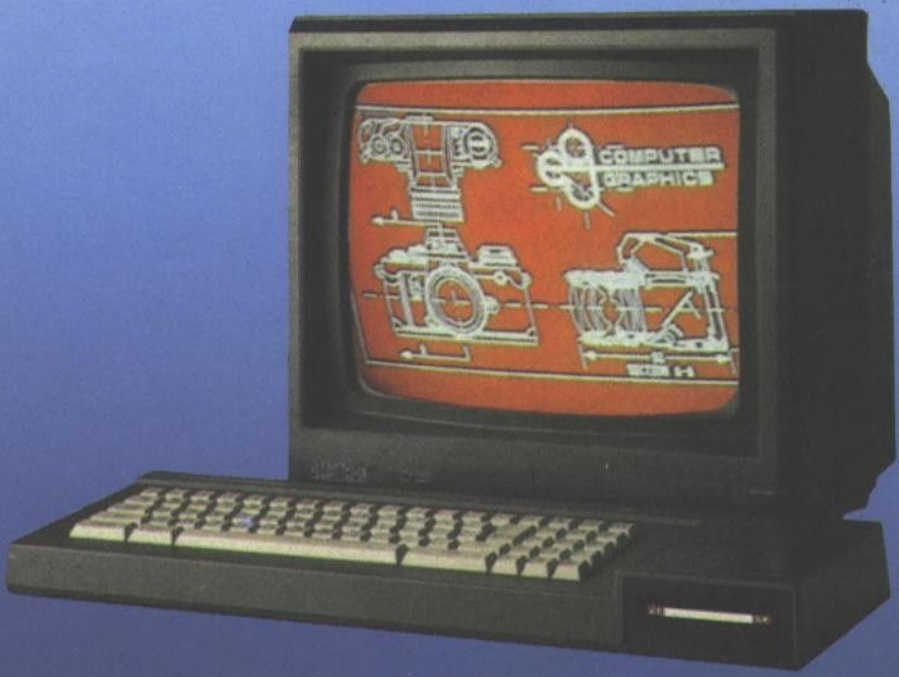
El libro del AMSTRAD



I. Ramón
P. Buera
V. Trigo

PARANINFO

El libro del AMSTRAD

CPC 464 • CPC 472 • CPC 664 • CPC 6128

Ignacio RAMON FERRER
Pedro BUERA PEREZ
Vicente TRIGO ARANDA

El libro del AMSTRAD

CPC 464 • CPC 472 • CPC 664 • CPC 6128

1986

PARANINFO S.A.

MADRID

IMPRESO EN ESPAÑA
PRINTED IN SPAIN

ISBN: 84-283-1486-1

Depósito Legal: M. 24.238.—1986

PARANINFO S.A. Magallanes, 25 - 28015 MADRID (07301/36/86)

Artes Gráficas Benzal, S. A. - Virtudes, 7 - 28010 MADRID

Indice de Materias

Prólogo

Con este libro pretendemos que el lector tenga la oportunidad de conocer exhaustivamente el lenguaje BASIC del ordenador AMSTRAD en los modelos:

CPC 464
CPC 472
CPC 664
CPC 6128

Es importante resaltar que este volumen no es una adaptación de los modelos anteriores al nuevo CPC 6128, sino que se ha escrito para todos ellos, indicando y desarrollando las particularidades de cada uno.

Hemos intentado que cualquier persona, aunque no tenga conocimientos de dicho lenguaje, pueda seguir fácilmente el desarrollo de todos los capítulos. Para lograrlo se van introduciendo los diferentes conceptos poco a poco hasta llegar a las cuestiones más complejas.

Gran cantidad de ejemplos y ejercicios resueltos acompañan a los desarrollos teóricos y, para afianzar todavía más los conocimientos adquiridos, al final de la mayoría de los capítulos se proponen una serie de programas para realizar, cuyas soluciones se encuentran en las páginas finales del libro.

LOS AUTORES

1

El Teclado

El teclado es un dispositivo de entrada mediante el cual podemos comunicarnos con el ordenador. En el caso de nuestro AMSTRAD, está dividido en dos bloques;

- El bloque situado a la izquierda, de gran semejanza con el teclado de una máquina de escribir.
- El bloque situado a la derecha, formado por tres columnas de cinco teclas cada una, que constituyen el llamado teclado numérico,y las teclas de movimiento del cursor.

En el primer bloque nos encontramos con dos tipos de teclas;

- **Teclas de caracteres;** que al presionarlas nos presentan en pantalla tres tipos de caracteres:

  Alfabéticos: las letras del alfabeto, tanto mayúsculas como minúsculas.

  Numéricos: los dígitos de 0 al 9.

  Especiales, como por ejemplo; !, , ?, ¿, $,....

