Amstrad Educativo

295 Ptas.

Nº 5

- Fichas del Amstrad
- Doctor Logo
- El programa técnico del mes:

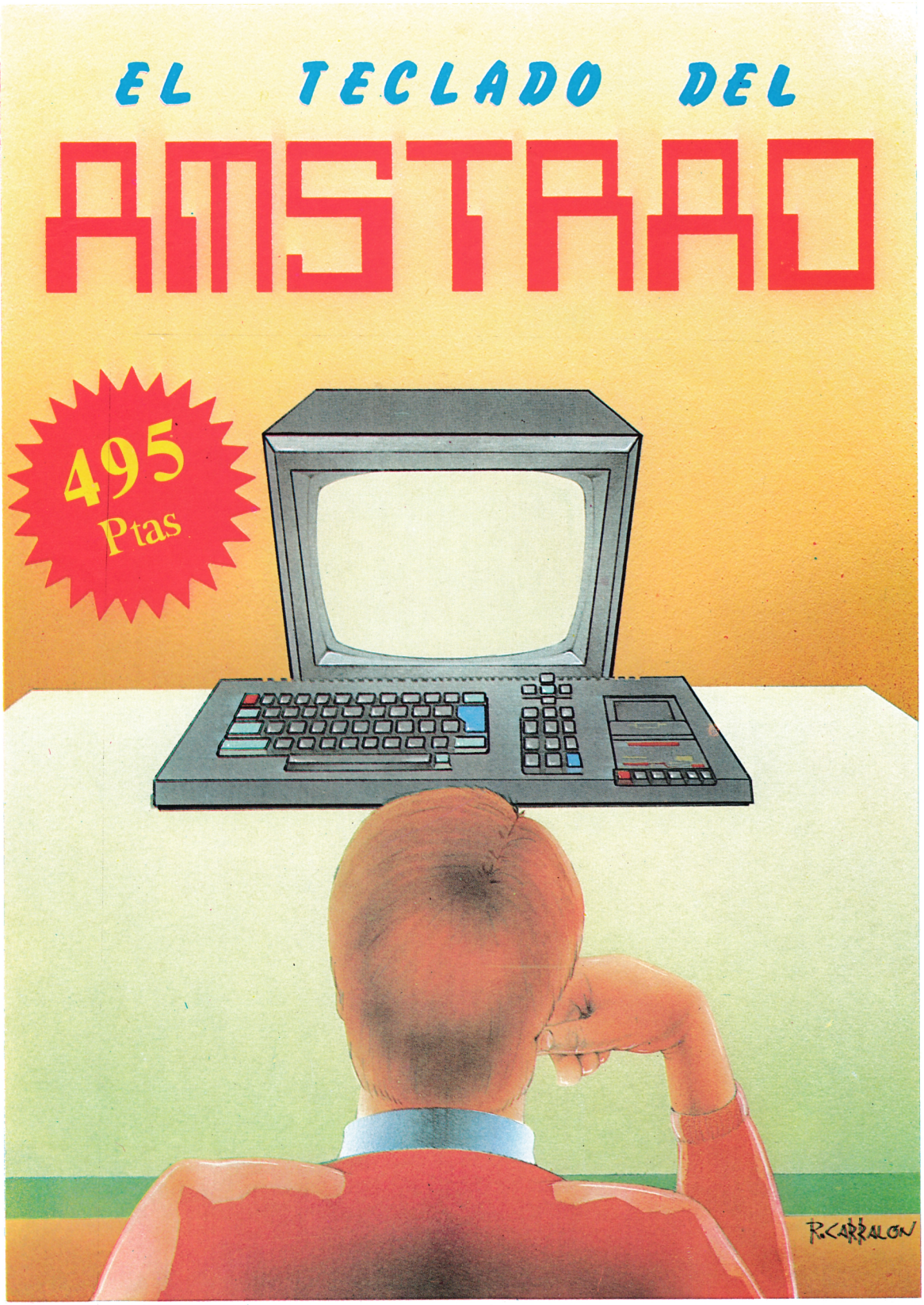
  Teclas definidas

- Logo del Amstrad
- Programando en Basic:

  Revoltillo de letras

¡ya está a la venta!

EL TECLADO DEL AMSTRAD



EDITORIAL

Muchas gracias por el interés que estáis mostrando por nuestra revista, a través de vuestras cartas, opiniones y sugerencias.

Estamos ya tabulando estas indicaciones para irlas adaptando a nuestro esquema, y la planificación se irá ajustando, de esta forma, a nuestras necesidades.

Los programas irán gradualmente haciéndose más complicados y difíciles, a medida que vayáis dominando los niveles más elementales.

Si tenéis en mente algún programa que os gustase ver desarrollado, hacédnoslo saber y nos pondremos manos a la obra inmediatamente para que, cuanto antes, lo veáis publicado en vuestra revista.

El objetivo es que nuestro apelativo "Educativo" sea cada vez más eso, una plataforma que te permita dominar, cada día más a fondo, las posibilidades de tu aparato, formando un triángulo entre la revista, tu aparato y tú.

Esperamos vuestras opiniones.

SUMARIO

Edita: Grupo Editorial G.T.S., S. A.
Bailén, 20-1.º Izda.
28005 Madrid
Telf.: 266 6601-02.
Director: Antonio Bellido.
Colaboran en este número:
Juan M. Pintor, Víctor J. Campo López.
Maquetación: Amalia Moreno.
Secretaria de Redacción: Mercedes Jamart.
Publicidad: Dpto. propio.
Bailén, 20-1.º Izda.
28005 Madrid
Telf.: 266 6601-02

Fotocomposición:
Fotocom, S. A.
Fotomecánica:
La Unión
Imprime:
Gráficas FUTURA Sdad. Coop. Ltda.
Villafranca del Bierzo, 21-23
Políg. Ind. Cobo Calleja
FUENLABRADA (Madrid)
Distribuye:
R.B.A. Promotora de Ediciones, S. A.
Travesera de Gracia, 56 Atico, 1.ª.
Tfno 93 200 82 56
Depósito Legal:
M. 8.904-1986

EL AMSTRAD Y EL CP/M

Por A. BELLIDO

SAVE

Función asignada: Esta orden de CP/M-80 transfiere un determinado conjunto de *bytes* desde el TPA (área de programas transitorios) a un disco instalado en la unidad de disco especificada, asignando el nombre de fichero indicado en la propia orden. Los *bytes* transferidos arrancan de la dirección 256 dec. (100 hex) de la RAM.

El área de usuario activa en el momento de ejecutarse SAVE queda asociada, a todos los efectos, con el nuevo fichero que se genera en el disco.

Forma de obtenerla: Escribir SAVE seguido del número de *páginas* a transferir (ver comentarios) y del nombre del fichero escrito sin ambigüedad.

1.° A >SAVE 40 SALTO.
Esta orden implica la grabación de 40 páginas de memoria en el disco situado en la unidad A, asignándole el nombre de fichero SALTO. La transferencia comienza en la dirección 256 dec. (100 hex) de la RAM y llega a la 10.240 dec. (28 FF hex).

2.° A >SAVE 40 B: SALTO.
Esta orden es idéntica a la anterior, pero la grabación se efectuará en el disco instalado en la unidad B.

3.° A >SAVE 40 SALTO, USER 9, SAVE 40 SALTO.
Con esta línea de orden se graba el fichero SALTO en el área de usuario que esté activa y, a continuación, se activa el área de usuario 9 y se vuelve a grabar, en este área, el fichero SALTO.

Comentarios: Esta orden no suele estar incluida en los manuales de los ordenadores por ser considerada, probablemente, como de uso restringido a especialistas y programadores en lenguaje de asamblea.

Nosotros, aquí, aprovecharemos la ocasión que nos brinda la explicación de "página" para introducirnos en algunos conceptos involucrados con la memoria interna y el microprocesador. No obstante, las líneas que siguen no son de necesaria lectura para la interpretación de los asuntos que aquí se tratan.

En un microprocesador de 8 *bits,* del tipo del Z80, el *tamaño de la palabra* que utiliza viene dado por la *amplitud del camino de datos —data path width—* y corresponde a las ocho líneas usadas para llevar y traer datos a aquel.

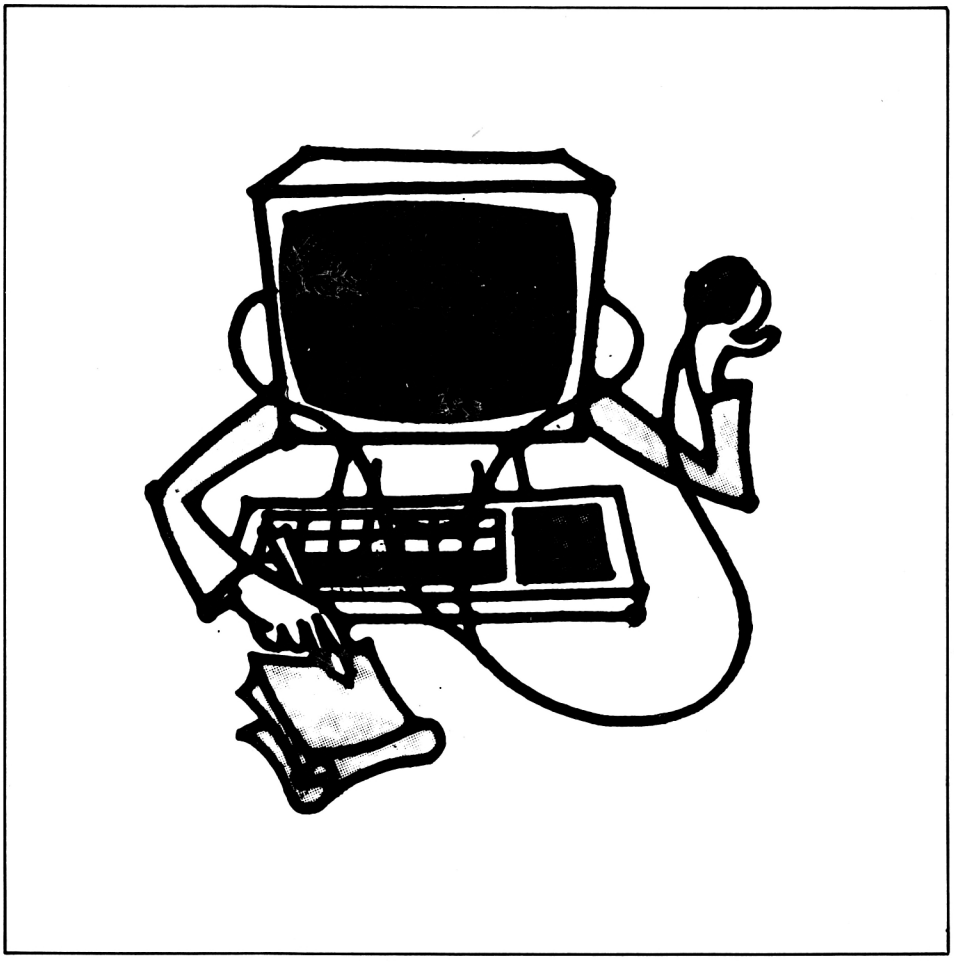
Cada línea transfiere un bit de información.

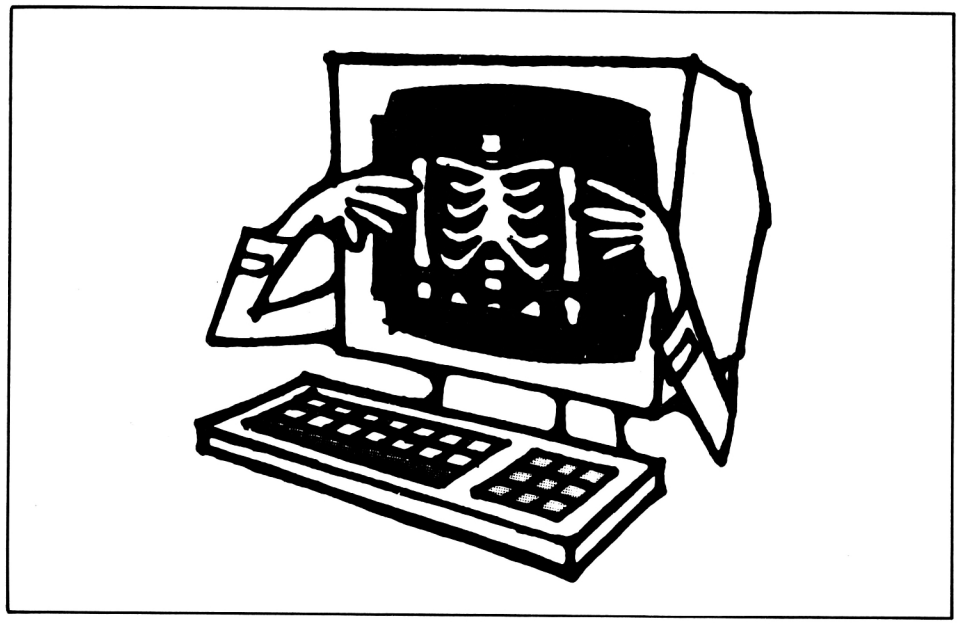
Las ocho líneas transmiten un *byte* por vez.

Al conjunto de estas líneas se le conoce por *bus* de datos, refiriéndonos por D_0 a la que lleva el bit menos significativo y por D_7, a la que hace lo propio con el más significativo y por D. En función de esto, un *byte* de información del tipo 10101010, implicaría los siguientes estados en las líneas;

D_7	D_6	D_5	D_4	D_3	D	D_1	D_0
1	0	1	0	1	0	1	0

En este sentido, y como se apuntó en su momento, es más cómodo re-





ferirse a los números binarios por sus equivalentes hexadecimales (hex) por la facilidad de conversión entre ambas bases.

Así, el binario 10101010 B le corresponde el hexadecimal AAH.

Por otra parte, el microprocesador está capacitado para escribir información en la memoria interna y leerla de ella, *byte* a *byte* secuencialmente, como ya se explicó al hablar de las direcciones de memoria.

Lo usual es referirse a estas direcciones por sus miembros expresados en hexadecimal.

Antes de leer o escribir un byte en una posición de memoria, el microprocesador debe discriminarla por medio de su dirección. Esto sucede a través del *bus* de direcciones. Cada línea de este *bus* transmitirá uno de dos estados posibles (0 ó 1), con lo cual si el bus de direcciones está compuesto por 16 líneas, estará capacitado para direccionar 2^{16} posiciones de memoria o, lo que es igual, 65.536 *bytes* (64 k bytes).

Las líneas del *bus* de direcciones (address) se designan por la letra A con un subíndice del 0 al 15:

$$A_{15}\ A_{14}\ A_{13}\ A_{12}\ A_{11}\ A_{10}\ A_9\ A_8$$
$$A_7\ A_6\ A_5\ A_4\ A_3\ A_2\ A_1\ A_0$$

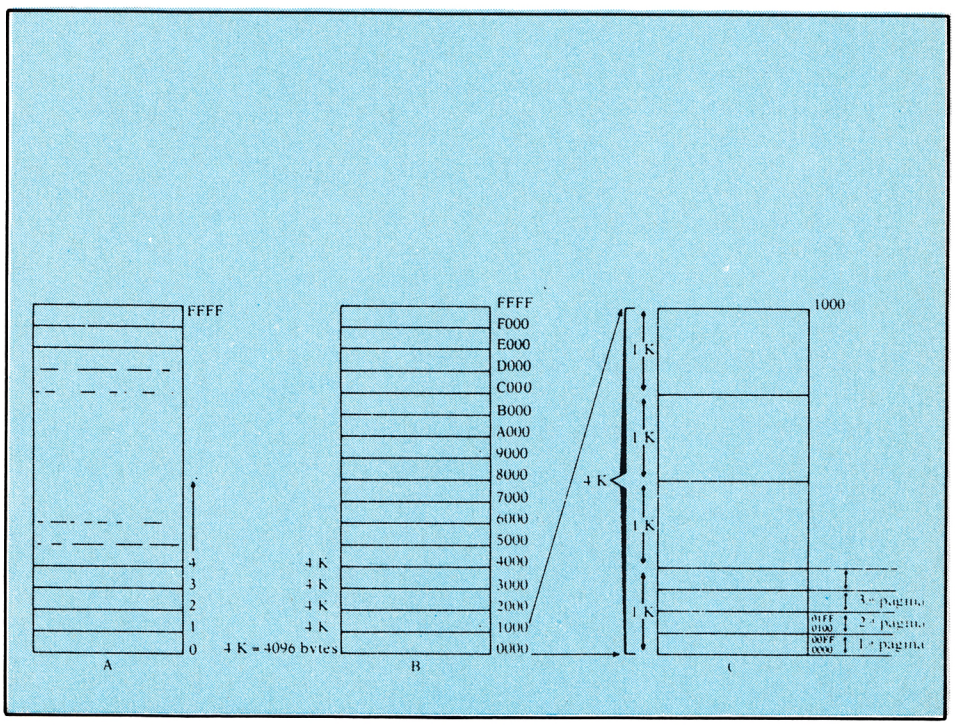
Con estos antecedentes, una dirección dada en el *bus* podría ser:

1111010100001010

F 5 0 A

Este número binario, tras la oportuna conversión al hexadecimal F50A, es evidentemente más cómodo de manejar.

