que están dedicados a procedimientos, y que se llaman, por ello, lenguajes procedurales. Existen, además, algunos dialectos de lenguajes convencionales (Object Pascal, Objetive C, Object Assembler...), que incluyen con el lenguaje normal (procedural) herramientas suficientes para hacer una eficaz programación orientada al objeto (o al menos así lo definen sus autores). Por último, hay casos (como el NEON) en que, partiendo de un lenguaje convencional (FORTH en el caso ya citado de NEON), desarrollan un verdadero lenguaje orientado al objeto.

Muchos lenguajes de programación soportan este paradigma de proceso que llamamos «procedimiento». Los procedimientos activos actúan sobre los datos pasivos que le son pasados. Un ejemplo típico puede ser la función raíz cuadrada que toma un número y devuelve su raíz cuadrada.

En un lenguaje muy ligado al tipo de los datos, como el Pascal, es típico disponer de una versión diferente de raíz cuadrada para cada tipo de dato X (normalmente se obtiene un resultado con coma flotante). En un lenguaje de tipo «late binding» (la «ligazón» de procedimientos con datos se realiza «a posteriori») como el LISP, en el momento de la ejecución se detecta el tipo de dato de X, realizándose las operaciones necesarias para este tipo de dato. Esta clase de operaciones genéricas son normalmente primitivas restringidas a un pequeño tipo de datos, como números, o son funciones definidas en los términos de esas primitivas.

Los lenguajes orientados al objeto emplean, por el contrario, una aproximación a la programación tipo «centrada en el dato» o «centrada en el objeto». En lugar de pasar los datos a los procedimientos, es necesario pedir a los objetos (datos) que realicen las operaciones sobre ellos mismos. Para enfatizar esta diferencia, pondremos algunos ejemplos de un lenguaje genérico orientado al objeto de tipo Smalltalk. En este tipo de lenguaje el nombre del objeto suele ir seguido por: operación y, a continuación, seguido de cualesquiera argumentos; y va terminada (la instrucción corres-

pondiente) por un punto. Por ejemplo, una expresión para obtener la raíz cuadrada de x tendrá el siguiente aspecto:

x :sqrt.

Lo que significa es que a x se le pide que realice la operación :sqrt sobre ella misma. Se dice, pues, que x es el receptor del mensaje :sqrt.

Un ejemplo más complicado puede ser la operación producto escalar. El producto escalar de dos vectores unidimensionales x e y da como resultado un escalar:

s :dot y.

En este caso, se dice que x realiza una operación producto escalar sobre ella misma y el argumento y. Se puede obtener la raíz cuadrada del producto de x e y u y asignar el resultado a la variable z. Vemos cómo se realiza:

z←(x :dot y) :sprt.

Se utilizan paréntesis para indicar el orden de evaluación (aunque realmente no son necesarios en este caso, ya que se supone que se comienza a evaluar de izquierda a derecha).

Puede ser útil que, antes de continuar con estos conceptos, veamos cómo se articulan los elementos básicos usados en estos lenguajes. El objeto al que nos hemos estado refiriendo (por ejemplo x, en el caso de la raíz cuadrada) es una instancia de clase. La clase proporciona toda la información necesaria para construir y utilizar objetos de un tipo particular: sus instancias. (Por esta razón en ciertos casos a una clase se le denomina fábrica-factory). Cada instancia pertenece a una clase, y, a su vez, una clase puede tener instancias múltiples (muchas instancias).