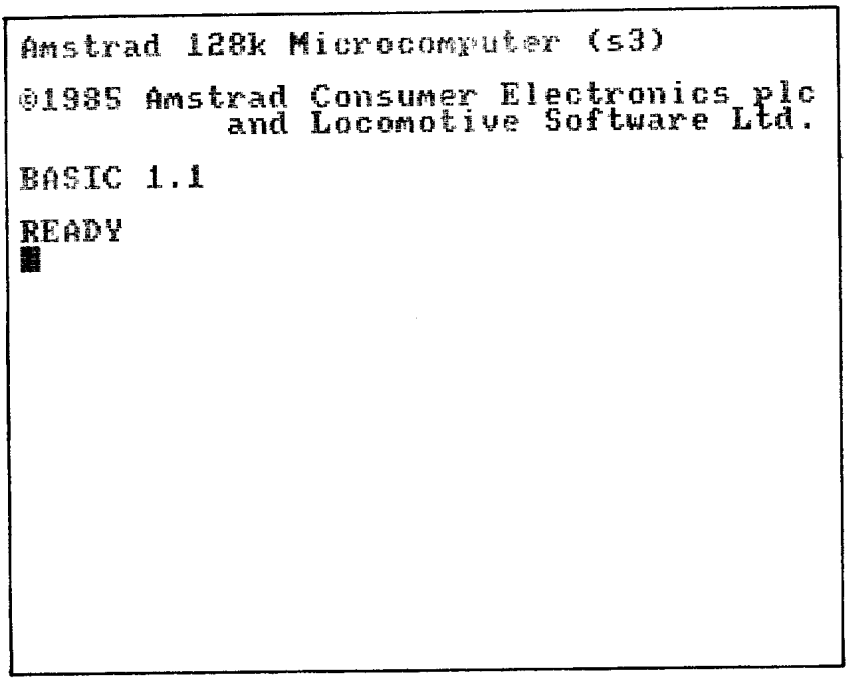
- **Teclas de control,** denominadas de diferente forma según se trate del teclado español o del teclado inglés.

español	**inglés**
FIJA MAYS	CAPS LOCK
MAYS	SHIFT
CONTROL	CTRL
INTRO	ENTER
BORR	DEL
CLR	CLR
COPIA	COPY

Además, en el modelo CPC 6128, encontraremos la tecla "RETURN" tanto en el teclado español como en el inglés.

MANEJANDO EL TECLADO

Al conectar el ordenador, se tiene la imagen:



observar que al final del mensaje de bienvenida, también aparece en la pantalla un cuadradito del tamaño de un carácter y al que llamaremos "cursor".

Si presionamos una tecla de las que llevan impreso un carácter alfabético, veremos que éste ha aparecido, en minúscula, en la pantalla y que además el cursor se ha colocado a su derecha señalando la posición donde irá el siguiente carácter a imprimir. Presionando la barra espaciadora se deja un espacio en blanco.

Para escribir con mayúsculas se mantendrá presionada una de las dos teclas "MAYS" presentes en nuestro teclado y, simultáneamente, la tecla deseada. Este efecto no es permanente por lo que al dejar de presionar "MAYS" se volverá a escribir con minúsculas. Para fijar las mayúsculas se debe presionar la tecla:

FIJA
MAYS

Si se vuelve a presionar, se escribirá de nuevo en minúsculas.

Hay teclas que llevan impresos dos caracteres sobre ellas, por ejemplo;

```
!   "   $   &
1   2   4   6
```

En todos casos el carácter inferior se obtiene cuando se presiona la tecla sin más y el carácter superior si se presiona simultáneamente la tecla "MAYS".

Si queremos escribir en mayúsculas y además con los signos superiores, debemos mantener pulsada la tecla CONTROL a la vez que pulsamos "FIJA MAYS" una sola vez.

Observar que, al completar una línea, el cursor salta automáticamente al comienzo de la línea siguiente.

Movimientos del cursor

La posición del cursor, en la pantalla, puede modificarse con ayuda de las teclas situadas en la parte derecha del teclado y que van marcadas con flechas.

Por ejemplo, si tenemos la pantalla inicial, y pulsamos tres veces la tecla ↓ y cinco veces la tecla →, al marcar la A se tendrá:

```
Amstrad 128k Microcomputer (s3)

©1985 Amstrad Consumer Electronics plc
          and Locomotive Software Ltd.

BASIC 1.1

READY


     A
```

Las teclas de caracteres y las teclas de movimiento del cursor son repetitivas; esto significa que manteniendolas presionadas su acción se consigue de nuevo.

Teclas "BORR" y "CLR"

Se emplean para corregir errores.

"BORR" borra el carácter situado a la izquierda del cursor.

"CLR" borra el carácter situado en la posición del cursor.

Supongamos que, pretendiendo escribir el nombre propio "Juan", hemos escrito:

```
Juam
```

Para corregir el error cometido, debemos colocar en el lugar que ocupa la "m", una "n". Podemos hacerlo de dos formas distintas:

1) Pulsando la tecla "BORR" borraremos la m, y a continuación escribiremos la "n" en su lugar.
2) O bien, movemos un lugar a la izquierda el cursor hasta situarlo encima de la "m". Al pulsar CLR ésta letra se borra y escribimos la letra correcta.

EJERCICIO 1

Vamos a corregir los errores cometidos al teclear la frase siguiente:

```
Ahora,unavez que see ha procedado
```

en lugar de:

```
Ahora,una vez que se ha procedido
```

Primero separaremos las dos palabras que aparecen juntas en "unavez". Para ello colocaremos el cursor encina de la "v" y presionaremos la barra espaciadora.