La dirección de memoria más baja es la 00000hex (0 Dec.), y la más alta FFFF hex (65.536 dec.)



En la figura A queda esquematizado el mapa de memoria de un microprocesador con un *bus* de direcciones de 16 bits, partiendo de la dirección 0 y llegando sucesivamente hasta la FFFF hex.

En la figura B, tenemos el mismo mapa, pero dividido en zonas de memoria de 4 K, donde se puede observar que éstas varían acordes con el dígito más significativo hexadecimal.

En la figura C, se divide cada zona de 4 K en cuatro *recintos de memoria* de 1 K *byte,* y cada recinto en cuatro *páginas de memoria,* de lo que podemos concluir que una página es la cuarta parte de un recinto o, dicho de otro modo, 256 *bytes.*

CP/M se reserva la primera página (O-FF) para su uso, con lo cual el TPA empieza propiamente en 100 hex (2.ª página) y a partir de ahí se iniciará la transferencia de las páginas indicadas en la orden SAVE.

TYPE

Función asignada: Mostrar en pantalla el contenido de un fichero de datos —en caracteres ASCII, no programas—,

Para obtener lo mismo por mediación de la impresora, activaremos esta salida —P—, y para detener temporalmente toda salida usaremos S.

El fichero referenciado debe existir en el directorio del área de usuario activa.

Forma de obtenerlo: Teclear TY-

PE, seguido de la referencia al fichero cuyo contenido se reclama escrito sin ambiüedad.

Ejemplos:

1.° Supongamos que al analizar el directorio de un determinado disco instalado en la unidad A no recordamos el contenido del fichero LIBROS .SEC.
En este caso recurriremos a esta orden:
A > TYPE LIBROS. SEC
Con lo cual, al pulsar <cr> para hacerla efectiva, obtendremos:
1. "D. Quijote", "Cervantes", "Planeta", "Letra clara"
2, "El mus", "Mingote", "Prensa Española", "Enseña y entretiene".
3, "La divina comedia", "Dante", "S.A.P.E.", "Letra excesivamente pequeña"
Que es el contenido actual del fichero en cuestión.

2.° Si el disco del ejemplo anterior hubiera estado colocado en la unidad B, la orden debería haber sido:
A >TYPE B: LIBROS. SEC

Comentarios: CP/M, en este caso, localiza el fichero referenciado y transmite su contenido desde el disco a la pantalla. Si el fichero contiene algún tipo de instrucción no comprendida entre los caracteres ASCII capaces de ser impresos, lo más probable es que el computador se bloquee —o "cuelgue"— y deba ser reinicializado.

ORDENES PERMANENTES

1.° ¿Qué obtendremos con la orden DIR C:?
Respuesta: Todos los nombres de los ficheros contenidos en el disco instalado en la unidad C y en el área de usuario activo.

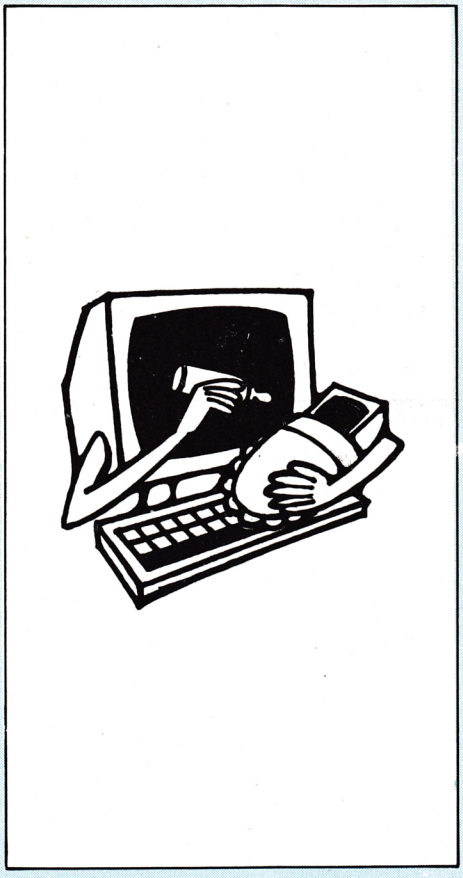
2.° ¿Y con DIR *.*?
Respuesta: Todos los nombres de los ficheros contenidos en el disco instalado en la unidad por defecto y en el área de usuario activa.

3.° ¿Y con DIR?
Respuesta: Igual que en la anterior.

4.° ¿Y con DIR *?
Respuesta: Todos los nombres de los ficheros, que no tengan asignado tipo.

5.° ¿Y con DIR ? R?????? .PRU?
Respuesta: Todos los nombres de los ficheros cuyo segundo caracter del nombre sea R y de tipo PRU.

6.° Dar una orden para borrar el fichero PRUEBA. UNO de la unidad por defecto.
Respuesta: A > ERA PRUEBA. UNO



7.° Borrar el mismo fichero, pero de la unidad B.
Respuesta: A > ERA B: PRUEBA UNO.

8.° Borrar todos los ficheros sin tipo de la unidad por defecto.
Respuesta: A > ERA *.

9.° ¿Qué sucede al dar la orden ERA *.*?
Respuesta: Que CP/M nos da la oportunidad de rectificar mediante el mensaje "ALL (Y/N)?" (TODOS (S/N)?, y si pulsamos "Y", todos los ficheros de la unidad y área de usuario activa son borrados.

10.° ¿En qué forma afecta el área del usuario 12 la orden anterior, si el área de usuario activa es la 0?
Respuesta: De ninguna forma.

11.° ¿Qué efecto produce la siguiente orden?: A > REN EJEMPLO UNO = B: PRUEBA UNO.
Respuesta: Asignamos el nombre EJEMPLO.UNO al fichero PRUEBA.UNO, con lo cual este último desaparece del directorio.

12.° ¿Qué efecto produce esta otra orden?: A > REN C: EJEMPLO. UNO = B: PRUEB.UNO.

Respuesta: Aparte de un error, ninguna ya que REN sólo actúa sobre un mismo disco.

13.° ¿Qué orden hay que dar para conocer el contenido de EJEMPLO.UNO?
Respuesta: TYPE EJEMPLO. UNO.

14.° ¿Qué maniobra tendrá que hacer el operador para congelar la salida de información producida por la orden anterior?
Respuesta: ^S (Pulsar CONTROL y la tecla S, simultáneamente). Para "descongelar", pulsa cualquier tecla.

15.° ¿Qué maniobra tendrá que hacer el operador para conseguir que la salida de información anterior se produzca, además de por pantalla, por impresora?
Respuesta: ^P antes de dar la orden TYPE. Para cancelar salida a través de impresora dar otro P.

16.° ¿Qué efecto produce esta orden?: A > C:
Respuesta: C>. Esto quiere decir que la unidad por defecto ahora es la C.

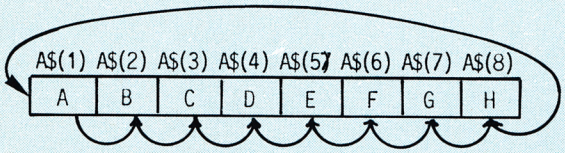
17.° Cambie el actual número de área de usuario (O) a el área 12.
Respuesta: A > USER 12.

LOS FICHEROS EN EL AMSTRAD

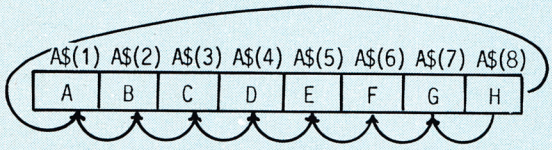
Por A. BELLIDO

ROTACIONES:

Las rotaciones son un caso particular de los deslizamientos por los cuales no se pierde ningún elemento, dando lugar a situaciones como las que se representan en estas figuras:



Rotación a la derecha

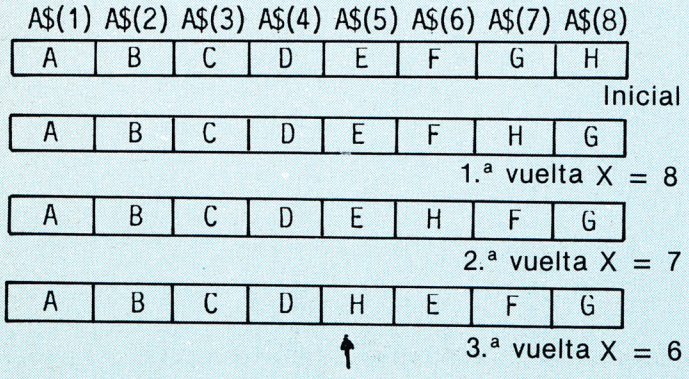


Rotación a la izquierda

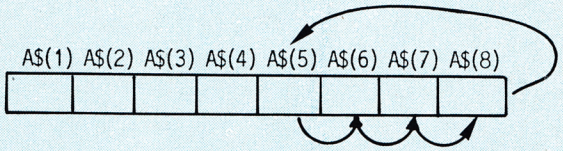
Los programas correspondientes son:

Rotación a la derecha desde el elemento quinto:

```
100 FOR X = 8 TO Y STEP -1
110 A = A$(X): A$(X) = A$(X-1): A$(X-1) = A
120 NEXT X
```



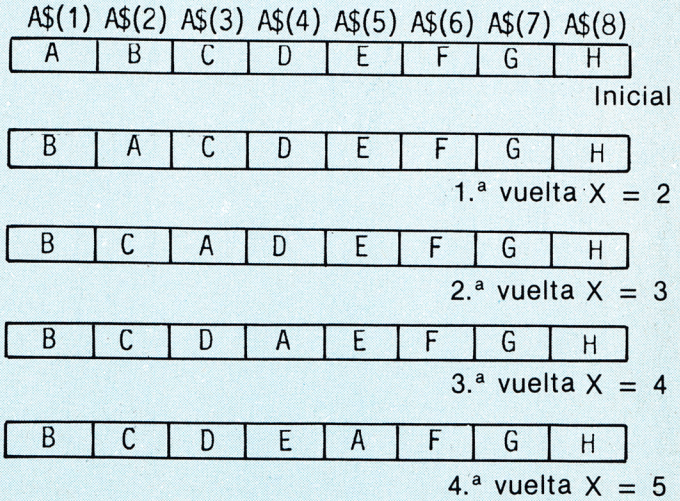
En esta ocasión se ha producido una rotación a la derecha entre el elemento quinto y el último de la matriz, siendo su representación gráfica:



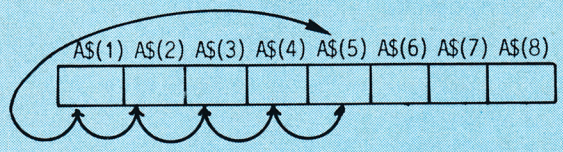
Para conseguir una rotación a la derecha completa de todos los elementos de una matriz bastará con establecer los límites del bucle FOR/NEXT entre el número representativo del subíndice más alto de la matriz y el correspondiente al más bajo más uno.

Rotación a la izquierda desde el elemento quinto:

```
100 FOR X = 2 TO 5
110 A = A$(X): A$(X) = A$(X-1) A$(X-1) = A
120 NEXT X
```



En esta ocasión se ha conseguido una rotación a la izquierda entre el elemento quinto y el primero de la matriz, siendo su representación gráfica:

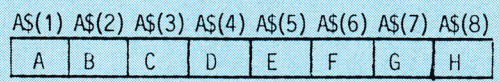


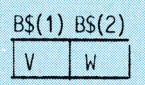
Para conseguir una rotación a la izquierda completa de todos los elementos de una matiz, bastará con fijar los límites del bucle FOR/NEXT entre 2 y el número representativo del subíndice más alto de la matriz.

Existen otras posibilidades de manipulación de los elementos de las matrices como deslizamientos y rotaciones combinados, deslizamientos y transferencias entre matrices, etc., problemas, todos ellos susceptibles de ser resueltos en base a lo expuesto hasta el momento.

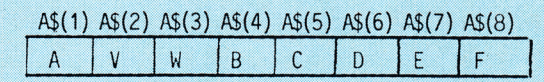
Concluiremos este epígrafe resolviendo un ejercicio que plantee y resuelva uno de los casos propuestos.

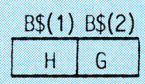
EJERCICIO: Dadas las matrices A$ ( ) y B$( ) cuya representación gráfica es:





Intercalar los valores contenidos en B$(1) y B$(2) en la matriz A$( ) a partir del elemento A$(2) provocando una rotación doble a partir de este elemento que concluya con una transferencia de los elementos de A$( ) a B$( ), siendo la situación final esperada la que sigue:





Solución:

```
100 c = 1: y = 3
110 FOR X = 8 TO y STEP -1
120 A = A$ (X): A$(X) = A$(X-1): A$(X-1) = A
130 NEXT X
140 B = B$(C): B$(C) = A$(X): A$(X) = B
150 c = c + 1: y = y + 1
160 IF c > = 2 THEN 110
```

La idea general de esta rutina es provocar una rotación en la matriz A$( ) desde el elemento segundo a la derecha, con lo cual el elemento pasa a la segunda posición, momento en el que se intercambia el contenido actual de A$(2) —H— con el de B$(1) —V— Hecho esto se repite el proceso a partir del tercer elemento, para lo cual la variable y toma el valor cuatro y c el valor 2. Esto se consigue mediante el mecanismo de rotación (OJO NO SE VE) las líneas están dedicadas a los necesarios ajustes de las variables de control y repetición del bucle si se cumple la condición de borde establecida en 160.

CLASIFICACION:

En el epígrafe anterior, hemos visto cómo almacenar información, numérica o alfanumérica, en los elementos de una matriz, siendo estos elementos una serie de variables capaces de contener, por tanto, un valor.

En base a esto, supongamos que las variables con subíndice que integran la matriz A$( ) contienen los valores que siguen:

```
A$(0) = "QUEVEDO"
A$(1) = "LOPE"
A$(2) = "CERVANTES"
A$(3) = "GONGORA"
```

Y que nuestra intención es preparar un programa que las clasifique alfabéticamente. Dicho de otra forma, el contenido de A$(0) deberá ser "CERVANTES", el de A$(1) "GONGORA", el de A$(2) "LOPE", y el de A$(3) "QUEVEDO".

Una primera aproximación, nos podría llevara al siguiente listado:

```
10 DIM a = $(3)
20 FOR x = 0 TO 3
30 READ a$(x)
40 NEXT x
50 DATA QUEVEDO, LOPE, CERVANTES, GONGORA
60 '************************************
```

```
 70 REM *** CLASIFICACION ***
 80 BANDERA = 0
90 IF A$(0) ⩾ A$(1) THEN A = A$(0): A$(0) = A$(1): A$(1)
= A
100 IF A$(1) ⩾ A$(2) THEN SWAP A$(1), A$(2): BANDE-
RA: 1.
110 IF A$(2) ⩾ A$(3) THEN A = A$(2): A$(2) = A$(3): A$(3)
= A BANDERA = 1
130 IF BANDERA = 1 THEN 80
140 PRINT A$(0), A$(2), A$(3)
150 END
```

Un simple vistazo a este programa, nos permite ver que la variable BANDERA tomará el valor 1, en tanto en cuanto haya un intercambio de contenido entre las variables de la matriz, es decir, mientras no estén clasificadas.