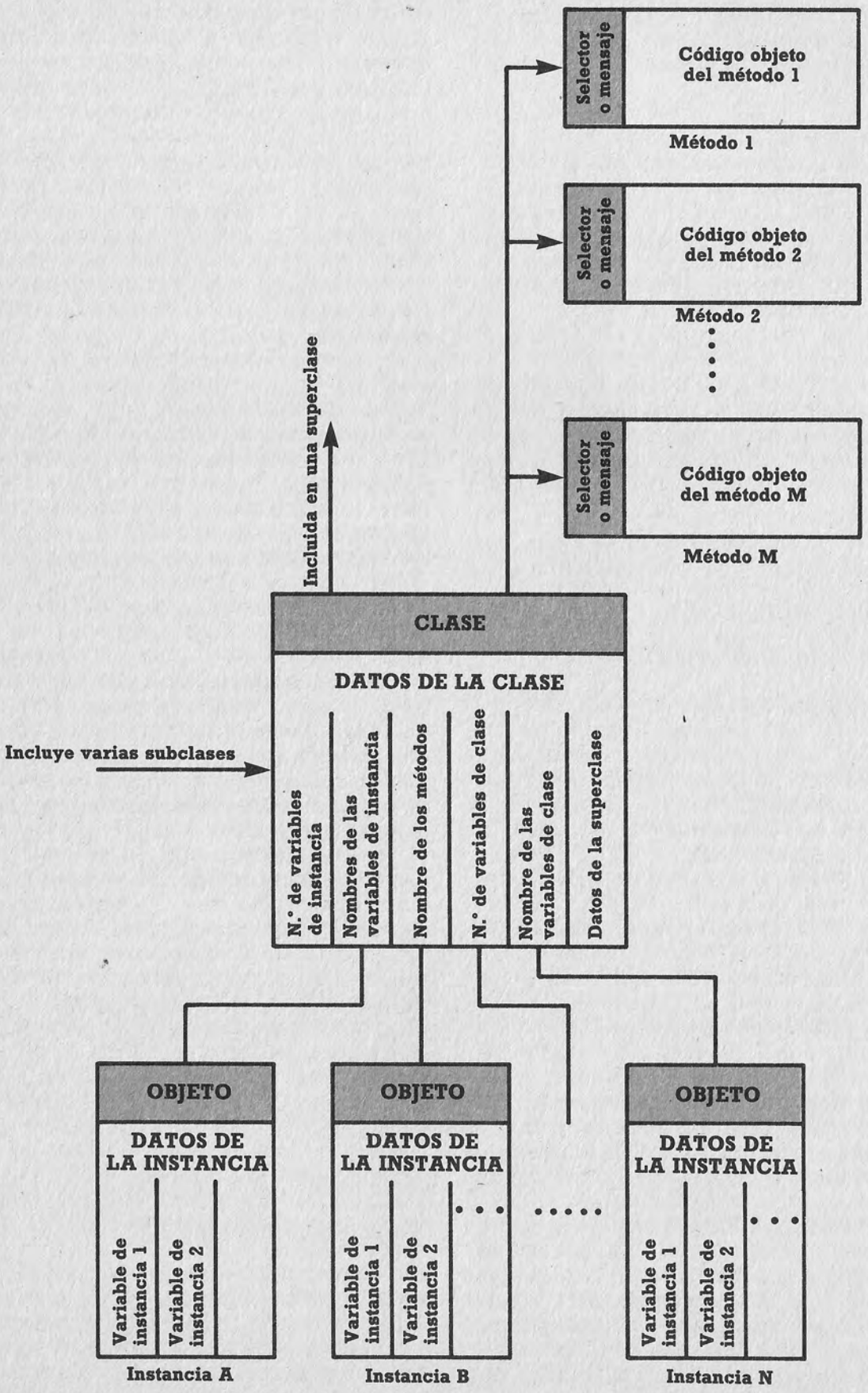
La clase también proporciona almacenamiento para los métodos. Los métodos son simplemente procedimientos a los que se llama enviando selectores a una de las instancias de la clase (a lo que también se llama enviar mensajes). Los métodos residen en la clase para ahorrar almacenamiento, ya que todas las instancias de una clase tienen un conjunto idéntico de métodos. Estos pueden alojar temporalmente variables para utilizarlas durante la ejecución del método. Esas variables temporales son muy semejantes a las variables locales de los procedimientos de Pascal, en que su valor se pierde en cuanto se abandona el procedimiento. Cada instancia tiene adjudicado un almacenamiento para mantener su estado individual. Este estado se referencia mediante variables de instancia. Las variables de instancia pueden ser datos tipo primitiva, como, por ejemplo, enteros, otros objetos o ambos, dependiendo del lenguaje. Cada objeto dispone de su propio conjunto de variables de instancia. Tanto las variables de instancia como las temporales pueden ser referenciadas libremente, dentro del alcance de un método de un objeto. Pero, a diferencia de las variables temporales, el valor de las variables de instancia no se pierde cuando se abandona el método del objeto.