En segundo lugar habrá que borrar una "e" de "see". Esto se puede conseguir de dos maneras:

1) Colocando el cursor inmediatamente a la derecha de "see" y con "BORR" borrar la última letra.
2) Situando el cursor sobre cualquiera de las dos letras repetidas y pulsando "CLR".

Y, por fin, borraremos la "a" de procedado, siguiendo cualquiera de los dos procedimientos del apartado anterior. Una vez hecho esto el cursor habrá quedado sobre la "d" y ya sólo tendremos que teclear la letra que falta, la "i".

Teclado numérico

A la hora de trabajar con números se puede, indistintamente, usar la parte superior del teclado principal o bien el teclado numérico.

Además el teclado numérico tendrá otra misión que veremos más adelante.

NOTAS

- En lugar de la "COMA DECIMAL", que, empleamos nosotros, en BASIC se utiliza el "PUNTO DECIMAL", por lo que escribimos 3.14 y no 3,14.

- Distinguiremos entre la vocal "O" y el dígito "0". Sin embargo, en la pantalla del ordenador y en los listados que suministramos, el dígito cero aparece como 0.

- Al trabajar con números decimales se admite también la "notación inglesa decimal". Así, se puede escribir .001 en lugar de 0.001.

2

Primeros pasos

Sabido es que nuestro ordenador, como todos los de su categoría,utiliza el lenguaje BASIC. Esta denominación corresponde a las siglas de:

Beginner's All–purpose Symbolic Instruction Code

es decir, código de instrucciones simbólicas de uso general para principiantes.

Es un lenguaje de alto nivel; lo cual quiere decir que está más adaptado al programador que al ordenador; los lenguajes de bajo nivel que se acercan más al ordenador que al programador, son más rápidos pero también son más dificiles de aprender.

El BASIC es un lenguaje que tiene, entre otras, las cualidades siguientes:

- Fácil asimilación.
- Potente.
- Flexible.
- Eficaz.
- Ampliamente difundido.

El que nuestro AMSTRAD trabaje, de entrada, con este lenguaje no significa que no pueda hacerlo también en otros; de momento tenemos la posibilidad de emplear LOGO, FORTH, PASCAL, FORTRAM y COBOL que son lenguajes de alto nivel también muy interesantes.

PRIMERAS INSTRUCCIONES

Llamamos instrucción a una orden que le damos al ordenador para que realice una determinada tarea. Todas las instrucciones estarán escritas en inglés, ya que el lenguaje BASIC fué desarrollado en EEUU, y serán de fácil comprensión.

Hay que ser muy meticuloso a la hora de teclar una instrucción; ya que

cualquier error, que para nosotros podría parecer sin importancia, para el ordenador es un obstáculo insalvable.

Sin embargo, es indiferente que la instrucción se escriba en mayúsculas o en minúsculas; ya que el ordenador la entenderá de ambas formas.

Comencemos ahora a describir las primeras instrucciones.

CLS (CLear Screen: limpia pantalla)

Tecleando CLS y presionando la tecla INTRO (ENTER o RETURN), se observa que el efecto que tiene es el de borrar la pantalla.

- Siempre hay que pulsar la tecla INTRO (ENTER o RETURN) para indicarle al ordenador que ya hemos acabado de teclear la instrucción y que debe ejecutarla.
- El efecto que ha provocado esta instrucción es el apuntado, pero la pantalla no queda completamente limpia si no que aparece el mensaje:

```
Ready (listo)
```

  Esta es una manera, que tiene nuestro ordenador, de comunicarnos que todo ha ido bien y que queda a la espera de la instrucción siguiente.

- Si en lugar de CLS hubieramos teclado por error CLD, el ordenador, que no está preparado para entender esa orden, nos devuelve el siguiente mensaje:

```
Syntax error (error sintáctico)
```

PRINT (imprimir)

Necesitamos de esta instrucción cuando deseemos escribir en la pantalla. Veamos algunos ejemplo:

- Tecleando: `PRINT 12` (y pulsando INTRO, ENTER o RETURN)

se consigue imprimir el número 12.

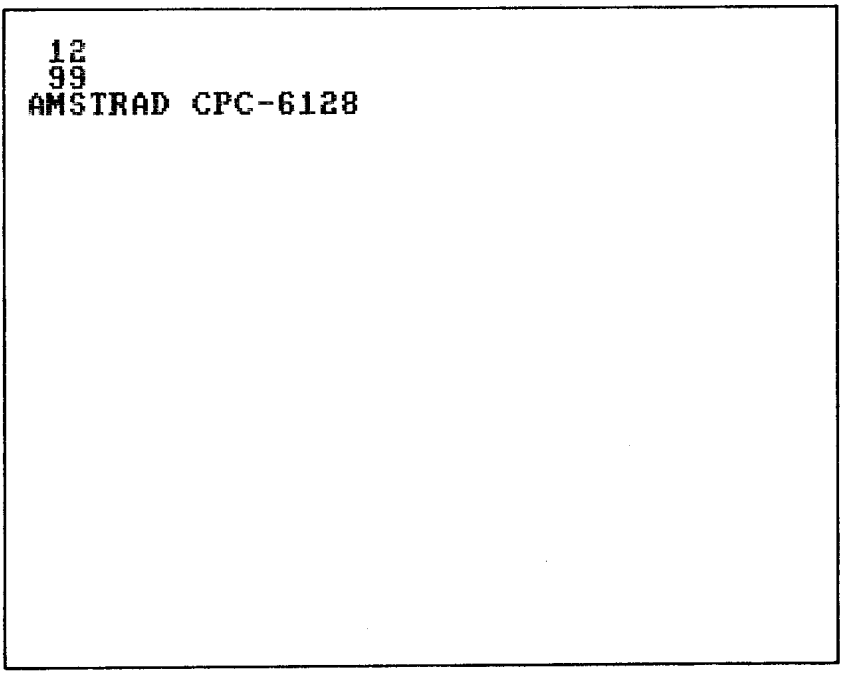
- Con: `PRINT 42+57` (y pulsando INTRO ENTER o RETURN)

obtendremos el resultado de la suma de esos dos números.