En el momento en que BANDERA llegue a la comparación de la línea 130 con el valor 0 la lista está ordenada, permitiéndose la impresión de los elementos de la matriz (línea 140).

Esta forma de clasificación exige aumentar el número de comparaciones entre los elementos a ordenar a medida que aumentan estos.

El siguiente algoritmo permite esta operación para cualquier número de elementos sin que el número de pasos del proceso varíe.

CLASIFICACION POR EL METODO DE REBOND

```
 10 DIM A$ (4)
 20 FOR X = 0 TO 4
 30 READ A$ (X)
 40 NEXT X
 50 DATA QUEVEDO, LOPE, CERVANTES, GONGORA,
    GOMARA
 60 ****************************************
 70 REM *** CLASIFICACION REBOND ***
 80 BANDERA = 0
 90 FOR C = 0 TO 3
100 IF A$(C1) ⩾ A$(2) THEN A = A$(C1): A$(C1) = A$(2):
A$(2) = A BANDERA = 1
110 NEXT C
120 IF BANDERA = 1 THEN 80
130 PRINT A$(0), A$(1), A$(2) A$(3), A$(4)
```

Como se puede observar, este procedimiento está basado en el mismo criterio usado en el programa precedente.

Mientras la BANDERA llegue a la comparación de la línea 120 con el valor 1 el proceso se repite desde la línea 80 y, por tanto, la línea 130 sólo se alcanza cuando BANDERA es 0.

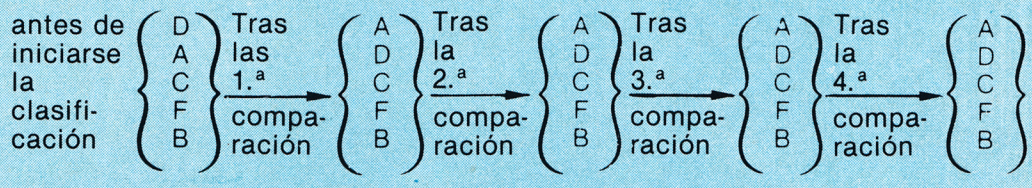
Con respecto al límite impuesto a la variable C en la instrucción de la línea 90, se debe considerar que el último elemento no puede compararse con otro ulterior.

CLASIFICACION POR EL METODO DE LA BURBUJA

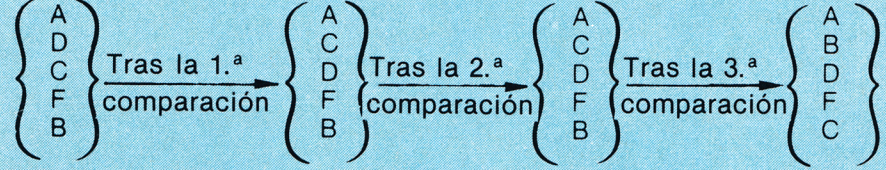
```
 10 DIM A$(4)
 20 FOR X = 0 TO 4
 30 READ A$ (X)
 40 NEXT X
 50 DATA QUEVEDO, LOPE, CERVANTES, GONGORA,
    GOMARA
 60 ****************************************
 70 REM *** CLASIFICACION BURBUJA ***
 80 FOR C1 = 0 TO 3
 90 FOR C2 = 1 TO 4
100 IF A$(C) A$(C + 1) THEN A = A$(C): A$(C) =
A$(C + 1): A$(C + 1) = A
110 NEXT C2
120 NEXT C1
130 PRINT A$(0), A$(1), A$(2), A$(3), A$(4)
```

Con este procedimiento y a través del bucle que se inicia en la línea 80, se coloca cada elemento de la matriz con todos los que le siguen excepto el último, claro mediante la instrucción de la línea 100, se garantiza que los valores contenidos en las respectivas variables llegarán a la 130 en orden alfabético de menor a mayor.

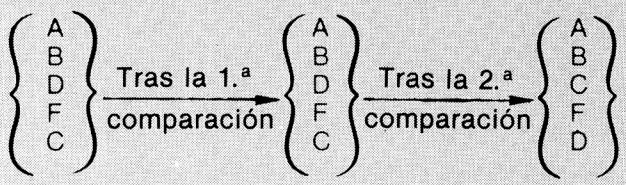
Este método no es el más rápido, pero si el más evidente, y para verlo en acción pongamos que el contenido de los elementos de la matriz A$( ) es:



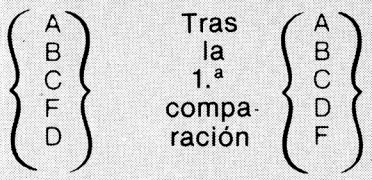
En esta serie de comparaciones la variable C1 es 0 y C2 ha tomado 1, 2, 3 y 4 sucesivamente.



En esta serie de comparaciones la variable C1 es 1 y C2 ha tomado 2, 3 y 4 sucesivamente



En esta serie de comparaciones la variable C1 es 2 y C2 ha tomado 3 y 4 sucesivamente.



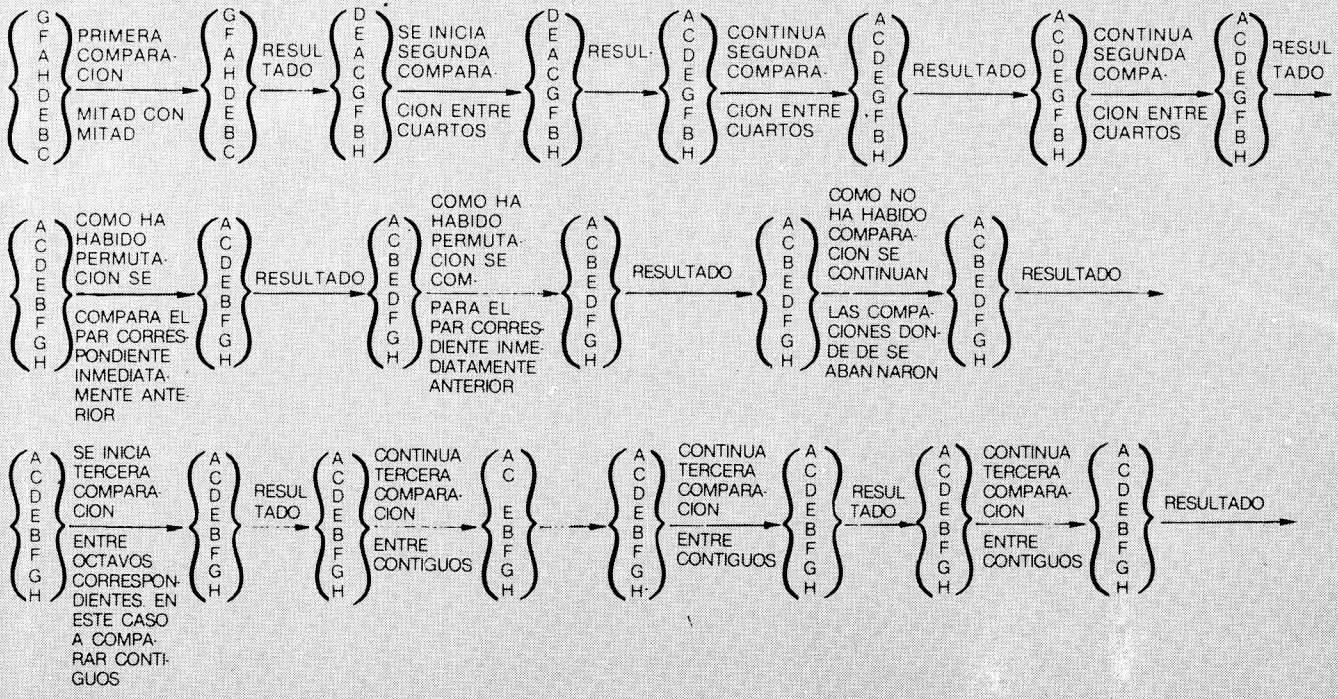
En este serie de comparaciones la variable C1 es 3 C2 ha tomado exclusivamente.

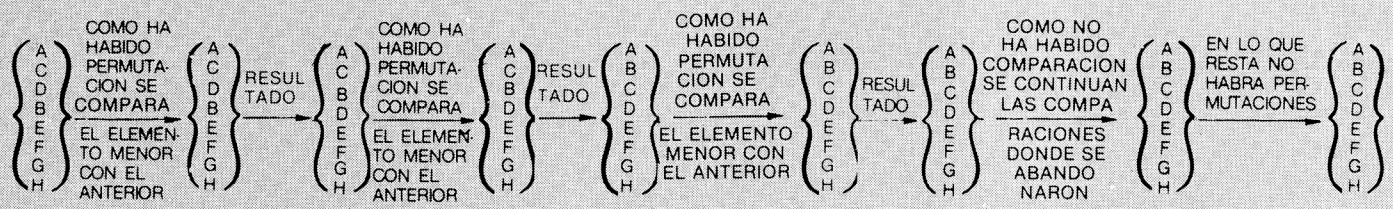
CLASIFICACION POR EL METODO DE SHELL-METZNER

Los dos métodos anteriores se han caracterizado por comparar cada variable de la matriz con la que le sigue en orden, así: A$(n) con A$(n + 1), y, si es necesario, intercambiar su contenidos.

En esta ocasión se va a comenzar por comparar una mitad de la lista de elementos con la otra para concluir que en la mitad inicial están las variables cuyo contenido, es alfabéticamente menor que el de los correspondientes en la mitad final. A continuación, se compara el cuarto inicial de la lista con el que le sigue, y éste con el tercer cuarto, y éste con el último, quedando en los cuartos comparandos más próximos al inicio las cadenas inferiores alfabéticamente. El siguiente paso será comparar el octavo inicial de la lista con el segundo octavo, etc., y así sucesivamente hasta comparar cada elemento con el que le sigue, y, si fuera necesario en esta estapa —o en las anteriores— una permutación, tendríamos que dar marcha atrás en el proceso de comparaciones anterior para asegurar que tras el proceso de comparaciones anteriores sigue siendo efectivo.

Para aclarar lo expuesto, supongamos una lista desordenada de ocho elementos, P.e.:





Un programa que responde a este proceso de clasificación es:

```
 10 DIM A$(7)
 20 FOR X = 0 TO 7
 30 READ A$(X)
 40 NEXT X
 50 C = X -1
 60 DATA GARCIA, FERNANDEZ, ANTON, HERNANZ,
    DESOTO, ENTERRIA, BILBAO, COLL
 70 ****************************************
 80 REM *** CLASIFICACION SHELL -METZNER ***
 90 D = INT (C/2)
100 WHILE D > = 1
110 CS = 0
120 WHILE CS < = C-D
130 VA = CS
140 IF A$ (VA) A$ (VA + D) THEN A =  A$(VA): A$(VA)
= A$(VA + D): A$(VA + D) = A VA = VA-D: IF VA ⩾
= 0 THEN 140
150 CS = CS + 1
160 WEND
170 D = INT (D/2)
180 WEND
190 PRINT A$(0), A$(1), A$(2), A$(3), A$(4), A$(5), A$(6),
A$(7)
```

Entre las líneas 10 y 60, se dimensiona una matriz y se valoran sus elementos. La línea da lugar a la primera división de la lista de elementos en dos mitades; mitades se fija el "paso" o "distancia" de los elementos a comparar.

En la línea 100, se establece la condición de que mientras la división de la lista de particiones mayores o iguales a uno el proceso continúa. En otra palabras, la clasificación continuará hasta comparar elementos conjuntos contiguos: "paso" = 1.

En la línea 110, se inicializa la variable CS representativa de los subíndices de los elementos de la matriz.

En la línea 120, se establece la condición que garantiza que los dos elementos del par actual a comparar está dentro de la lista.

En la línea 130, se valora una variable auxiliar (VA) con el contenido de CS, siendo esta variable auxiliar la que representará el subíndice de los elementos a comparar, con lo que se preserva CS de cualquier contingencia como la que se puede producir en la línea siguiente.

La línea 140 provoca el intercambio —si ha lugar— de contenido de los elementos que se comparan y, a continuación, deduce de VA el valor del "paso" para dar "marcha atrás" en el proceso de comparaciones, asegurando así que el orden alfabético establecido previamente sigue siendo correcto tras el último intercambio.

En 150, se pasa al siguiente subfijo.

En 160, se cierra el bucle dando lugar, a partir de 120, a la repetición del proceso con el siguiente elemento de la lista, aún con el mismo "paso".

A la línea 170, sólo se pasa cuando todos los elementos correspondientes de la actual subdivisión de la lista han sido comparados. En este momento se procede a subdividir la lista nuevamente, dando con ello lugar a un nuevo "paso".

En la línea 180, se cierra el bucle dando lugar, a partir de 100, a todo el proceso de comparaciones que exige el nuevo "paso".

Finalmente, en 190 se pide la impresión de los elementos de la lista para comprobar que están ordenados.

El método de Shell Metzner, es como mucho, el más rápido de los expuestos en este epígrafe y dependerá de la lista a clasificar el utilizar uno u otro.

Existen otros métodos de clasificación, como el QuickSoct, que no se tratan aquí. A continuación se expone una listado para clasificar según Shell-Metzner, pero evitando el uso de la instrucción WHILE/WEND.

```
 10 DIM A$ (7)
 20 FOR X = 0 TO 7
 30 READ A$ (X)
 40 NEXT X
 50 C = X -1
 60 DATA GARCIA, FERNANDEZ, ANTON, HERNANZ,
    DESOTO, ENTERRIA, BILBAO, COLL
 80 REM *** CLASIFICACION SHELL-METZNER ***
 90 D = INT (C/2)
100 CS = 0
110 VA = CS
120 IF A$(VA) A$(VA + D) THEN A = A$(VA): A$(VA) =
A$(VA + D): A$(VA + D) = A VA = VA-D: IF VA > =
0 THEN 120
130 CS = CS + 1
140 IF CS < = C-D THEN 110'
150 D = INT (D/2)
160 IF D> = 1 THEN 100
170 PRINT A$(0), A$(1), A$(2), A$(3), A$(4), A$(5), A$(6),
A$(7l)
180 END
```

La interpretación de estas líneas pasa por el mismo razonamiento ya expuesto en el anterior listado.

Bien, de una forma u otra, ya estamos en condiciones de clasificar cualquier lista.

EL BASIC DE

Por A.

LET	LET variable = ex

INTERPRETACION:

Asigna a una variable el valor de una expresión.

POSIBILIDADES:

Recordemos previamente que una *variable* contiene un *valor* que puede cambiar a lo largo de un algoritmo, pero que permanece fijo mientras no haya una instrucción que así lo determine.

Existen dos clases de variables: *numéricas* y de *caracteres.*

Las variables numéricas se representan por cualquier conjunto de caracteres —a veces, por una sola letra seguida de un número— con la condición de que el primero sea una letra. Ejemplo: LET ABc1 = *45.*

Las variables de caracteres se representan por una sola letra seguida del símbolo $. De esta forma, el computador sabe que el contenido de la variable será un texto, que deberá ser introducido entre comillas. Ejemplo: LET A$ = "Cervantes".

Al referirnos a las variables, dijimos que son elementos capaces de contener un valor, *una expresión,* por lo contrario, *es un valor.*

Nada impide que el valor de una expresión varíe de acuerdo con los valores de las variables que forman parte de la expresión.

FORMA DE TECLEAR LA INSTRUCCION

Teclear LET, el nombre de la variable numérica, el signo igual (=) y la expresión numérica, seguido de **ENTER.**

Teclear LET, el nombre de la variable alfanumérica, el signo igual (=) y la expresión alfanumérica entre comillas ("), seguido de **ENTER.**

Recordar:

— Que el nombre de la variable numérica puede ser una letra o una letra seguida de otros caracteres.
— Que el nombre de la variable alfanumérica siempre es una letra seguida del símbolo $.