Los procesos se realizan enviando mensajes a los objetos; cada mensaje invoca un método dentro de la clase del objeto. Normalmente los métodos (en su ejecución) envían mensajes a otros objetos, que a su vez invocan a otros métodos, etc., hasta alcanzar el punto en el que se invoca el método primitivo. En este punto termina la cadena de envíos de mensajes. Cada mensaje enviado puede devolver (normalmente devolverá) un resultado al emisor (por ejemplo, x :sqrt. devuelve la raíz cuadrada de x). El resultado final de todos esos envíos de mensajes es normalmente el cambio de estado de uno o más objetos. En algunos casos, sin embargo, el mensaje se envía simplemente para llamar a una primitiva, teniendo un efecto colateral externo con el mundo de los objetos, por ejemplo, accediendo a un sistema de ficheros externos o controlando el hardware.

Para soportar totalmente la programación orientada al objeto, los lenguajes de programación deben tener cuatro características: ocultación de la información, abstracción de los datos, ligaduras dinámicas y herencia. Examinemos estas características en detalle y su implementación en algún lenguaje procedural, para ilustrar por qué son necesarias las cuatro características y por qué los lenguajes orientados al procedimiento convencionales no pueden soportar la programación orientada al objeto.

Ocultación de la información

La ocultación de la información es importante para asegurar fiabilidad y capacidad de remodelación de los sistemas de software reduciendo las interdependencias entre los distintos componentes de este software. El es-


Selector o mensaje
Código objeto del método 1
Método 1
Selector o mensaje
Código objeto del método 2
Método 2
Selector o mensaje
Código objeto del método M
Método M
Incluida en una superclase
CLASE
DATOS DE LA CLASE
Incluye varias subclases
N.° de variables de instancia
Nombres de las variables de instancia
Nombre de los métodos
N.° de variables de clase
Nombre de las variables de clase
Datos de la superclase
OBJETO
DATOS DE LA INSTANCIA
Variable de instancia 1
Variable de instancia 2
Instancia A
OBJETO
DATOS DE LA INSTANCIA
Variable de instancia 1
Variable de instancia 2
Instancia B
OBJETO
DATOS DE LA INSTANCIA
Variable de instancia 1
Variable de instancia 2
Instancia N

tado de un módulo de software está contenido en las variables privadas, y sólo es visible desde el interior del rango de alcance del módulo. Los datos son manipulados únicamente por un conjunto limitado de procedimientos. Además, como el estado interno de las variables de un módulo no es modificable directamente desde el exterior del módulo, un interfaz de módulo diseñado con cuidado puede permitir cambiar las estructuras internas de los datos y procedimientos sin que ello afecte la implementación de otros módulos de software.

Los lenguajes más modernos, e incluso el FORTRAN, pueden soportar en cierto grado la ocultación de información. El ISO (standard) Pascal es una excepción a notar, ya que no permite la declaración de variables estáticas internas a un procedimiento.

Abstracción de datos

La abstracción de datos puede considerarse un medio de utilizar la ocultación de información. Un programador define un tipo de dato abstracto, consistente en una representación interna más un conjunto de procedimientos utilizados para acceder y manipular los datos. El Modula-2 proporciona mecanismos de abstracción de datos excelentes.