– También se pueden imprimir mensajes en los que entren todo tipo de caracteres, no solamente numéricos; lo único que tendremos que recordar es que el mensaje deberá ir entre comillas. Así, con

```
PRINT "AMSTRAD CPC-6128"
```

al pulsar INTRO, ENTER o RETURN tendremos:



NOTAS

1) El presionar INTRO, ENTER o RETURN es imprescindible para que la orden se ejecute; desde este momento no lo seguiremos repitiendo.

2) Por lo general es necesario dejar un espacio en blanco tras PRINT, pero si éste va seguido de comillas no es obligatorio. Así PRINT12 nos dará error sintáctico y PRINT"12" no.

3) Siempre que el ordenador encuentra una instrucción PRINT, y mientras no se le indique lo contrario, escribirá en la línea siguiente a la última empleada.

4) La instrucción PRINT es de las pocas que tiene abreviatura, ya que en lugar de

```
PRINT 12
```

podemos teclear

```
?12
```

obteniendo el mismo efecto.

En este caso no hace falta espacio en blanco detrás del signo "?", ya que el ordenador lo coloca automáticamente.

5) En aquellas ocasiones en que se deben usar comillas, es suficiente con colocar las primeras. Por lo tanto:

```
PRINT "AMSTRAD CPC-6128
```

equivalente a:

```
PRINT "AMSTRAD CPC-6128"
```

6) Observar la diferencia entre

```
PRINT 42+57   y   PRINT "42+57"
```

en el primer caso se imprime el resultado de la operación y en el segundo la expresión entrecomillada tal y como está.

MODE (modo)

Admite tres formas posibles:

```
MODE Ø
MODE 1
MODE 2
```

Con esta instrucción seleccionamos el tamaño de la escritura en pantalla, además de borrar todo lo que hubiera en ella.

Al conectar nuestro ordenador, y de forma automática, se selecciona el MODE 1.

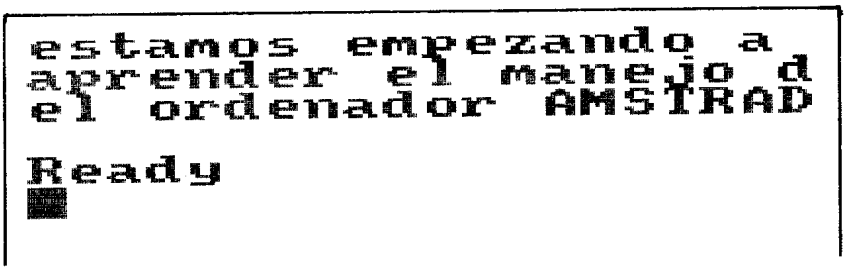
En cualquier de los tres modos, disponemos de una pantalla con posibilidad de contener 25 líneas de texto, siendo el número de columnas de 20, 40 y 80 según sea el MODE 0, MODE 1 ó MODE 2 el elegido.

Evidentemente, cuantas más columnas admita, más pequeña tendrá que ser la letra, ya que en todos los casos se emplea la misma parte de pantalla.

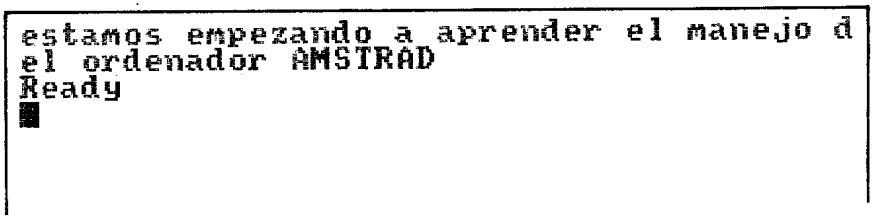
Por ejemplo, si con MODE 0 tecleamos:

```
PRINT "estamos empezando a aprender el mane
jo del ordenador AMSTRAD"
```

tendremos:



si hubiera sido en MODE 1 tendríamos:



y en MODE 2

```
estamos empezando a aprender el manejo del ordenador AMSTRAD
Ready
```

Hay que añadir que el espacio en blanco detrás de MODE es imprescindible puesto que si lo omitimos dará error de sintaxis.

OPERADORES ARITMETICOS

El ordenador emplea los siguientes operadores aritméticos:

```
+ SUMA.
- RESTA.
* MULTIPLICACION.
/ DIVISION.
↑ POTENCIACION.
```

El producto y la suma se obtiene con SHIFT y las dos teclas que están a la derecha de la "L", en los primeros modelos, y con las dos teclas situadas a la izquierda de RETURN en los modelos más recientes.