EJEMPLO:

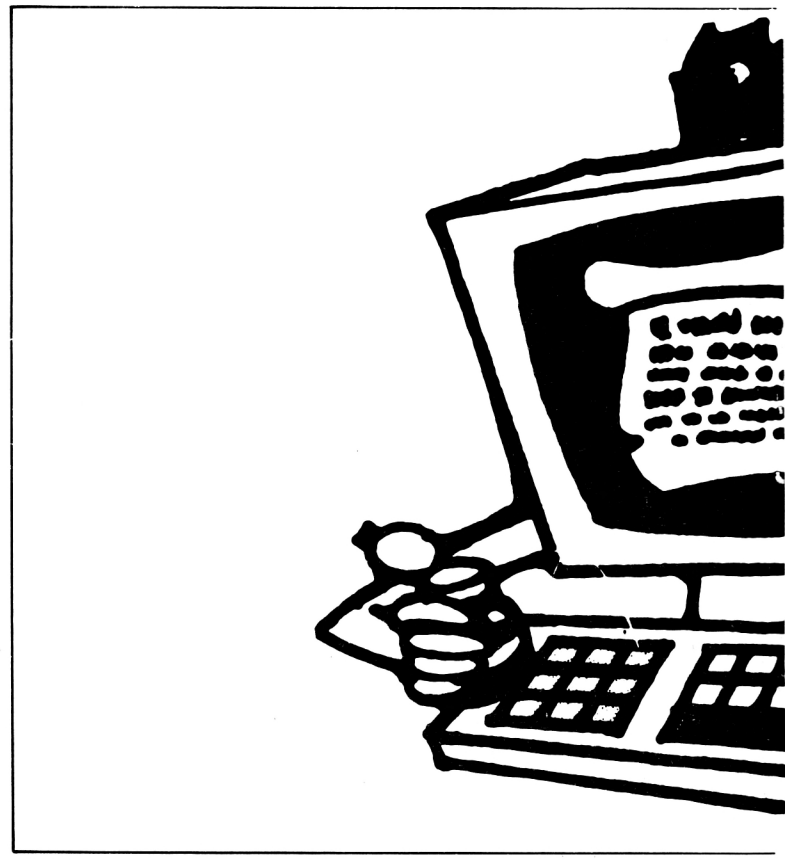
1. Definir una expresión que determine el valor del importe de una conferencia con Sevilla, sabiendo que el valor de cada "paso" de contador lo podemos averiguar en la Telefónica.

SOLUCION:

Empezaremos por fijar y nombrar las variables que intervienen:

Llamaremos **P** a la variable que contendrá el valor de un "paso".

Llamaremos **N** a la variable que contendrá el número de "pasos".



BELLIDO

presión | **DEJAR**

Y llamaremos **I** a la variable que contendrá el importe de la conferencia con Sevilla.

Hecho esto, tendremos que definir la *expresión* que determine el valor del importe, en función de los valores que contengan las variables implicadas:

I = P * N

Es evidente que el importe de la conferencia será igual al resultado de multiplicar el número de "pasos" por el valor de un "paso".

Supongamos que, después de observar en el contador el número de pasos y de habernos informado en la Telefónica del valor de un "paso", los valores de las variables **P** y **N** son, respectivamente, **10** y **25.** Es decir, un "paso" vale 10 ptas. y el número de "pasos" de la conferencia ha sido 25. En función de esto, la forma de asignar variables en BASIC, teniendo en cuenta el comando LET, será:

```
LET P = 10
LET N = 25
LET I = P * N
```

Como podemos ver, la expresión en una variable numérica puede ser tanto un número como una expresión matemática, la cual, finalmente, es un número.

EJEMPLO:

2. Asignar a una variable de caracteres la expresión: **El importe de su conferencia es.**

SOLUCION:

```
LET A$ = "El importe de su conferencia es"
```

De esta forma, el ordenador sabe que la variable **A$** contiene el texto entrecomillado.

Observe el espacio que queda entre *es* y las comillas. Sirve para evitar que la cifra que representa el valor de la conferencia quede impresa junto a *es.*

EJEMPLO:

3. Escriba un programa que, una vez ejecutado, nos dé el importe de una conferencia con Sevilla de 25 "pasos", siendo el valor de cada paso 10 ptas.

SOLUCION:

PROGRAMA	COMENTARIOS
10 LET P = 10 **20 LET N = 25** **30 LET I = P * N** **40 LET A$ = "El importe de su conferencia es "** **50 PRINT A$; I**	Con las cuatro primeras líneas asignamos nombre y contenido a las variables. Con la última línea nos limitamos a ordenar la impresión del contenido actual de las variables A$ e I.

El importe de su conferencia es 250

EJERCICIOS:

1. *Costes:*
   1.1. ¿Cuánto valen 5 Kg. de melocotones a 115 ptas. el Kg.?
   1.2. ¿Cuánto valen 5 Kg. de melocotones a 115 ptas. el Kg. y 7 Kg. de melón, a 96 ptas, el Kg.? Hallar el valor total.
   1.3. ¿Cuánto me devolverán en la frutería, si para pagar 5 Kg. de melocotones, a 115 ptas., el Kg. y 7 Kg. de melón, a 96 ptas. el Kg. entrego 1.500 Ptas.?
2. *Ahorro:*
   2.1. Si Juan tiene 2.500 Ptas. y gasta 1.875 ¿cuánto le queda?
   2.2. A Juan le dan sus padres 1.000 ptas. cada semana y gasta 325 ptas. en el cine y 500 en libros. ¿Cuánto dinero ahorra en 15 semanas?
3. *Areas:*
   3.1. Hallar el área de un cuadrado de 6 cm. de lado.
   3.2. Hallar el área de un círculo circunscrito a un cuadrado de 6 cm. de lado.
   3.3. Hallar el área de la corona circular comprendida entre las circunsferencias inscrita y circunscrita a un cuadrado de 6 cm. de lado.
   3.4. Hallar el área lateral y el área total de un cilindro de 5 cm. de radio y 12 cm. de altura.
4. *Volúmenes:*
   4.1. Hallar el volumen de un cubo de 10 cm. de arista.
   4.2. Hallar el volumen de un cilindro inscrito en un cubo de 20 cm. de arista.
   4.3. Hallar el volumen de un cilindro circunscrito a un cubo de 10 cm. de arista.
   4.4. Hallar la diferencia de volumen entre los cilindros inscrito y circunscrito a un cubo de 10 cm. de arista.
5. *Lenguaje:*
   5.1. Vocabulario:
   Dadas las dependencias de una casa (vestíbulo, comedor, cocina, etc.) escribir el programa que, al ejecutarlo, imprima en pantalla: *Las dependencias de mi casa son: el vestíbulo, etc.*
   5.2. Gramática:
   Dadas las diferentes clases de palabras (artículo, nombre, adjetivo, etc.) escribir el programa que, ejecutado, imprima en pantalla *Las clases de palabras que pueden formar el sintagma nominal son: el artículo, el nombre, etc.*
6. *Geografía e Historia:*
   6.1. Geografía:
   Dada una relación de ríos españoles, escribir el programa que, ejecutado, imprima en pantalla: *Los ríos españoles que desembocan en el Atlántico son: El Miño, El Duero, etc.*
   6.2. Historia:
   Dada una relación de personajes españoles, escribir el programa que, ejecutado, imprima en pantalla: *Son personajes importantes en el descubrimiento y conquista de América: Colón, Pizarro, etc.*
7. *Ciencias Naturales:*
   7.1. Dada una relación de las partes de una planta, escribir el programa que, ejecutado, imprima en pantalla: *Las partes del aparato vegetativo de una planta son: la raíz, el tallo, etc.*
   7.2. Dada una relación de vertebrados, escribir el programa que, ejecutado, imprima en pantalla:
   Son mamíferos...
   Son aves...
   Son reptiles...
   Son peces...
   Son anfibios...

SOLUCIONES:

1.1.
```
1 REM Ficha "LET" Ej. 1.1.1
10 LET K = 115: REM "Precio del kg."
20 LET m = 5: REM "Número de kg."
30 PRINT "Valen"; k : m
```

1.2.
```
1 REM Ficha " LET " Ej. 1.1.2
10 LET p = 115: REM "Precio del kg. de melocotón"
20 LET p1 = 96: REM "Precio del kg. de melón"
30 LET k = 5: REM "Kilos de melocotón"
40 LET k1 = 7: REM "Kilos de melón"
50 PRINT "El valor total es "; p * k + 1 * k1
```

1.3.
```
1 REM Ficha "LET " Ej. 1.1.3
10 LET p = 115: REM "Precio del kg. de melocotón"
20 LET p1 = 96: REM "Precio del kg. de melón"
30 LET k = 5: REM "Kilos de melocotón"
40 LET k1 = 7: REM "Kilos de melón"
50 PRINT "me devolverán ";—p* k + —
   p 1*k1 + 1500;" pts."
```

2.1.
```
1 REM Ficha "LET "Ej. 1.2.1.
10 PRINT "Le queda "; 2500-1875
```

2.2.
```
1 REM Ficha " LET " Ej. 1.2.3
10 PRINT "Ahorra "; 15*1000-15*325-15*500
```

3.1.
```
1 REM Ficha "LET "Ej. 1.3.1.
10 PRINT "El área es ";6*6;" cm.2."
```

3.2.
```
1 REM Ficha "LET " Ej. 1.3.3.
10 PRINT "El área es"; 2*PI*SQR 2*6;" cm2.
```

3.3.
```
1 REM Ficha "LET " Ej. 1.3.4
10 PRINT "El área es: ";PI*2*62/22—
   PI*62/22; cm2."
```

3.4.
```
1 REM Ficha " LET " Ej. 1.3.5.
```

```
10 PRINT "El área lateral es ";2*PI*5/2*
   12;" cm2."
20 PRINT "El área total es ";2*PI*5/2*12
   +2*PI*52/22;cm2."
```

4.1.
```
1 REM Ficha "LET " Ej. 1.4.1
10 PRINT "El volumen es ";103;" cm2."
```

4.2.
```
1 REM Ficha "LET " Ej. 1.4.2
10 PRINT "El volumen es ";PI*102/22*
   10;" cm2."
```

4.3.
```
1 REM Ficha: "LET " Ej. 1.4.3
10 PRINT "El volumen es ";PI*2*102/22*
   10;" cm2."
```

4.4.
```
1 REM Ficha: "LET " Ej. 1.4.4
10 PRINT "El volumen es ";PI*2*102/22
   *10—PI*202/22*10; cm2."
```

.5-1.
```
1 REM Ficha: "LET "Ej. 2.1.1
10 LET A$ = "vestíbulo, "
20 LET B$ = "cocina, "
30 LET C$ = "comedor, "
40 LET D$ = "salón, "
50 LET E$ = "dormitorios, "
60 LET F$ = "cuarto de baño, "
70 LET G$ = "servicio, "
80 LET H$ = "trastero, "
90 LET I$ = "jardín, "
100 PRINT "Las dependencias de mi casa
    son ";
110 PRINT A$ + B$ + C$ + D$ + E$ +
    F$ + G$ + H$ + I$
```

5.2.
```
1 REM Ficha:" LET " Ej. 2.1.2.
10 LET A$ ="artículo, "
20 LET B$ ="nombre, "
30 LET C$ ="adjetivo, "
40 LET D$ ="pronombre, "
50 LET E$ ="verbo, "
60 LET F$ ="adverbio, "
70 LET G$ ="preposición, "
80 LET H$ ="conjunción, "
90 PRINT "Las clases de palabras que
    pueden formar el sintagma nominal,
    son ";
100 PRINT A$ + B$ + C$
```

6.1.
```
1 REM Ficha: "LET "Ej. 2.2.1
10 LET A$ ="Bidasoa, "
20 LET B$ ="Nervión, "
30 LET C$ =" Nalón, "
40 LET D$ ="Navia, "
50 LET E$ ="Mino, "
60 LET F$ ="Duero,"
70 LET G$ ="Tajo, "
80 LET H$ ="Guadiana, "
90 LET I$ ="Guadalquivir, "
110 LET K$ ="Ebro, "
120 LET L$ ="Ter, "
130 LET M$ ="Llobregat, "
140 PRINT "Los ríos españoles que desem
    bocan en el Atlántico, son ";
```

6.2.
```
1 REM Ficha: " LET " Ej. 2.2.2
10 LET A$ = "Colón "
20 LET B$ = "Hernán Cortés, "
30 LET C$ = "Palafox, "
40 LET D$ = "Núñez de Balboa, "
50 LET E$ = "Viriato, "
60 LET F$ = "Aníbal, "
70 LET G$ = "Francisco Pizarro, "
80 LET H$ = "Elcano, "
90 LET I$ = "Juan de Austria, "
100 PRINT "Son personajes importantes
    en el descubrimiento y la conquista de
    América, ";
150 PRINT A$ + B$ + C$ + D$ + G$ + H$
```

7.1.
```
1 REM Ficha:"LET "Ej. 2.3.1.
10 LET A$ ="raíz, "
20 LET B$ ="flor, "
30 LET C$ ="tallo, "
40 LET D$ ="fruto, "
50 LET E$ ="pistilos, "
60 LET F$ ="polen, "
70 LET G$ ="corola, "
80 LET H$ ="hojas, "
90 LET I$ ="estambres, "
100 PRINT "Las partes del aparato
    vegetativo de una planta, son ";
150 PRINT A$ + C$ + H$
```

7.2.
```
1 REM Ficha:"LET "Ej.2.3.2.
10 LET A$ ="conejo,"
20 LET B$ ="tortuga"
30 LET C$ ="serpiente, "
40 LET D$ ="perro, "
50 LET E$ ="gato, "
60 LET F$ ="gorrión, "
70 LET G$ ="aguila, "
80 LET H$ ="camello,"
90 LET I$ ="salmón,"
100 LET J$ ="sardina,"
110 LET K$ = "jurel,"
120 LET L$ = "ballena,"
130 LET M$ ="rana,"
140 PRINT "Son mamíferos,";
150 PRINT A$ + D$ + E$ + H$ + L$
160 PRINT "Son aves,";
170 PRINT F$ + G$
180 PRINT "Son reptiles,";
190 PRINT B$ + C$
200 PRINT "Son peces,";
210 PRINT I$ + J$ + K$
220 PRINT "Son anfibios,";
230 PRINT B$ + M$ + B$
```

LOWER$ ¡("expresión")!

Retorna la **expresión** alfanumérica con todos los caracteres que la integran en minúsculas.

Ejemplo:

```
PRINT "A b c D"
Retorna
a b c d
```

MAX ¡(Lista de expresiones)!

Retorna el mayor valor de la **lista de expresiones.**

Ejemplo:

```
10 x = 2: y = 30 * x
20 PRINT MAX (4, x, y)
Retorna
60
```

MERGE ¡(nombre de un programa)!

Permite mezclar un programa procedente del disco con el que ocupa la memoria del ordenador, instituyendo en éste, aquellas líneas de programa que, procedentes del programa a mezclar, coincidan en su número.

Los programas salvados como ficheros protegidos pueden ser mezclados.

MID$ ¡(expresión", expresión1, expresión2)!

Retorna una subcadena de **expresión 1** cuyo primer caracter es el que ocupa la posición dada por **expresión 1** y que está formada por tantos caracteres como indica **expresión 2.**

Ejemplo:

```
  10 A$ = "A B C D E F G H"
 20 B$ = MID$ (A$,LEN (A$) 12, 3)
 30 PRINT B$
Retorna
DEF
```

MIN ¡(Lista de expresiones)!