El mecanismo de abstracción de datos de Modula-2 proporciona cierto grado de protección, ya que no dispone de acceso directo al estado interno de las pilas (stacks). La pila se manipula a través del proceso del módulo, y procedimientos de interrogación. Existen, sin embargo, unos cuantos problemas con la solución del Modula-2. Uno es que los procedimientos utilizados por un módulo deben tener nombre 'únicos' en todo caso, cualificados. Por ejemplo, si un módulo utiliza (importa) dos tipos de datos abstractos diferentes, Stack-Pila y Queue-Cola, deben inicializarse variables de esos tipos, y los procedimientos de inicialización definidos para esos tipos deben tener nombres diferentes, como InitializeStack y InitializeQueue, o su utilización debe calificarse como en StackInitialize y QueueInitialize. Esto último hace que el programa resulte menos versátil.

El lenguaje Ada resuelve parcialmente ambos problemas, a través de dos características del lenguaje: un solapamiento de operadores y unas unidades genéricas de programa. El solapamiento de operadores permite que el programa utilice varios operadores con el mismo nombre. La distinción entre operadores se puede determinar en el momento de compilar examinando los tipos y el número de los parámetros, exactamente igual que cuando se utiliza para añadir enteros o números reales en la mayoría de los lenguajes modernos. Las unidades de programa genéricas permiten la definición de un módulo para utilizarlo con diferentes tipos de datos. La unidad de programa genérica es una plantilla que puede ser parametrizada con tipos actuales durante la compilación de los programas (utilizando sus capacidades).

Si usted lo que desea es utilizar la pila para almacenar elementos heterogéneos, surgirá otro problema. No será suficiente con la solución del momento de compilación, ni del solapamiento de operadores o las unidades de programa genéricas. La solución es la ligazón dinámica (dynamic binding).

GLOSARIO DE TERMINOS UTILIZADOS EN LOS LENGUAJES ORIENTADOS AL OBJETO

Abstracción de datos. La abstracción de datos es una de las cuatro cualidades básicas que debe tener un lenguaje para ser considerado útil para la programación orientada al objeto. Consiste en la cualidad que tiene el lenguaje mediante la cual el programador puede definir un tipo de datos abstracto independiente de los procedimientos con los cuales va a ser utilizado este tipo de datos.

Ataduras dinámicas. (Ver **Ligazón dinámica.**)

Centrado en el dato, en el objeto. Dícese del lenguaje de programación también llamado dedicado al objeto, en contraposición a los lenguajes dedicados al procedimiento.

Clase. Conjunto de objetos formado por todos los que tienen un determinado tipo. Cada uno de los objetos incluidos en una clase recibe el nombre de «instancia» de la clase. Las clases pueden ser divididas en subclases para un mejor tratamiento de los datos y adecuación a los procedimientos definidos sobre esa clase o subclase. Varias clases pueden a su vez ser englobadas en una superclase.

Fábrica. Otra denominación del concepto de clase en el que se hace alusión a la cualidad que tiene la clase de disponer de toda la información necesaria para construir y utilizar nuevos objetos o instancias de clase.

Factory. Nombre inglés correspondiente al término castellano fábrica.

Herencia. Una de las cuatro cualidades que hacen que un lenguaje sea considerado útil para la programación orientada al objeto. Mediante la herencia, la subclase de una clase hereda las variables de esa clase y los métodos definidos para dicha clase.

Instancia de clase. Nombre que se da a un objeto cuando se quiere subrayar su cualidad de elemento individual incluido en una clase.

Ligazón. Relación que se establece entre los datos (u objetos) y los procedimientos, funciones o métodos que a ellos se refieren o que los manipulan, referida básicamente al tipo de ese dato.

—, de última hora (late binding). Ligazón que se establece en el momento de la compilación del programa.

—, dinámica. Cuando no está establecida una ligazón permanente entre los datos u objetos y los procedimientos métodos o funciones que los manipulan, sino que se establece en el momento de ser ejecutados dichos métodos o funciones.