La resta y la potenciación se obtienen con las dos teclas situadas a la derecha del 0.

La división está 3 teclas a la derecha de la "M".

Probar:

```
PRINT 3*5
PRINT 8/4
PRINT 2↑3
```

Cuando se emplean varios operadores en la misma expresión, existe un orden de prioridad para su ejecución.

No obtenemos el mismo resultado si en la expresión:

```
3*5+2
```

primero se ejecuta el producto y después la suma que si primero se ejecuta la suma y después el producto. En el primer caso el resultado es 17 y en el segundo 21.

El orden de prioridad, en la ejecución de operaciones, en BASIC es el mismo que el empleado en el cálculo aritmético elemental, es decir:

```
1 - POTENCIACION.
2 - PRODUCTO Y DIVISION.
3 - SUMA Y RESTA.
```

Si hay operaciones que tienen la misma prioridad, se ejecuta primero la operación situada más a la izquierda de la expresión. Así, en:

```
PRINT 6/2*3
```

como el producto y la división tienen igual prioridad, primero se ejecuta el cociente y después el producto, con lo que el resultado es 9.

Este orden de prioridad se puede alterar con el uso del paréntesis, ya que cuando en una expresión aparecen paréntesis, primero se realizan todas las operaciones incluidas entre ellos. Así, en el caso anterior, si queremos que primero se ejecute el producto y después el cociente, tendremos que poner:

```
PRINT 6/(2*3)
```

y el resultado será 1.

Si no fuese por estas prioridades el empleo del paréntesis estaría más generalizado.

EJEMPLO 1

Vamos a ver, paso a paso, en qué orden se realizan las operaciones siguientes:

```
2*(3.5-1)/5+3↑2
```

1 - El paréntesis, quedando: `2*2.5/5+3↑2`
2 - La potencia, quedando: `2*2.5/5+9`
3 - El producto, quedando: `5/5+9`
4 - El cociente, quedando: `1+9`
5 - Y, por fin, la suma, dando como resultado: `1Ø`

Comprobarlo introduciendo PRINT 2*(3.5−1)/5+3↑2. Evidentemente,

sólo se imprimirá el resultado final, no los pasos intermedios.

Por medio de la potenciación podemos calcular raices de cualquier indice, ya que raiz de índice n del número x es x↑(1/n).

Así, la raiz cúbica de 8 será: 8↑(1/3). Observar que si no se coloca paréntesis el significado de la expresión sería 8/3.

Hay dos nuevos operadores aritméticos en nuestro AMSTRAD, que son:

```
\   : DIVISION ENTERA.

MOD: RESTO DE LA DIVISIÓN ENTERA.
```

Al dividir, por ejemplo 25 entre 3, sin sacar decimales, obtenemos 8 de cociente y 1 de resto.

Con el ordenador se puede calcular dicho cociente empleando la instrucción:

```
PRINT 25\3
```

y el resto se obtendrá con:

```
PRINT 25 MOD 3
```

Con estos nuevos operadores, el orden de prioridad de las operaciones aritméticas queda del siguiente modo:

```
1.- ↑
2.- MOD
3.- * y /
4.- \
5.- + y -
```

Como ejercicio recapitulador de estas operaciones, completa el siguiente cuadro sin utilizar tu ordenador:

OPERACION	RESULTADO
2+3*5	17
(2↑3)*5	4Ø
2↑3*2	
2+3*5-4*4	
16↑(1/2)	
1+5*(18\7)	
18 MOD 7	
2*6/3	
2↑3-3*6	

SEPARADORES

Para una mejor presentación del resultado de una operación, podemos proceder del siguiente modo:

```
PRINT "243+125="243+125
```

imprimiéndose en la pantalla:

```
243+125=368
```

La primera parte del mensaje se imprime tal cual, ya que está entrecomillada, y en la segunda se efectúa la operación.

Se puede observar que no hay ninguna separación entre las dos partes. Esto no siempre se podrá hacer así, ya que lo habitual es que sea necesario el empleo de "separadores".

Estos separadores son dos:

```
Punto y coma: ;
Coma: ,
```

Veamos el efecto del punto y coma. Con la orden:

```
PRINT "ABC";"DE"
```

se obtiene en pantalla:

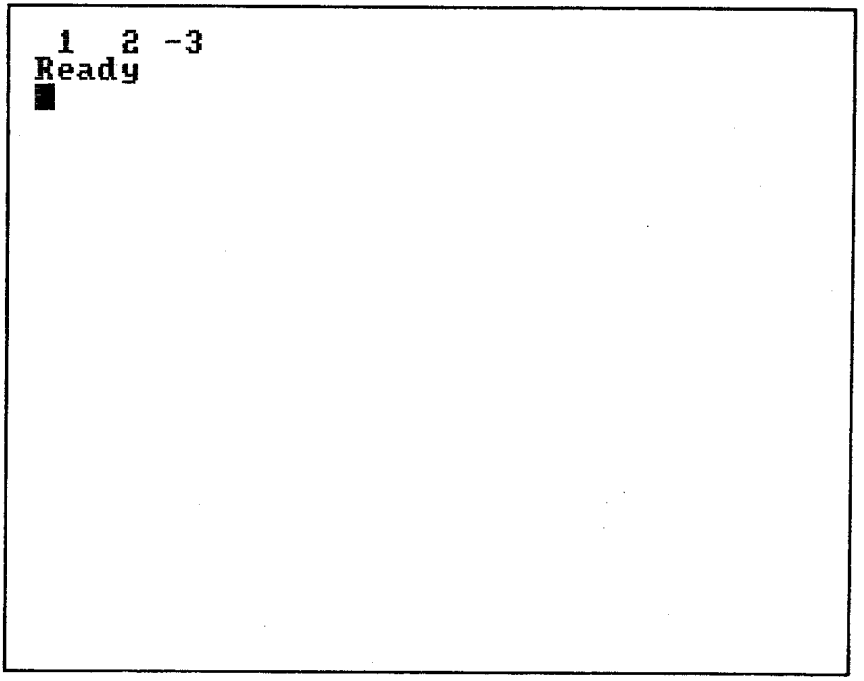
```
ABCDE
Ready
```

En este caso se imprime una expresión junto a la otra sin ninguna separación.

Pero tratándose de números, el efecto es otro; probemos con:

```
PRINT 1;2;-3
```

y observaremos que al imprimir un número, se reserva un espacio a la derecha y otro a la izquierda; éste último para el signo aunque si es positivo no se imprime. El resultado de la anterior instrucción será:

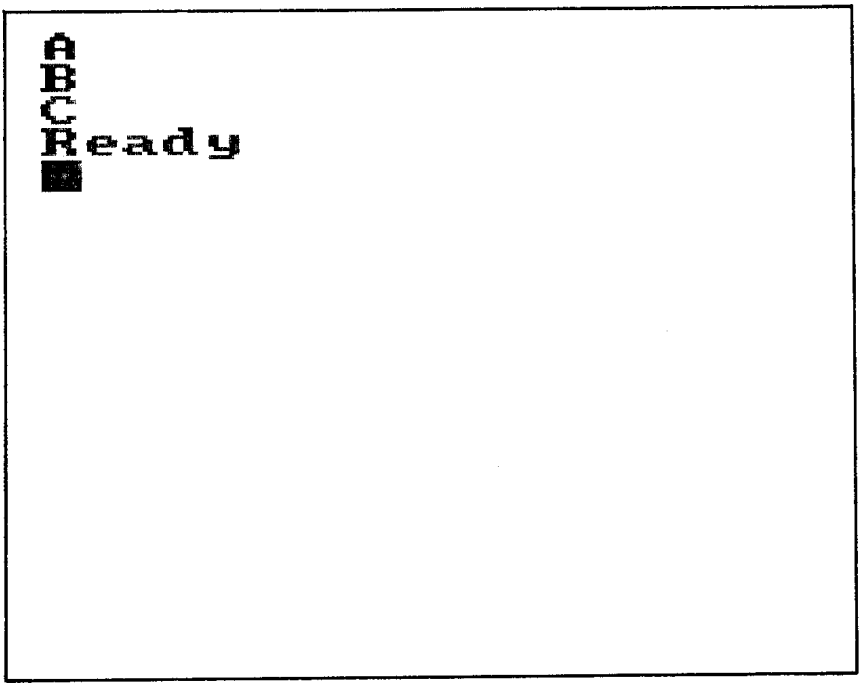


El efecto de la coma es diferente según sea el MODE en que se trabaje.

Así, con MODE 0, la instrucción:

```
PRINT "A","B","C","D"
```

dará como resultado:

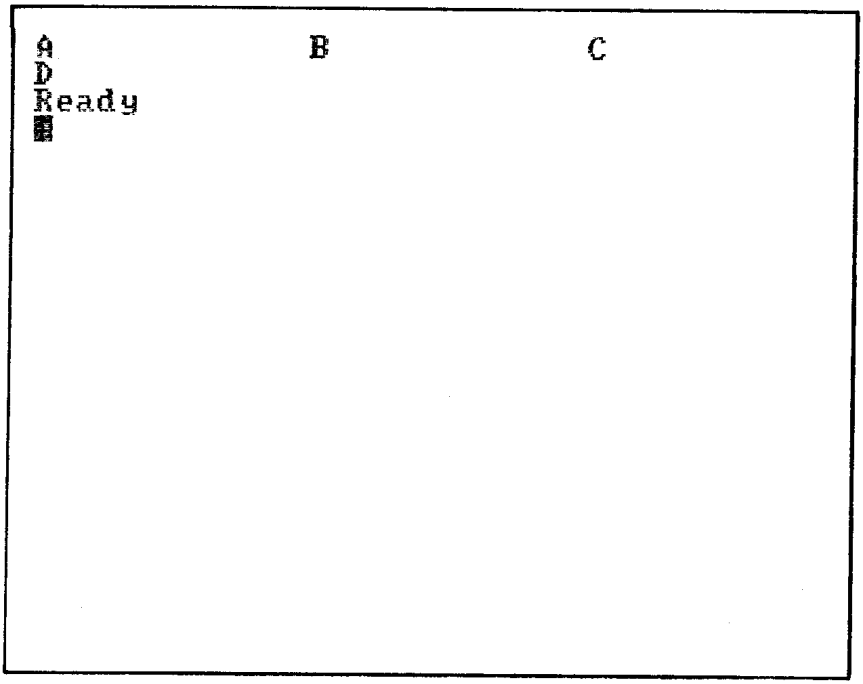


por lo que el efecto de la coma es el de saltar a la línea siguiente.