Retorna el menor valor de la **Lista de expresiones.**

Ejemplo:

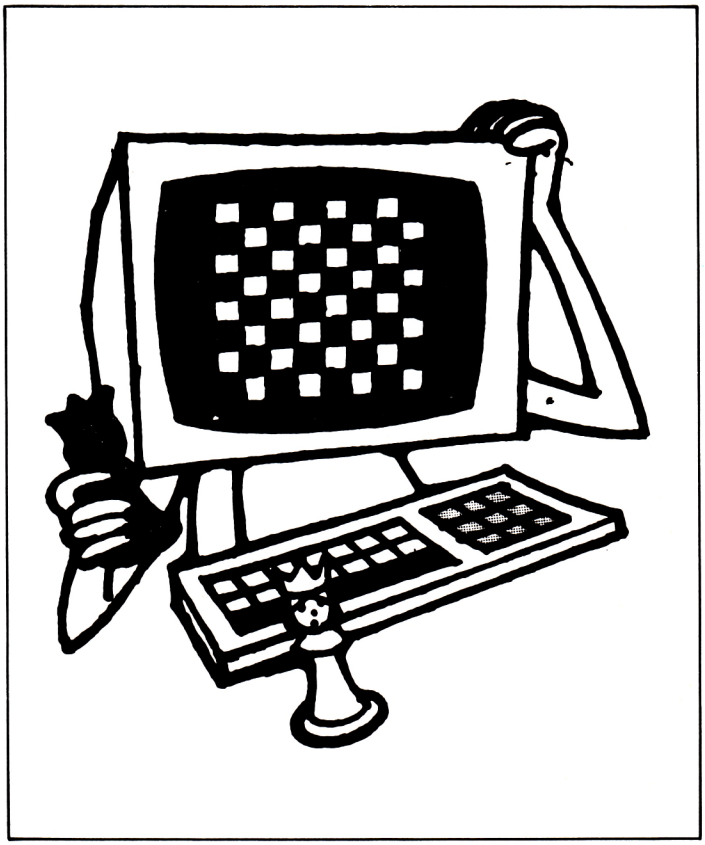
```
 10 x = 2: y = 30 * x
 20 PRINT MIN (4, x, x)Z
Retorna
2
```

expresión 1, MOD expresión 2

Retorna el resto de la división de **expresión1** entre **expresión2**

Ejemplo:

```
PRINT 13 MOD 6
Retorna
1
```



NEW

Borra cualquier programa y variables que existan en memoria, manteniendo el contenido de pantalla, la definición de todos y los atributos de presentación en pantalla.

ON BREAK CONT

Produce una continuación del programa tras una orden ESC. Para cancelar su efecto hay que recurrir a una reinicialización del sistema.

Ejemplo:

```
10 ON BREAK CONT
20 PRINT "A": CLS
30 GOTO AD
```

Resultado
Este programa se ejecutará ininterrumpidamente aún cuando se pulse ESC.

ON BREAK GOSUB ¡número de línea!

Produce una salto a la subrutina que comienza en **número de línea** al pulsar dos veces ESC.

ON BREAK STOP

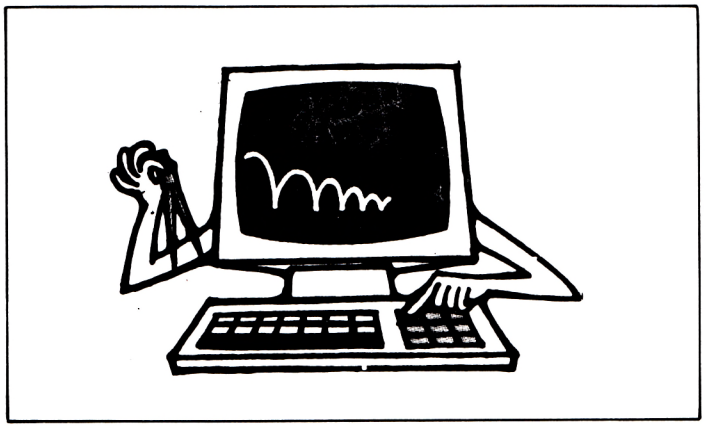
Produce una interrupción del programa que se está ejecutando, inhabilitando ON BREAK CONT y ON BREAK GOBUB.

ON ERROR GOTO ¡número de línea!

Produce un salto a **número de línea** en la lectura del programa cuando aparece un error durante la ejecución.

Ejemplo:

```
10 ON ERROR GOTO 40
20 CLS : c = 250
30 PRINT c, CHR$ (c): c = c + 1: GOTO BO
```



```
40 PRINT "Detectado un error"
50 INPUT "Pulse RETURN para continuar"; 0
60 GOTO 10
```

Resultado
Este programa se ejecutará sin problemas hasta que la variable c supera 255 y pretenda la impresión de un caracter SCCI inexistente, en cuyo momento el programa sigue la ejecución en la línea 40.

ON ¡expresión! GOSUB ¡lista de números de línea!

Produce un salto a la subrutina cuya línea de comienzo ha sido seleccionada por **expresión** dentro de la **lista de números de líneas.**

Ejemplo:

```
 10 CLS: PRINT "Pulse tecla de la opción elegida".
 20 PRINT "1... ALTAS"
 30 PRINT "2... BAJAS"
 40 PRINT "3... MODIFICACIONES"
 50 INPUT T
 60 ON T GOSUB 100, 200, 300
 70 GOTO 10
100 REM COMIENZA RUTINA DE ALTAS
110 ....
190 RETURN
200 REM COMIENZA RUTINA DE BAJAS
210 ...
290 RETURN
300 REM COMIENZA RUTINA DE MODIFICACIONES
310 ...
390 RETURN
```

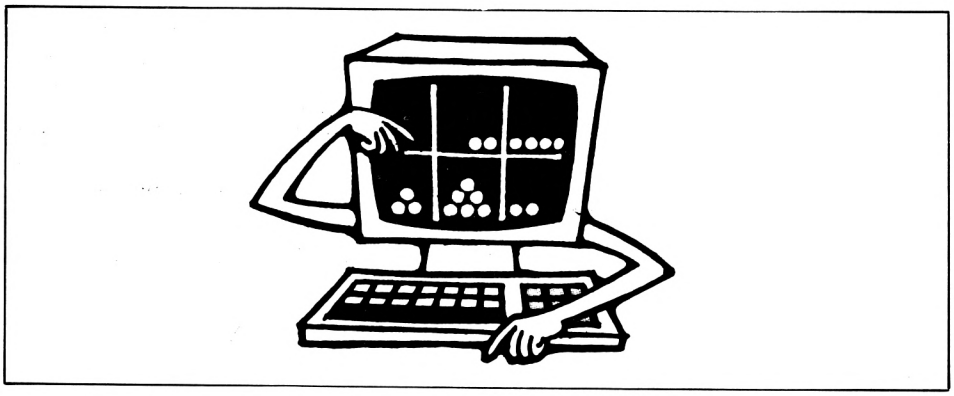
PROGRAMAS COMENTADOS EN BASIC

REVOLTILLO DE LETRAS

El programa que presentamos este mes es un juego, se trata de adivinar una palabra sencilla, que el ordenador nos presentará de forma algo confusa, varias de sus letras, e incluso todas, han sido cambiadas de lugar. Trate de ordenarlas, sólo dispone de una oportunidad.

Aunque las palabras que se proponen son sencillas verá que no resulta nada fácil con ellas, si quiere puede cambiarlas fácilmente por otras o aumentar su número con pequeñas variaciones que explicamos a continuación.

DESCRIPCION DEL PROGRAMA



LINEA	COMENTARIO
20	Utilizamos el modo de pantalla 2 y escribimos un mensaje de presentación del juego.
30	Inicializa dos var, alfanuméricas auxiliares, dándoles como contenido el valor nulo, que en textos equivale a nada o conjunto vacío de caracteres.
40	Restaura el puntero de las líneas DATA, de manera que este apunta hacia la primera palabra de la línea 340.
50	Establece un bucle desde uno hasta un valor aleatorio que como máximo será 20, número de palabras que se han dispuesto en los DATA.
60	Se van leyendo sucesivamente las palabras, comenzando por la primera hasta la que indique el límite superior aleatorio del bucle.
70	Fin del bucle de lectura.
90	Se inicia el bucle de cambio de sitio de las letras. Va desde uno hasta el total de letras que contiene la palabra elegida.
100	H contiene un número aleatorio que como máximo valdrá la longitud de la palabra elegida.
110	Introduce en H$ un fragmento de la palabra, un solo caracter de la misma el que se encuentra en la posición H, calculada en 100.
120	Si el caracter extraido es un * quiere decirse que ya se saco antes, por tanto debe buscar otro, hasta sacar todos.
130	Introduce en la posición H de donde saco el caracter, un as-

terisco, para controlar que caracteres se van extrayendo.

140 Se construye una nueva cadena, a base de añadir los caracteres que se van extrayendo aleatoriamente (por H) y que se contienen en H$, de tal forma que al salir del bucle (línea 150), la var. P1$ contendrá la palabra desordenada.

170 Comienza el juego, se imprime la palabra desordenada en vídeo inverso, para ello se utiliza el caracter de control (24) antes de la palabra a fin de cambiar el color y después para volver a la normalidad.

180 Esta línea y la siguiente producen un efecto de llamada (timbre) para reclamar la atención sobre la palabra que acaba de salir.

190 Sirve para introducir la palabra ordenada por nosotros y se sitúa en la última línea por efecto del LOCATE.

200 Convierte todos los caracteres introducidos en mayúsculas, ya que no es lo mismo para el ordenador una caracter minúsculas o uno solo de sus caracteres, ya no sería igual a la elegida (Todas las palabras en las líneas DATA deben escribirse con mayúsculas7.

220 Control doble, por un lado P2$ = " " significa que hemos apretado ENTER sin introducir ninguna palabra, es decir, no la conocemos, por otro si P2$ (palabra introducida) es distinta (<>) de PALBIS® (palabra original) quiere decir que no hemos acertado. Si se verifica una de las dos condiciones el programa se transfiere a la línea 300, que nos dará un mensaje oportuno seguido de la palabra oculta y de un adorno musical.

230 En caso contrario A C I E R T O!, enhorabuena y música para celebrarlo.

260 Ambos caminos confluyen a esta línea que es un control para seguir jugando. Este control se realiza con INKEY$ de forma que si pulsamos la S o la s comenzará denuevo el juego con otra palabra, con cualquier otro fin, y si no pulsamos nada se mantiene un bucle indefinido, por la línea 270, que retorna a sí misma.

MODIFICACIONES

Para añadir más palabras basta con introducirlas en nuevas líneas DATA no se le olvide contarlas, pues el número total de palabras debe introducirlo en la línea 50 dentro del paréntesis, en lugar de 20, que son las propuestas.

Como sólo se ofrece una oportunidad de acertar, podían establecerse al menos dos, o más, ya que a veces pueden formarse varias palabras con el grupo de caracteres que nos presenta el ordenador y evidentemente el sólo toma por buena, la que tiene programada.

La presentación en pantalla puede hacerla más a su gusto, pues este detalle no se ha cuidado demasiado a fin de facilitar al comprensión del programa.

```
10 REM REVOLTILLO DE LETRAS
20 MODE 1:PRINT "DEBE ENCONTRAR LA PALABRA QUE SE OCULTA EN ESTE
   CONJUNTO DESORDENADO DE LETRAS"
30 P1$="":P2$=""
40 RESTORE
50    FOR K=1 TO  INT(RND*20)+1:'Numero de palabras=20
60 READ PAL$
70 NEXT K
75 PALBIS$=PAL$:N=LEN (PAL$)
80    FOR K=1 TO  N
90 H=INT(RND*N)+1
100 H$=MID$(PAL$,H,1)
110 IF H$="*" THEN 90
120 MID$(PAL$,H,1)="*"
130 P1$=P1$+H$
140 NEXT K
150 IF P1$=P2$ THEN 50
160 LOCATE 13,10: PRINT CHR$(24);" ": P1$;" ";CHR$(24)
162 ENV 2,1,14,1,14,-1,20
163 SOUND 1,140,-1,0,2
170 LOCATE 1,24:  INPUT "CUAL ES LA PALABRA";P2$
180 P2$=UPPER$(P2$)
190 IF P2$="" OR P2$<>PALBIS$ THEN 300
200 LOCATE 10,14: PRINT "A C E R T O !"
205 SOUND 1,90,-1,0,2
206 SOUND 1,100,-1,0,2
210  LOCATE 1,24:PRINT "Quiere jugar otra vez (S/N)"
220 z$=INKEY$: IF z$=""THEN 220
230 IF z$="S" OR z$="s" THEN RUN
240 END
300 LOCATE 6,14: PRINT "LO SIENTO, LA PALABRA ERA ",,CHR$(24);
   PALBIS$;CHR$(24)
302 ENT -2,6,-1,1,1,6,1
304 SOUND 1,150,250,14,0,2
310 GOTO 210
400 DATA MARIPOSA,ESTAMENTO,BAUL,MESA,JARRON
410 DATA DINERO,PEON,ESCALERA,LAPIZ,VENTANA
420 DATA NARIZ,POZO,COCODRILO,SALIDA,PAN
430 DATA SILLA,CAZO,LIBRO,TELA,MANTECA
```

EL LOGO DEL AMSTRAD

V.1. INTRODUCCION

Todos los programas en Logo se llaman procedimientos. Existen lenguajes en los que los procedimientos son pequeños programas que forman parte del programa que los define. Es un modo de trabajo en programación estructurada con el cual se puede "definir", una nueva instrucción en el programa, esa nueva instrucción es justamente el nombre asignado al procedimiento y se puede utilizar como cualquiera de las instrucciones iniciales del lenguaje (que, recordémoslo, en Logo se llaman primitivos). En Logo las cosas se llevan un poco más allá en este sentido porque **todos** los programas son procedimientos. Una vez definido un procedimiento, en una sesión de trabajo con Logo, puede usarse como cualquiera de los primitivos. Pero debemos tener presente que, al concluir la sesión de trabajo, los primitivos serán borrados de la memoria o espacio de trabajo. Más adelante veremos cómo se pueden almacenar en el disco para poder recuperarlos cuando se necesiten.

Resulta de lo anterior que es posible **inventar** todas las órdenes que se quiera en Logo. Por lo tanto una manera muy conveniente de trabajar es la de definir **utilidades** que se puedan usar en cualquier momento sin más que recuperarlas desde el disco. Supongamos, por ejemplo, que deseamos diponer de una instrucción que dibuje círculos del radio deseado. Tal orden no existe como primitivo en Logo. Podemos entonces dedicar un tiempo de nuestro estudio del lenguaje a definir un procedimiento como el deseado. Pero una vez conseguido basta almacenarlo en el disco y recuperarlo y usarlo, como un primitivo más, cada vez que sea necesario. Esto deberíamos te-

nerlo siempre presente al trabajar en Logo, para ganar tiempo y para trabajar estructuradamente.