—, primitiva (early binding). Ligazón establecida de un modo permanente, como se realizan en la mayoría de los lenguajes procedurales.

Método. Procedimiento o funciones definidos para una clase de objetos.

Mensaje. Llamada a un objeto. Los mensajes, junto con los objetos, constituyen el núcleo central de un lenguaje de programación orientado al objeto.

Neón. Lenguaje de programación orientada al objeto, basado en el conocido lenguaje FORTH.

No procedural. Dícese de los lenguajes de tipo declarativo orientados al objeto, en con-

traposición de los orientados al procedimiento (que son lenguajes convencionales procedurales).

Object Pascal. Objetive C, etc., dialectos de los correspondientes lenguajes procedurales, preparados para su utilización en la programación orientada al objeto.

Objeto. Cada uno de los elementos individuales de información (dato) sobre los que operan los métodos, en un entorno de programación orientada al objeto. También llamado instancia de clase.

Ocultación de la información. Una de las cuatro cualidades que debe tener un lenguaje de programación para ser considerado útil en la programación orientada al objeto. El estado de un módulo de software está contenido en las variables privadas y sólo es visible desde el interior de dicho módulo, con lo cual la modificación de otros módulos externos no le afecta, y se puede mantener la independencia de los diferentes componentes de un programa.

Orientado al objeto, lenguaje. Lenguaje no procedural en el cual la información (dato) toma la forma de objetos organizados en clases y superclases.

Paradigma. Referido a los sistemas de representación del conocimiento, ejemplo o esquema básico al cuál se adecuan los datos.

Poliformismo dinámico. Cualidad de un lenguaje de programación consistente en que un mismo mensaje o función puede dar resultados diferentes dependiendo del receptor actual del mensaje.

Primitivas. Funciones básicas proporcionadas por el propio lenguaje.

Procedimiento. Bloque de instrucciones que realizan una determinada tarea.

Procedural. Dícese del lenguaje orientado al procedimiento, es decir, aquél en que la estructura general del programa viene dada por los diferentes procedimientos a aplicar a los datos.

Receptor del mensaje. Dato (objeto) que recibe un mensaje (u orden) y la ejecuta sobre sí mismo con el concurso del (o los) parámetro(s) que se le suministran en el propio mensaje.

Selectores. (Ver **Mensaje.**)

Smalltalk. Lenguaje de programación orientada al objeto. Con la definición del Smalltalk se comenzaron a desarrollar los lenguajes no procedurales orientados al objeto.

Solapamiento de operadores. Cualidad de un lenguaje de programación orientada al objeto mediante el cual es posible la utilización de varios operadores con el mismo nombre, quedando éstos identificados por otras cualidades adicionales, como el método en que han sido definidos, etc.

Subclase. Conjunto de objetos dentro de una clase que comparten unos métodos específicos. (Ver **Clase.**)

Superclase. Conjunto de varias clases para las que se puede definir algún método común.

Tipo de datos, lenguaje ligado al. Dícese de los lenguajes de programación en que la relación entre los procedimientos y los datos se realiza mediante el mecanismo de ligazón primitiva.

Variables de instancia. Variables definidas para cada instancia de clase u objeto dentro de una clase. Mediante las variables e instancias se mantiene el estado de referencia del objeto de que se trate.

Variables temporales o de método. Variables definidas en el interior de un método y que sólo conservan sus valores en el interior del método. Cuando éste se abandona, se pierde el valor de las variables temporales o de método.

Diádica, operación. Operación que maneja dos operandos.

Diádico, operador Booleano. Operador Booleano que tiene dos operandos. Los operadores diádicos Booleanos son AND, la equivalencia, la exclusión, el OR exclusivo la inclusión, NAND NOR y OR.

Dialectos. Variantes de que se dispone en las diferentes implementaciones de un lenguaje de programación.