En MODE 1, y con

```
PRINT "A","B","C"
```

obtendremos:

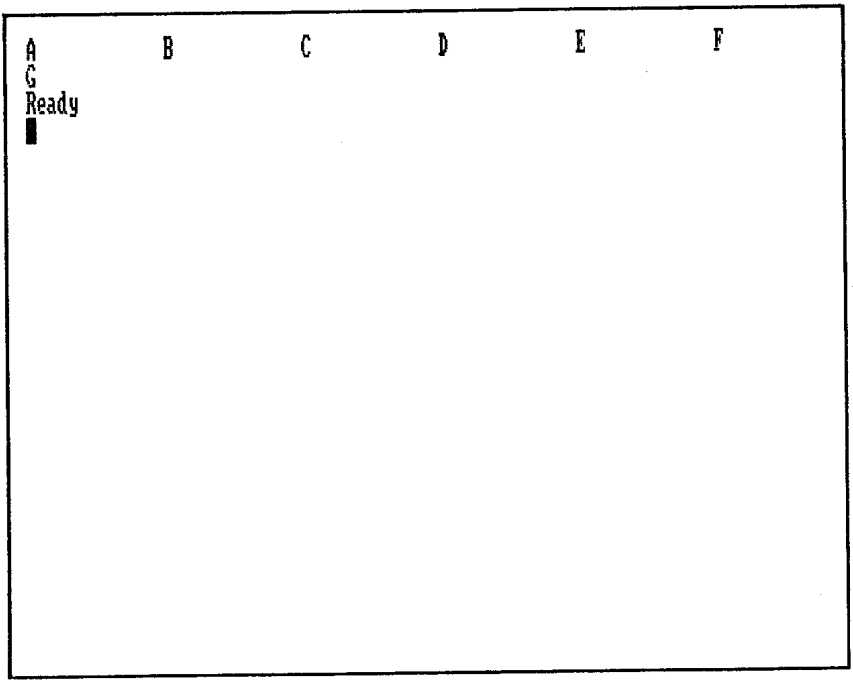


En este modo, cada vez que se encuentra con una coma, salta 1/3 de línea.

Con MODE 2 y

```
PRINT "A","B","C","D","E","F","G"
```

obtendremos:



ya que, en este caso, cada vez que encuentra una coma, salta 1/6 de línea.

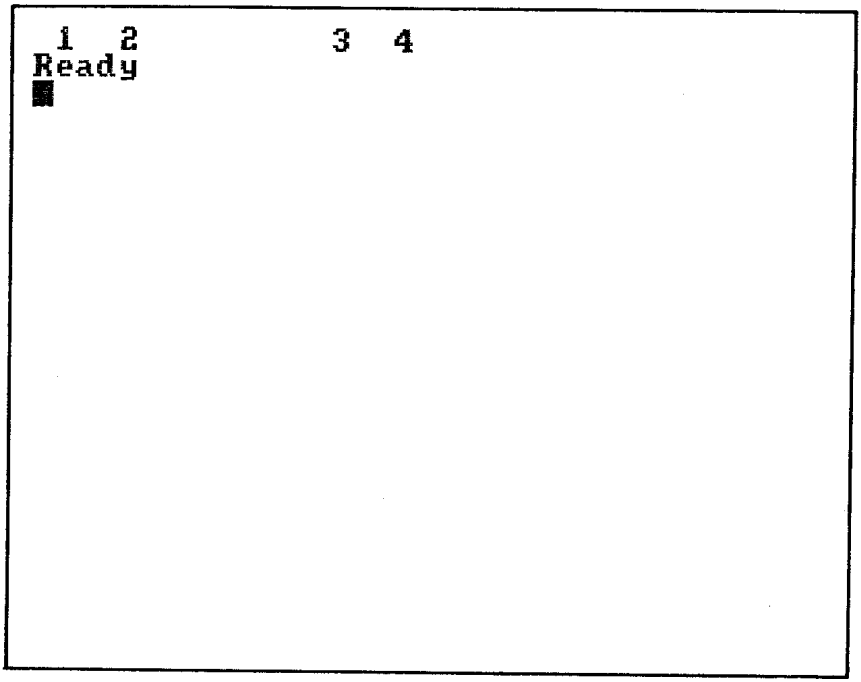
Tratándose de números, el efecto es el mismo pero con la puntualización ya señalada anteriormente de los espacios que se reservan antes y después del número.

Los saltos que hemos comentado son los preestablecidos, pero más adelante veremos una instrucción que puede modificarlos a nuestro gusto.

En una misma expresión, puede ir ambos tipos de separadores; así en MODE 1

```
PRINT 1;2,3;4
```

imprimirá:



VARIABLES

En BASIC, habitualmente, utilizamos valores o expresiones que pueden ser variables. Será necesario, en este caso, asociarles un nombre.