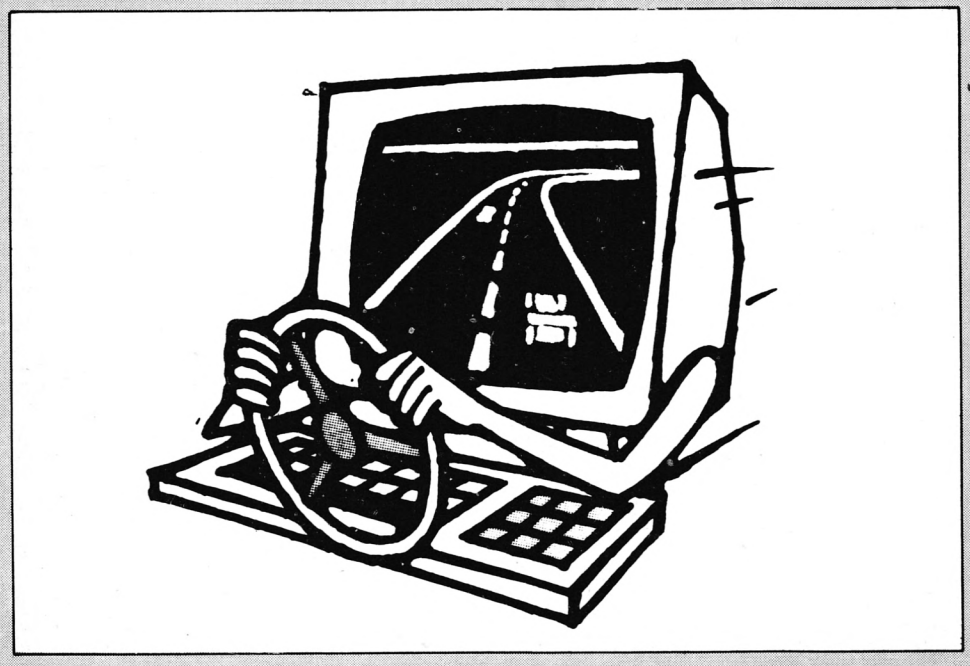
V.2. DEFINICION

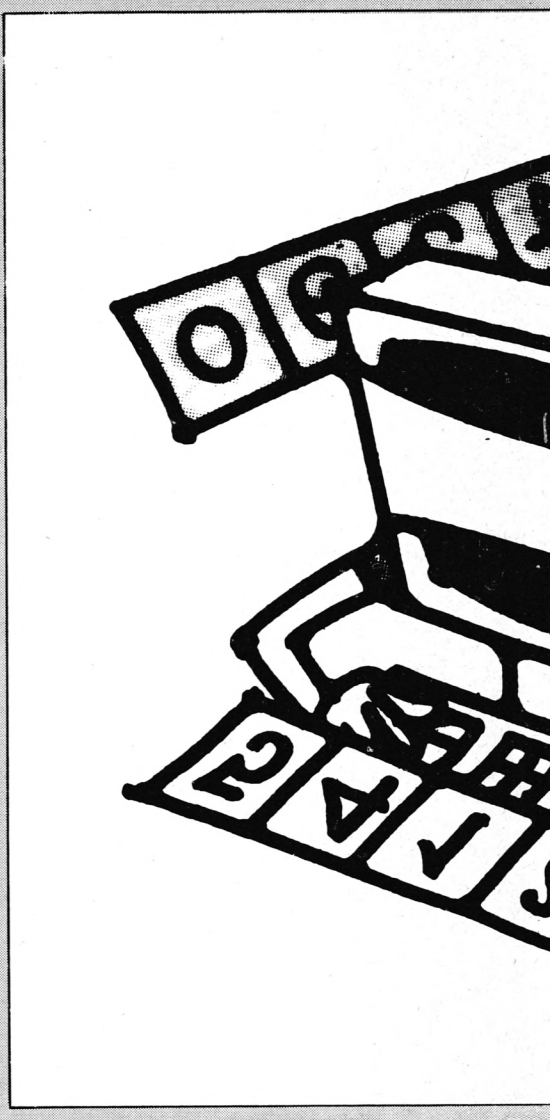
Se puede definir un procedimiento en Dr. Logo de varias maneras. Comenzaremos viendo las formas interactivas y haremos una breve referencia a la forma no interactiva.

Podemos considerar que todo procedimiento consta formalmente de tres partes: el **nombre,** el **cuerpo** y el **final.** Lo primero que ha de hacerse para comenzar la definición de un nuevo procedimiento es decidir su nombre. Los nombres de los procedimientos deben cumplir una serie de requisitos para evitar que al intentar ejecutarlos se produzcan ambigüedades que Dr. Logo no nos dejara seguir adelante con la definición del procedimiento. Por el momento los nombres de los procedimientos deben consistir enuna sola palabra en la que no se puede incluir ningún separador de palabras (excluidos así los caracteres usados en las operaciones aritméticas, por ejemplo), no puede empezar con un número, y si comienza con el caracter "." entonces no puede seguir un número. Además los nombres no pueden coincidir con los de los primitivos. Lo mejor es dar nombres sencillos a los procedimientos, independientemente de que se acepten o no otras posibilidades. De este modo nos permitimos sugerir que comiencen con una letra (para respetar las asignaciones de nombres a objetos en la mayoría de. lenguajes), que usen letras números y el caracter, que no sean muy largos y, sobre todo, que resulten lo más indicadores posible de la acción que realizan. Por supuesto si nuestro deseo no es respetar los consejos anteriores siempre podemos intentar un nombre cualquiera y esperar para ver si es o no rechazado por Dr. Logo.

Ejemplo:

Son nombres válidos de procedimientos:
casa
casa1
casa.1
cuadrado
t. 1. a
(nombre nada aconsejable desde luego)
.'
//a
n''j
''k
?n
&t
Triángulo
Escalera. de. caracol
Son nombres inválidos (mensajes de error obtenidos en línea inferior)
8casa
8casa isn't a name or procedure
i/6
/6 isn't a parameter





t + y
+ y isn't a parameter

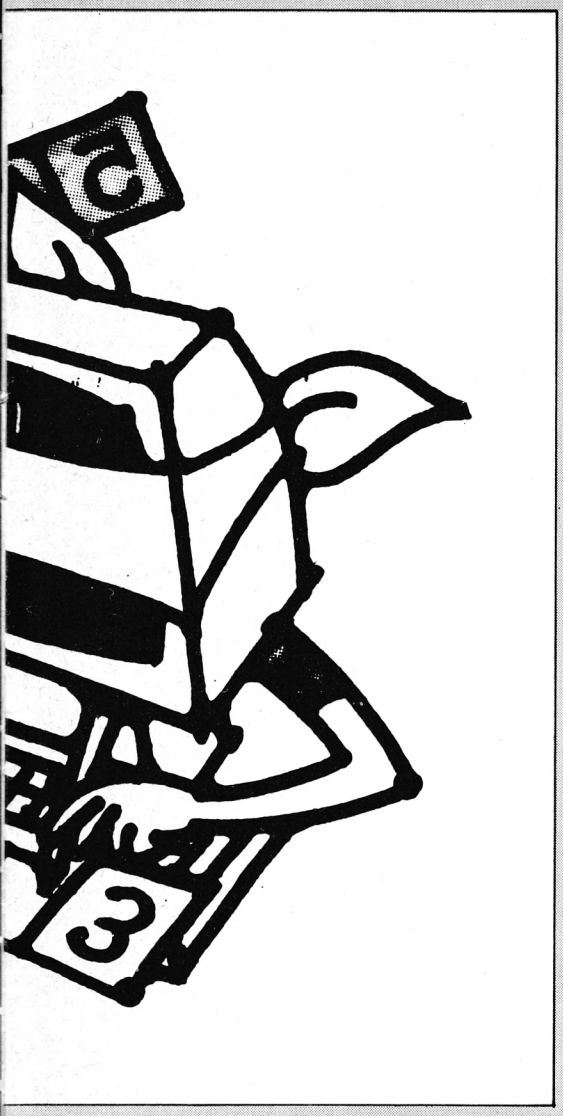
+ k
+ is a primitive

fd
fd is a primitive

aa [ss]
[ss] isn't a parameter

[uno]
uno] isn't a parameter

Aconsejamos pues que los nombres sean sencillos, así como que no se diferencia entre letras mayúsculs y minúsculas (aunque Dr. Logo si diferencia y considera válido el nombre FD aunque fd es un primitivo, así como permite la existencia simultánea de los procedimientos Casa y casa).



Cuando se escriben las órdenes en el teclado para que sean obedecidas de inmediato una tras otra, se dice que se trabaja en modo interactivo o bien que se está en el nivel superior. Para definir un procedimiento se accede a otro modo de trabajo llamado modo definición. Supongamos que el nombre del procedimiento que se va a definir es **cuadrado.** Entonces la orden que sitúa a Dr. Logo en el modo definición es

to cuadrado

donde, naturalmente, la palabra cuadrado habrá de sustituirse cada vez por el nombre elegido para el procedimiento.

Cuando se ejecuta esta orden se abandona el modo interactivo. Esto lo indica Dr. Logo cambiando ul prompt: el modo interactivo es el caracter **?** y en modo definición lo es el caracter >. Todo lo que se escriba en este modo de trabajo pasa a formar parte de cuerpo del procedimiento, es decir, de su definición. Se puede incluir aquí toda clase de instrucciones: priitivos u otros procedimientos. Si se incluye el nombre de otro procedimiento, este deberá existir antes de que se ejecuta el que lo utiliza (de otro modo Logo no sabría ejecutarlo). No debemos, pues, esperar que las órdenes se ejecuten mientras las escribimos en este modo de trabajo: Dr. Logo las ejecutará todas seguidas cuando se llame a ejecución al procedimiento que se define.

En una misma línea (es decir, antes de pulsar "enter" se pueden incluir tantas órdenes como quepan; basta separar cada una de la siguiente por un espacio en blanco). Cuando se llega al final de un renglón sin pulsar "enter", Dr. Logo continúa en la línea inferior de modo que el último caracter de la línea superior se considera inmediatamente precedente del primero de la inferior (no importa pues que se partan así palabras); esta circunstancia la indica Logo situando el caracter **!** al final de la línea superior.

Todo lo que se escriba se hace en modo inserción. La diferencia entre el modo inserción y el modo typeover (sobrescritura) es la siguiente: en modo inserción los nuevos caracteres se insertan entre los ya existentes en el orden en que se escriban; en modo typeover los nuevos caracteres sustituyen a los antiguos. Con las teclas de movimiento del cursor este puede moverse a derecha e izquierda por la línea de trabajo (pero no se puede salir de ella sin pulsar enter). Para borrar el caracter situado a la izlquierda del cursor se usa la tecla rotulada **DEL** y para borrar el caracter sobre el que esta el cursor se usa la secuencia **D** (control-D).

Mientras se este trabajando en modo definición, sólo se puede editar (escribir o modificar) la línea actual. Tras cada pulsación de la tecla "enter" se abandona la línea anterior y se continúa el modo definición en la línea siguiente.

Cuando hayamos terminado de definir el procedimiento (habremos escrito todas las líneas de instrucciones que lo forman) debemos abnadonar este modo de trabajo para regresar al nivel superior (y posiblemente ejecutar el procedimiento para ver si funciona como esperabamos). Esto se hace escribiendo al comienzo de una nueva línea (es decir, inmediatamente a la derecha del caracter > la orden

end

Entonces aparecerá en pantalla el mensaje

cuadrado defined

(en vez de cuadrado aparecerá en cada caso el nombre del procedimiento) y el proceso habrá concluido. A

partir de este momento Dr. Logo conoce una nueva instrucción que puede usarse en la misma forma que cualquier otra. Si, en vez de ello, se pulsa en cualquier momento la tecla **escape** (rotulada ESC) entonces el procedimiento no será definido y aparecerá en pantalla el mensaje **Stopped!**

EJEMPLO V. 2

La siguiente secuencia de órdenes serán las que deben escribirse en el teclado para definir un procedimiento de nombre **cuadrado** y que dibuja un cuadrado de lado 100:

```
? to cuadrado
>repeat 4 [fd 100 rt 90]
>end
?cuadrado defined
```

Obsérvese que en el ejemplo anterior se ha incluido el prompt de Logo en cada línea. En adelante no se incluirá tal cincunstancia dado que el proceso es siempre el mismo.

El siguiente procedimiento podría considerarse como una utilidad: fija el modo de líneas de texto de la pantalla mixta en 2, borra los textos, borra los gráficos y sitúa la pantalla mixta:

```
to útil.1
setsplit 2
ct cs
ss
end
```

que podría escribirse en una sola línea:

```
to útil. 1
setsplit 2 ct cs ss
end
```

El siguiente procedimiento usa a los dos anteriores (util.1 y cuadrado) para dibujar un cuadrado y poner el nombre de la figura al pie de la página.

```
to cuadrado. b
util. 1
cuadrado
pr [cuadrado de lado 100]
end
```

EJEMPLO V. 3.

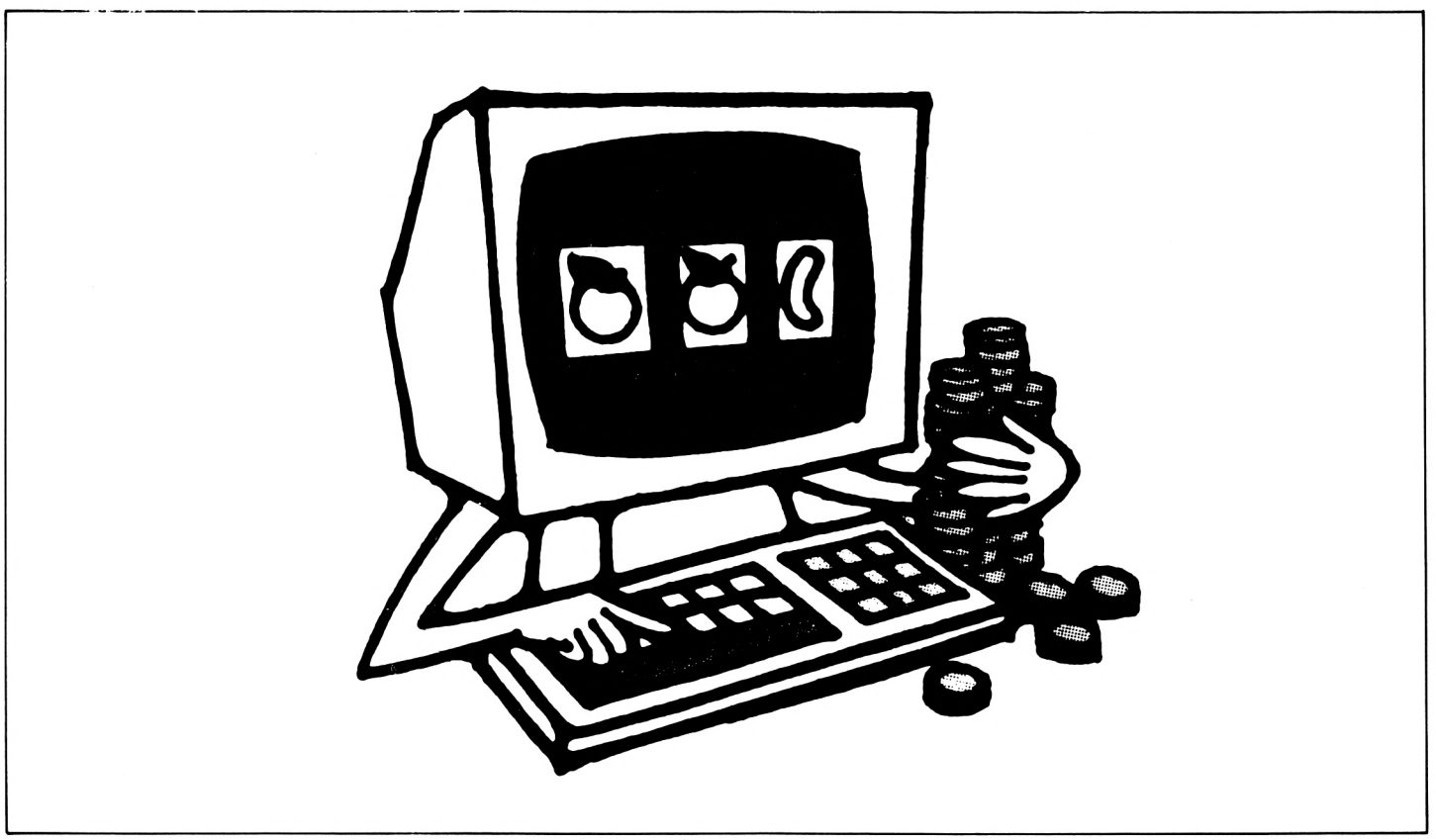
Los polígonos regulares que se estudiaron en el capítulo III pueden obtenerse como procedimientos de la siguiente forma:

```
(como ejemplo III.3.1)
to pentágono
cs ht
repeat 5 (fd 120 rt 360/5)
end
(como ejemplo III.3.2)
to hexágono
clean st
repeat 6 [fd 90 it 360/6]
end
(como ejemplo III.3.3.)
to cuadrados
cs ht
repeat 4 [bk 50 rt 90]
repeat 4 [fd 50 it 90]
end
```

(En todos estos ejemplos se han usado los enunciados indicados del capítulo III y la versión ya estudiada de las órdenes en el ejemplo III.4).

Si se define un procedimiento con el mismo nombre que uno ya existente, la antigua versión desaparecerá de la memoria. Esto no puede hacerse con los nombres de primitivos, como ya hemos visto.