Dicotómica, búsqueda. (Ver **Búsqueda dicotómica**).

Diferenciador. Dispositivo cuya función obtenida en la salida es la derivada con respecto a una o más variables. Red de resistencia-capacitancia que se utiliza para obtener los picos de una señal.

Diferencial, analizador. Un computador analógico que utiliza integradores interconectados para resolver ecuaciones diferenciales.

Diferencial, engranaje. En los computadores analógicos, un mecanismo que relaciona los ángulos de rotación de tres ejes, diseñado usualmente de tal modo que la suma algebraica de la rotación de dos de los ejes, es igual a dos veces la rotación del tercero. Este engranaje diferencial se suele utilizar para sumar y restar.

Digital. Dato establecido mediante dígitos, es decir, representado mediante valores discretos, no continuos. En contraposición a Analógico.

Digital, ordenador. Ordenador que opera con valores o datos discretos. En contraposición a Ordenador Analógico.

Digital/Diferencial, analizador. Ordenador en el que su unidad fundamental es un integrador digital, que opera de forma semejante a cualquier mecanismo integrador.

Digitalizar. Representar los datos mediante valores discretos a partir de una representación analógica, ρ continua.

Dígito. Símbolo que representa a uno de los enteros no negativos pertenecientes a una base de numeración. Por ejemplo, en notación decimal, un dígito es uno de los caracteres 0 a 9. También puede llamarse dígito a un carácter numérico

Dígito binario. (Véase **Binario, dígito.**)

Dígito de signo. Dígito mediante el cual se anota el signo del número.

Dígito significativo. Dígito utilizado con un fin concreto. Normalmente mediante el número de dígitos significativos se especifica el grado de exactitud alcanzado.

Dimensionar. Reservar espacio en la memoria central para las matrices de variables (variables que llevan subíndice o subíndices).

Dinámica, posición de almacenamiento. Técnica de almacenamiento en la que la posición que toman los programas y los datos está determinada por ciertos criterios aplicados en el momento en el que son necesarios.

Dinámica, programación. En investigación operativa, procedimiento de optimización de la solución de un problema multiuso, en el que se pueden tomar un cierto número de decisiones en cada etapa del proceso. En contraposición con programación convexa, entera, lineal, matemática, no lineal, cuadrática, etc.

Dinámico, almacenamiento. Dispositivo de almacenamiento de los datos que permite a éstos variar con el tiempo, por lo que no siempre pueden recuperarse. Los tambores magnéticos y los sistemas de disco son dispositivos de almacenamiento dinámico permanente. La línea de retraso acústico es un sistema de almacenamiento dinámico volátil.

Dirección. Número o nombre que identifica las posiciones individuales de la memoria. Puede estar contenida en una instrucción, e indica en qué posición de la memoria se encuentra el operando de que se trata.

Dirección constante. Dirección fija de carga de un programa.

Directo, acceso. En un proceso de obtención de datos de un sistema de almacenamiento, en el que el tiempo de acceso no depende en absoluto de la posición del último dato obtenido ni de la posición del dato. Sinónimo de acceso aleatorio.

Directo, dispositivo de acceso. Dispositivo cuyo tiempo de acceso a cualquier dato es independiente de la posición en la que se encuentre dicho dato.

Directo, subrutina de acceso. Subrutina que debe relocalizarse e introducirse en la rutina principal, cada vez que se utilice. También se llama subrutina abierta. En contraposición con subrutina cerrada.

Directorio. Indice en el que se encuentran todos los ficheros contenidos en esa memoria de masa. El directorio está también grabado en el mismo soporte de memoria (disco duro, diskettes, etc. que el fichero principal). Además de los nombres de todos los ficheros, en el directorio puede aparecer también otra información adicional, como longitud, fecha de creación, última actualización, etc.