Sucede con mucha frecuencia que lo que deseamos no es definir un nuevo procedimiento sino modificar uno ya definido. Según acabamos de decir, una posibilidad es definirlo de nuevo. Pero si los cambios son po-



cos y el listado del procedimiento largo, este es un método poco económico. Existe la posibilidad de acceder a otro modo de trabajo en el cual se pueden hacer exactamente los cambios necesarios con gran rapidez, dejando el resto como estaba. Este modo de trabajo se llama modo edición y se accede a el con la orden:

end "nombre

siendo **nombre** el del procedimiento que se desea editar. Cuando se ejecuta esta orden se accede a una pantalla en la que aparece toda la definición del procedimiento (o la parte que quepa si es muy largo), y en la última línea aparece a indicación **Edit.** Usando las mismas técnicas que en modo definición podemos editar cada una de las líneas. Además ahora las teclas de movimiento vertical del cursor permiten el traslado de este a través de las líneas que forman el procedimiento (esto es una mejora respecto del modo definición). Si se desea insertar una línea en un punto determinado, basta situar allí el cursor y pulsar entonces la tecla "enter". Entonces los caracteres situados a su izquierda continuarán en la misma línea, y los situados a su derecha pasarán a la línea inferior, que será una línea nueva (de modo que las líneas nferiores se desplazarán todas un lugar hacia abajo(. Si esta acción se lleva a cabo con e cursor situado al final de una línea, el efecto será el de la creación de una línea en blanco inmediatamente debajo de la línea actual.

Para abandonar el modo edición, se puede usar una de las dos siguientes posibilidades. Si se pulsa la tecla **escape** (rotulada ESC), entonces se regresa al nivel superior pero de modo que las acciones que se hayan realizado en el editor **no** serán tenidos en cuenta. Es como si nos arrepentimos de todo lo hecho en el editor. Esto se indica con el mensaje **Stopped!** Si lo que queremos es que todo lo que hayamos hecho en el editor **si** sea tenido en cuenta, entonces se pulsa la tecla **copy** para abandonarlo. El mensaje obtenido en este caso es como el obtenido cuando se abandona el modo definición.

EJEMPLO V.6.

Vemos a ver los procedimientos necesarios para dibujar un cuadrado

dividido en cuatro cuadrados iguales. La técnica que vamos a usar es la siguiente: el procedimiento llamado **cuadrado** dibuja un cuadrado de lado 70 a partir del punto en que se encuentre la tortuga al empezar su ejecución, y la deja precisamente en este mismo sitio y mirando hacia el mismo punto. El procedimiento **figura. 1** repite cuatro veces el procedimiento cuadrado seguido por un giro de 90 grados de la tortuga.

```
to cuadrado
repeat 5 [fd 70 rt 90]
end

to figura 1
repeat 4 (cuadrado rt 90)
```

EJEMPLO V.7.

No es casualidad que se gire **4** veces un ángulo de **90** grados: se tiene que 4 × 90 = 360 y 360 grados representa una vuelta completa de la tortuga. De modo que se puede actuar de modo similar para una cantidad distinta de cuadrados y números de grados que debemos girar la tortuga. Así, como 360 / 10 = 36, un procedimiento análogo al anterior pero formado por 10 cuadrados sería (usando el procedimiento cuadrado del ejemplo anterior):

```
to figura 2
repeat 10 (cuadrado rt 36)
end
```

en el que se han girado (a la derecha) 36 grados.

Existe todavía otra manera para definir procedimientos. Es una manera no interactiva mediante la cual es posible incluso que un programa escrito en Dr. Logo se automodifique. Como es una técnica avanzada queda su estudio para más adelante.

PROGRAMAS EN LOGO

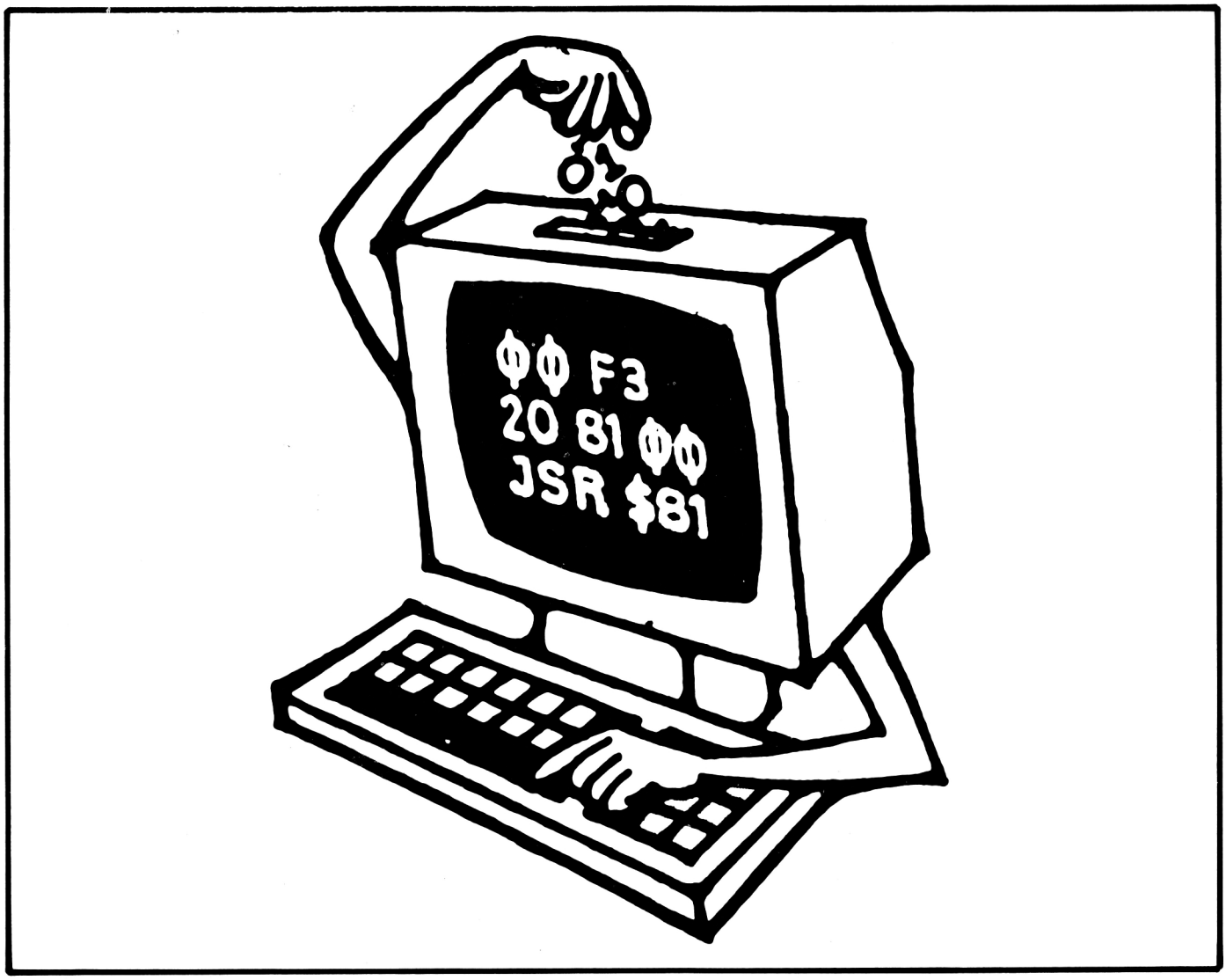
PREG

El programa de este número pretende ser un primer ejemplo de un programa interactivo en el que, por ejemplo a un niño, se le pueden ir presentando gráficos elementos construidos con la tortuga y haciendo algunas preguntas sobre ellos.

El programa es muy elemental y presenta en pantalla las indicaciones para ejecutarlo. Hay que decir, no obstante que como el nombre del programa principal es **preg** habrá de escribirse esta palabra para que comience la ejecución.

Ha de hacerse la advertencia de que este programa puede escribirse usando menos procedimientos y sentencias pero de construcción más potente y compleja. Tiempo habrá para economizar escritura en bien del razonamiento.

En el siguiente listado se han separado con una línea los procedimientos de que consta el programa para mayor claridad (no forman parte del listado). Igualmente debe tenerse en cuenta que se han incluido comentarios a las instrucciones del programa encerrándolos entre la secuencias de caracteres (* y *) como se hace en Pascal. **Ninguno de estos caracteres forma parte del programa.** Deben pues ignorarse todas las secuencias de caracteres encerradas entre (* y *) al escribir el programa en el ordenador.

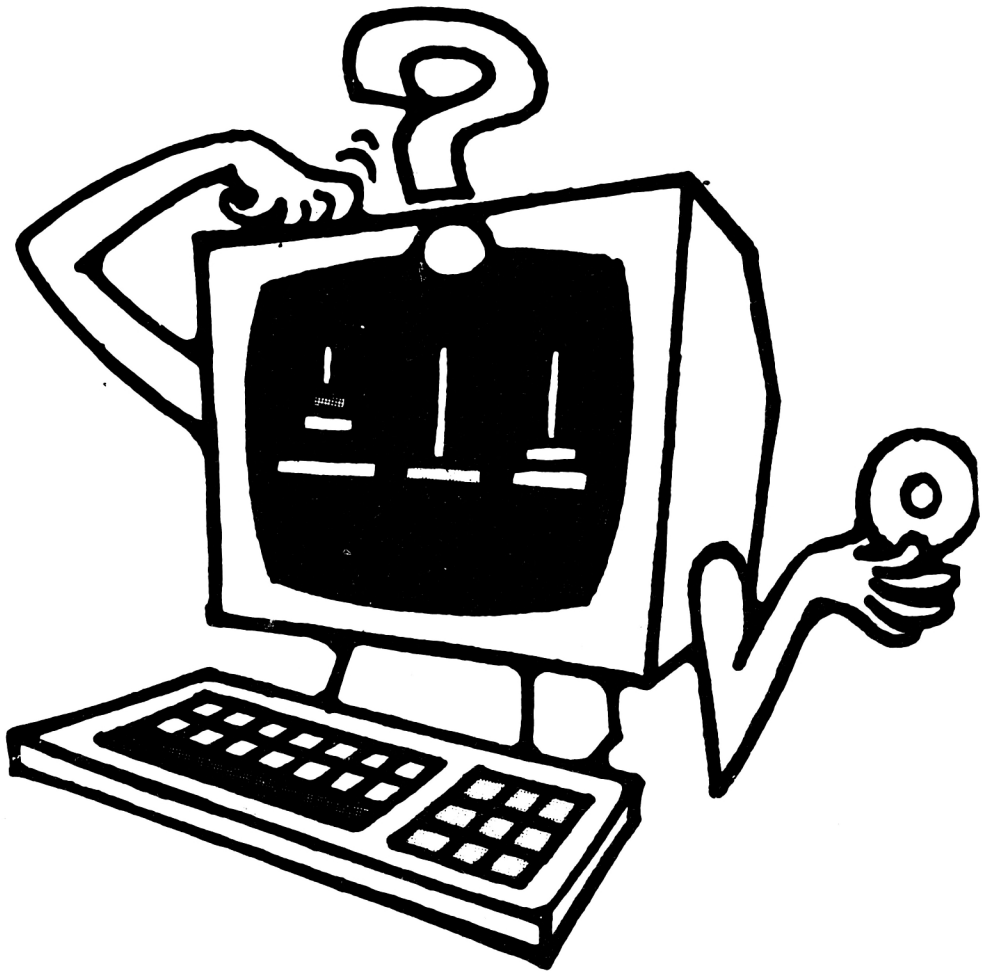


```
(* almacena en la variable "seguir el primer caracter que se teclee *)
ct
(* borra los textos *)
end
-----------------------------------
to cuadrado
ss ht
(* instala la pantalla mixta y esconde a la tortuga *)
repeat 4 [fd 150 rt 90]
(* dibuja un cuadrado *)
type [Como se llama este poligono?]
make "resp rl
(* almacena en la variable "resp la lista de palabras que se teclee
hasta la pulsacion de "enter".
if :resp=[cuadrado] [bien][mal "cuadrado]
(* Si el contenido de la variable "resp es la lista que solo contiene
la palabra cuadrado, entonces se ejecuta el procedimiento "bien"; de
otro modo se ejecuta el procedimiento "mal" con el input consistente
en la palabra "cuadrado" *)
areacuad
(* llamada a procedimiento (que se ocupa del area del cuadrado *)
end
-----------------------------------
to areacuad
make "lado random 10
(* asigna a la variable "lado un numero elegido por Dr. Logo al azar
entre 0 y 9 *)
pr []
type [Si su lado mide \ ] type :lado
type [\ cual es su area?]
make "area first rl
(* asigna a la variable "area el primer elemento de la lista que se
escriba en el teclado *)
if :area=:lado*:lado [bien][malarea :lado*:lado]
(* si el contenido de la variable :area es igual al resultado de
multiplicar el contenido de la variable :lado por si mismo entonces se
ejecuta el procedimiento "bien"; de otro modo se ejecuta el
procedimiento llamado "malarea" con el input :lado*:lado *)
end
-----------------------------------
to rectangulo
cs ss ct
(* borra graficos,instala pantalla mixta y borra textos *)
rt 90 pu bk 100 pd
(* situa a la tortuga en lugar conveniente para comenzar los graficos.
Para ello se gira 90 grados a la derecha, se levanta la pluma para que
no pinte al trasladarse, se hace que retroceda 100 pasos y se baja la
pluma de nuevo *)
repeat 2 [fd 200 rt 90 fd 100 rt 90]
(* dibujo de un rectangulo *)
type [Como se llama este poligono?]
make "resp rl
if :resp=[rectangulo] [bien][mal "rectangulo]
arearec
end
```

```
-----------------------------------
to arearec
make "base random 10 make "altura random 10
pr []
type [Si la base mide \ ] type :base
type [\ y la altura \ ] type :altura
type [\ cual es su area ?]
make "area first rl
if :area=:base*:altura [bien][malarea :base*:altura]
end
-----------------------------------
to triangulo
cs ss ct
repeat 3 [fd 200 rt 120]
(* dibuja un triangulo *)
type [Como se llama este poligono?]
make "resp rl
if :resp=[triangulo] [bien][mal "triangulo]
areatri
end
-----------------------------------
to areatri
make "base random 10 make "altura random 10
(type [Si la base mide] :base [y la altura] :altura [cual es su area?])
(* aunque se podia haber repetido la secuencia de instrucciones
analogas de los procedimientos areacuad y arearec, a modo de ejemplo
se ha usado en este procedimiento una sola sentencia type un tanto mas
compleja. *)
make "area first rl
if :area=:base*:altura/2 [bien][malarea :base*:altura/2]
end
-----------------------------------
to bien
make "aciertos :aciertos+1
(* asigna a la variable "aciertos una unidad mas que su valor
inmediatamente precedente *)
pr [BIEN !!]
type [tecla...]
make "seguir rc
end
-----------------------------------
to mal :palabra
make "fallos :fallos+1
type [MAL...La respuesta es \ ] pr :palabra
type [tecla...]
make "seguir rc
end
-----------------------------------
to malarea :n
pr [MAL...]
make "fallos :fallos+1
type [El area es \ ] pr :n
type [tecla...]
make "seguir rc
end
```

EL PROGRAMA TECNICO DEL MES

TECLAS DEFINIDAS



Una enorme ventaja, a veces poco aprovechada de nuestro Amstrad, consiste en la posibilidad de construir un teclado a medida de nuestras necesidades.

El programa que presentamos a continuación utiliza estas posibilidades, seguramente le será de gran utilidad y cuando menos, supondrá una comodidad al teclear sus programas, sobre todo si no es usted un gran mecanógrafo.

Si conoce otros ordenadores, habrá observado que en unos las instrucciones del Basic, comando, etc., se obtienen con sólo pulsar una tecla, como el Spectrum, sin embargo en otros, todas las palabras deben deletrearse. Su Amstrad le permite ambas cosas, deletrear las órdenes o introducirlas con una sola pulsación.

Para el primer caso, letra a letra, no hay dificultades, el uso es exactamente igual al de una máquina de escribir.

En el segundo caso hemos de definir en cada letra elegida la palabra deseada, por ejemplo hacer corresponder a la letra "ele", la palabra LIST. Para ello disponemos de dos intrucciones KEY y KEY DEF, veamos como usarlas correctamente.