Disco duro. Tipo de memoria externa cuyo aspecto físico es circular y rígido. Lleva una superficie recubierta por una sustancia magnética. En él se pueden almacenar los datos magnetizando selectivamente ciertas porciones. Está dividido en pistas y sectores. Tiene una capacidad de almacenamiento muy superior a los discos flexibles.

Disco flexible. Tipo de memoria consistente en una superficie de vinilo recubierta de una sustancia magnética. Los tamaños normales son de 8 pulgadas y de 5 y 1/4 pulgadas y de 3 1/2 pulgadas.

Disco óptico. Sistema de almacenamiento de disco que no utiliza técnicas magnéticas. El sistema de grabación de los datos utiliza rayos láser. Tiene una capacidad de almacenamiento muy superior a los discos convencionales. Hasta el momento, sin embargo, se utiliza en memorias ROM, sólo lectura, pero esta técnica está avanzando mucho.

Discreto. En contraposición a continuo. Se refiere a algo que toma valores en escalones fijos, y no valores intermedios. En otras palabras, sólo puede tomar un número finito de valores.

Diseño asistido por ordenador. (Véase **CAD.**).

Disk Operating System. (Véase **Sistema Operativo de Disco,** y **DOS.**).

Display. Presentación visual de los datos.

Disquete. (Véase **Disco flexible.**)

DLE. El carácter data link escape (carácter de salida de la conexión).

DMA (Direct Memory Access). (Véase **Memoria de Acceso Directo.**).

Doble conducto, lógica de. En circuitos asíncronos, cada variable lógica está representada por dos líneas eléctricas, que consideradas en su conjunto pueden tomar tres estados concretos: cero, uno y no dilucidable.

Doble precisión. Cuando se utiliza doble capacidad de memoria para representar un único número. Así se consigue doble precisión en los números decimales. Normalmente suele representarse un carácter de doble precisión en dos medias palabras de memoria.

Doble pulso, grabación en. Grabación magnética en la que los bits se almacenan en una celda formada por dos zonas de distinta polarización. El cero se representa con

una celda polarizada negativamente y positivamente. El uno, viene representado por una celda polarizada positivamente, seguida de negativamente.

Documentación. Conjunto de documentos o información sobre un programa. Muy importante en programas largos, si lo van a utilizar otras personas, o incluso las mismas que lo crearon si transcurre mucho tiempo. En ella se explican muchos detalles útiles sobre el programa (introducción de los datos, modificadores, etc.). Los programas largos deben ser autoexplicativos, introduciéndoles instrucciones REM con comentarios en los puntos en los que se considere necesario.

Documento. Medio en el que se registran datos, para ser utilizados, por ejemplo, un informe, un libro, un formulario, etc. Por extención, también se puede considerar cualquier registro grabado que pueda ser leído.

Documento, borde de referencia del. En reconocimiento de caracteres, borde del documento especificado, tomado como referencia para el alineamiento de los caracteres.

DOS (Disk Operating System). (Véase **Sistema Operativo de Disco.**)

Dot matrix. (Véase **Matriz de agujas.**)

Drive. Unidad de disco o cinta. La parte mecánica de una memoria de disco flexible o de cinta, que maneja su funcionamiento. Se suele utilizar para designar el dispositivo completo, en el que debe introducirse el disquete, la cinta magnética o el cartucho.

Driver. Controlador. Parte del software del sistema que controla un determinado dispositivo. La CPU para comunicarse con los periféricos necesita dos tipos de dispositivos unos: físicos (hardware-bus y controlador) y otros lógicos (software del controlador).

Dummy. Se refiere a parámetros o variables sin sentido que se colocan en una instrucción por razones sintácticas (por ejemplo, para rellenar un hueco que debería estar ocupado por una variable, pero que en el caso concreto que se considera no es de aplicación).

Dump. (Véase **Volcado.**)

Dúplex. En comunicaciones, cuando existen transmisiones en ambas direcciones, de forma independiente. También se puede utilizar un sistema half-dúplex-semidúplex.