Disponemos de 32 teclas programables, que llamaremos "teclas de función". Los valores ASCII que pueden ser expendidos son los comprendidos entre 128 y 159. Si observa su manual, en el apartado "Para su referencia...", verá que el teclado numérico de la derecha, le corresponden los valores 128 a 140.

Si no deseamos modificar estas teclas numéricas, a fin de que conserven su utilidad para manejar números de forma más cómoda, nos encontramos con que disponemos de los códigos 141 a 159 como "libres".

Cualquiera de estos códigos puede ser expendido hasta un máximo de 32 caracteres, pero teniendo en cuenta que el número total de caracteres expendidos no puede sobrepasa los 120.

1. NUMERO DE LA TECLA... Este número es que el que figura a la derecha de su ordenador, si se trata de un CPC6128, en cualquier caso los encontrará en su manual, en el apartado "Para su referencia" (Pág. 7/23 del CPC664 y Apéndice III, pág. 16 para el CPC464).

2. REPETICION... El valor 1 permite la repetición y el 0 la impide.

3. CARACTER NORMAL... Caracter que se obtiene al pulsar la tecla sin actuar simultáneamente sobre Shift o Control.

4. CARACTER CON SHIFT... Caracter obtenido al pulsar simultáneamente SHIFT + Tecla.

5. CARACTER CON CONTROL... Caracter obtenido al pulsar simultáneamente CONTROL + Tecla.

Veamos con un ejemplo como se utilizan estos parámetros. Si deseamos introducir la palabra PRINT en la letra "P", debemos introducir.

KEY DEF 27 , 0 , 112 , 80 , 142

Estos parámetros tienen el siguiente significado:

1. Valor = 27. Que corresponde a la tecla P

2. Valor = 0. No producirá repetición (No resulta útil obtener muchos PRINT seguidos, queremos sólo uno).

3. Valor = 112. Es el código ASCII de la letra "p" minúscula (ver tabla ASCII en el manual (Pág. 7/8 del CPC6128 y 664, Apéndice III Pág. 1 del 464).

4. Valor = 80. Es el código ASCII de la "P" mayúscula.

5. Valor = 142. Es el código ASCII que se obtendrá presionando CONTROL + P y que hemos seleccionado en el intervalo 141 a 159 disponible.

Una vez hecho esto sólo nos queda enseñarle al ordenador que al código 142 le corresponde la palabra "PRINT", esto se realiza añadiendo.,

KEY 142, "PRINT"

Y ya está, cada vez que pulsemos la tecla de CONTROL y simultáneamente la tecla "P", obtendremos en la pantalla la palabra PRINT. Como hemos visto cada definición se forma con dos órdenes distintas KEY, DEF y KEY.

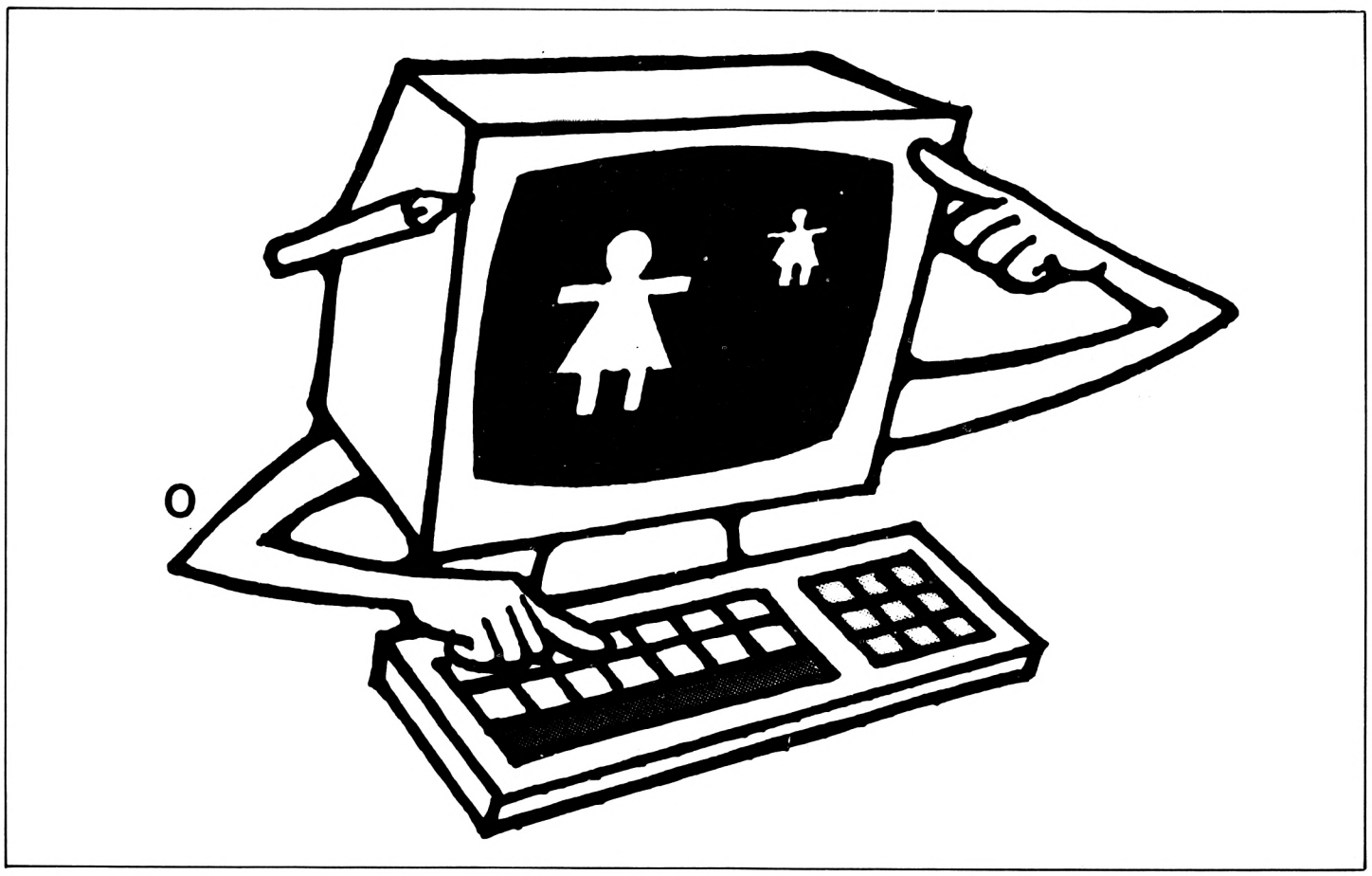
El programa que incluimos a continuación es una muestra de las enormes posibilidades que estas instrucciones nos brindan, puede utilizarlo como esta, o variarlo según sus gustos o necesidades.

LISTADO

Para su comodidad le indicamos las teclas programadas y sus textos asociados.

L = LIST
S = SAVE
P = PRINT
L = LOAD

G = GOTO
T = TO
R = RUN (+ ENTER)
E = EDIT
D = LOCATE
H = THEN
D = DATA



F = FOR
N = NEXT
I = INPUT
C = CHR$
Y = IF
U = USING

Como habrá observado algunas palabras llevan incluidos otros caracteres como " y (en incluso la pulsación de ENTER. Pude ver como se ha realizado, por el propio listado, teniendo en cuenta los códigos ASCII que les corresponden.

" 34
ENTER ... 13
(..... Dentro del propio texto

Para utilizar el programa le sugerimos que una vez tecleado y comprobado y antes de hacer RUN, lo salve en cinta o disco, ya que la última línea supone una destrucción del mismo. Esto permite que al iniciar un trabajo podamos cargar primero este programa con RUN "nombre" y automáticamente el teclado estra libre y el ordenador "limpio y listo" para comenzar a programar.

Puede construir un juego de programas de este tipo, dedicados a trabajos específicos, programación, manejo de discos, etc., de manera que le permita utilizar en cada momento el más idóneo.

```
5 REM        TECLAS DEFINIDAS.
             TECLADO ALFABETICO

10 KEY DEF 36,0,108,76,141
15 KEY 141,"LIST "
20 KEY DEF 27,0,11?,80,142
25 KEY 142," PRINT "
30 KEY DEF 60,0,115,83,143
35 KEY 143,"SAVE "+CHR$(34)
40 KEY DEF 69,0,97,65,144
45 KEY 144,"LOAD "+CHR$(34)
50 KEY DEF 52,0,103,71,145
55 KEY 145," GOTO "
60 KEY DEF 53,0,102,70,146
65 KEY 146," FOR "
70 KEY DEF 51,0,116,84,147
75 KEY 147," TO "
80 KEY DEF 46,0,110,78,148
85 KEY 148," NEXT "
90 KEY DEF 50,0,114,82,149
95 KEY 149,"RUN "+CHR$(13)
100 KEY DEF 35,0,105,73,150
105 KEY 150," INPUT "
110 KEY DEF 58,0,101,69,151
115 KEY 151,"EDIT "
120  KEY DEF 62,0,99,67,152
125  KEY 152,"CHR$("
130  KEY DEF 34,0,111,79,153
135  KEY 153," LOCATE "
140  KEY DEF 43,0,121,89,154
145  KEY 154," IF "
150  KEY DEF 51,0,116,84,155
155  KEY 155," THEN "
160  KEY DEF 42,0,117,85,156
165  KEY 156," USING "
170  KEY DEF 61,0,100,68,157
175 KEY 157," DATA "
300 CLS: DELETE
```

AMSTRAD
CPC 464, 664 y 6128
PROGRAMACION ESTRUCTURADA

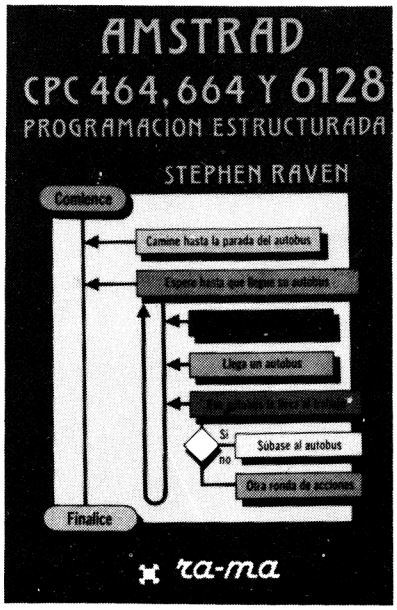
Cuando este libro salía para la imprenta el Amstrad lanzaba el micro CPC 6128 doméstico. Sus ventajas sobre el 664 es que está equipado con el doble de memoria de usuario RAM, un teclado diferente (que a mi gusto no es tan bueno como el del 664) y una nueva versión del sistema operativo industrial CP/M que le permite funcionar con un surtido más amplio de programas de gestión.

La buena nueva es que los programas escritos en BASIC para el 464 y el 664 funcionarán sin alteración en el 6128. Eso significa que debieran «currar» todos los programas basados en el disco 664 y el 99,5% de los programas en cinca 464. **Eso significa también que puedes aprovechar todas las pistas y trucos de programación de este libro para aprender a usar tu nuevo 6128.**

Aunque el 6128 se suministra con 128K de RAM, sólo 68K están disponibles para los programas en BASIC. Eso es porque el BASIC del 6128 se diseñó originalmente para el 464 y el 664 que sólo tienen disponibles 64K. Para ayudarte a solventar este problema, Amstrad ha incluido algunos comandos BASIC extra que pueden implantarse desde el disco. Con ellos se te permite usar el «repuesto» de 64K de RAM bien sea para almacenar imágenes de pantalla adicionales o bien como sistema rápido de archivo. Por el momento, no puedes usar esos 64K de repuesto para contener programas en BASIC.

Todos estos comandos extra quedan implantados en el sistema haciendo que se ejecute el programa en código máquina «Bankmanager» suministrado en tu disquete. Si quieres utilizar esa RAM de repuesto para almacenar en ella imágenes de pantalla, el Bankmanager (gestor de las bancadas de memoria) te proporciona dos comandos adicionales.

| SCREENCOPY y | SCREENSWAP que te permiten conservar hasta 4 imágenes de pantalla en la memoria adicional y luego **canjearlas** sobre la pantalla. Eso puede ser útil para juegos y animación en los que se prepara la nueva pantalla en la «retaguardia» y luego se pasa a «vanguardia» cuando es necesario. Cada imagen de pantalla se identifica con un número de bloque del 1 al 5. Para que se proyecte en el monitor es necesario moverla al bloque 1.



| SCREENCOPY hace que se **copie** toda una imagen de pantalla desde un bloque origen hasta un bloque destino. El contenido previo del bloque destino se pierde siempre al escribir encima. Por ejemplo | SCREENSWAP, 1, 4 pondría la imagen corriente en el bloque 4 y el contenido del bloque 4 pasaría a ser el proyectado en pantalla. Un ulterior | SCREENSWAP haría que las imágenes retornaran a la situación primitiva.

Si decides que quieres usar el repuesto de 64K de RAM para almacenar datos en lugar de imágenes en pantalla el «gestor de bancada» te proporciona 4 comandos BASIC adicionales:

| BANKOPEN permite que se **abra** la bancada especificando cuantos caracteres habrá en cada registro. La máxima longitud es de 255 caracteres. | BANKWRITE hace que se **escriba** un registro, | BANKREAD te permite la **lectura** de un registro y | BANKFIND hace que se **halle** un registro que concuerde con un literal dado y devuelve el número identificativo del registro de manera que puedas efectuar una lectura posterior para conocer los datos registrados.

Todos los comandos aparte del de apertura de bancada necesitan especificar una variable entera que en ella se contenga el «código de estado» y una variable literal para reseñar los datos cuya lectura o escritura se va efectuar.

P.V.P. 1.700 ptas.

NUESTRO PRÓXIMO NÚMERO

AMSTRAD EDUCATIVO

N.º 5

EL AMSTRAD Y EL CP/M

Ficheros en el AMSTRAD

Basic del AMSTRAD

Fichas del AMSTRAD

Programando en BASIC

Logo del AMSTRAD

Programa en LOGO

El Programa técnico del mes

Sound-on-Sound

SUS MEJORES RECUERDOS

CURSO DE BASIC + MICROORDENADORES

prácticas con...

Microordenador ZX SPECTRUM

Microordenador COMMODORE

Microordenadores AMSTRAD, MSX, PC

Para saber cómo hablar con los ordenadores

El Curso CEAC a Distancia, BASIC + Microordenadores, le va a introducir paso a paso, con un cuidado método, en uno de los temas más apasionantes de nuestros días: **la programación de ordenadores.**

Al aprender PRACTICANDO desde un principio a programar BASIC, lenguaje diseñado especialmente para dar los primeros pasos en programación, estará sentando las bases para el estudio de cualquier otro lenguaje de alto nivel.

Curso CEAC de BASIC + Microordenadores: un diálogo permanente con el ordenador.

Otros Cursos:
- Introducción a la Informática
- Electrónica (con experimentos)
- Contabilidad
- Fotografía
- Curso de Video
- Decoración

CEAC

CENTRO DE ENSEÑANZA A DISTANCIA AUTORIZADO POR EL MINISTERIO DE EDUCACION Y CIENCIA N.º 8039185
(BOLETIN OFICIAL DEL ESTADO 3-6-83)
Aragón, 472 (Dpto.) 08013 Barcelona
Tel.: (93) 245 33 06

ESTAS ENSEÑANZAS SE AJUSTAN AL ART. 35 DEL DECRETO 707/1976 Y A LA ORDEN MINISTERIAL DE 5/2/1979

Actúe ahora en su propio beneficio y pídanos información.

GRATUITAMENTE

Sí, deseo recibir a la mayor brevedad posible información sobre el Curso de: _____

_____

Nombre y apellidos _____ Edad ____

Domicilio _____

_____ Nº ____ Piso ____ Pta. ____ Tel. _____

C. Postal _____ Población _____

_____ Provincia _____

Profesión _____

CEAC. Aragón, 472
(Dpto.) 08013 Barcelona

o llame... (93) 245 33 06 de Barcelona