Así, por ejemplo, si queremos obtener el área de un cuadrado de lado 5, se sigue el siguiente proceso:

```
1.- Dar a L el valor 5.
2.- Definir la variable área: A=L*L
3.- Obtener el valor de A cuando L vale 5.
```

Esto mismo, en BASIC, se expresará con la instrucción LET (asigna) en la siguiente forma:

```
LET L=5
LET A=L*L
PRINT A
```

Observar las siguientes particularidades:

- Después de la instrucción LET, se debe dejar un espacio en blanco.
- El ordenador AMSTRAD, para ahorrarnos trabajo, permite suprimir LET. Es decir, se consigue el mismo efecto con:

```
L=5
A=L*L
?A
```

- Se puede definir una variable en función de otra. En nuestro caso, A está definida en función de L.
- En lugar de A=L*L también puedes escribir A=L↑2

Tres son los tipos de variables que usaremos:

Variable numérica real

Es la preestablecida en el ordenador y asigna números reales comprendidos entre 2.9*10↑−39 y 1.7*10↑38 tanto positivos como negativos.

El nombre que recibirán estas variables deberá estar compuesto únicamente por letras y dígitos, debiendo comenzar siempre por una letra.

Es decir, serán correctas las variables reales denominadas por:

```
A
A12B
bC1bA
```

y no lo son:

```
1A
127A2b
*A
```

El ordenador no distingue en el nombre de las variables entre mayúsculas y minúsculas. Comprobarlo con

```
A=36
?a
```

Variable numérica entera

Si la variable sólo puede tomar valores numéricos enteros comprendidos entre −32768 y 32767, podremos considerarla como variable entera, lo que nos permitirá ahorrar memoria y ganar en rapidez.

Las variables enteras se denominan igual que las reales pero añadiéndoles al final el signo %. Por ejemplo:

```
A%
A12%
```

Por las ventajas que ya hemos apuntado, se deben utilizar siempre que se pueda.

Variables alfanuméricas

Estas variables estarán formadas por caracteres de cualquiera de los tres estudiados; esto es, alfabéticos, numéricos y especiales. A la hora de definirlas tenemos que finalizar su nombre con el signo "$" y encerrar los caracteres que componen la variable entre comillas.

Por ejemplo:

```
A$="JUAN"
a12B$="J12*/"
A23$="Casa
```

Observar que, en el caso de una variable alfanumérica, puedes eliminar las últimas comillas.

El máximo número de caracteres que puede tener una variable alfanumérica es 255.

MUY IMPORTANTE

En BASIC aparecerán muy a menudo expresiones del tipo:

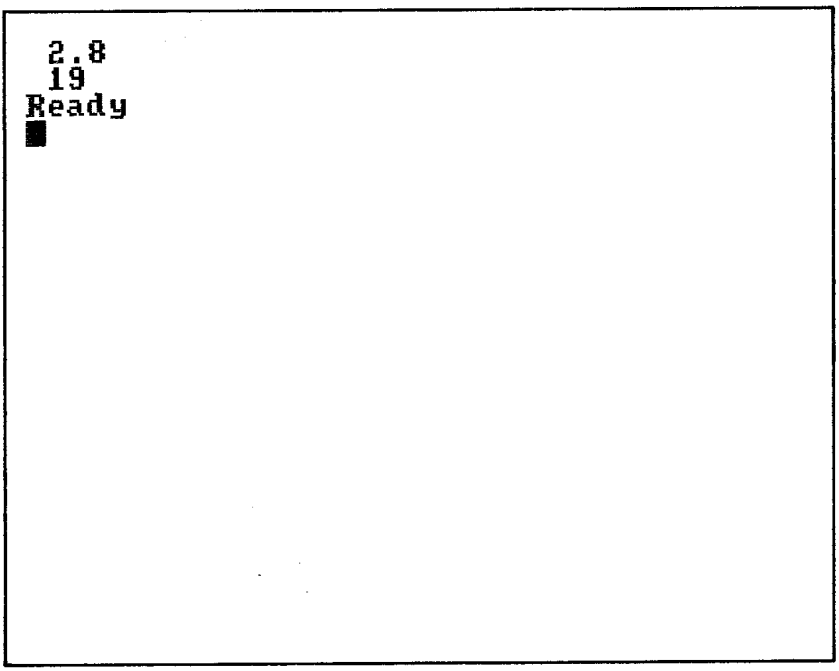
```
.A=A+1
```

que no tienen sentido desde el punto de vista matemático, pero que sí lo tendrá en nuestro lenguaje. A la variable A se le asigna su valor anterior aumentado en una unidad.

Así, con

```
L=2.8
?L
L=L+1
L=L*5
?L
```

se obtendrá:



NOTAS

1) Si un número real tiene más de nueve dígitos, el ordenador lo expresa mediante la llamada "notación exponencial".

```
a E b = a * 10↑b
```

Por ejemplo:

```
1234567890 se expresa 1.23456789E+09
-0.0000003789 se expresa como -3.789E-07
```

2) Se pueden definir variables reales en función de variables enteras. Así, tiene sentido escribir:

```
A%=12
AS=A%*5
```

3) Los nombres de las variables nunca pueden coincidir con los de las de las diferentes instrucciones que existen para AMSTRAD. Así, por ejemplo, con las instrucciones:

```
LET=34
PRINT=45.9
```

obtendremos el mensaje de error:

```
Syntax error
```

ya que esas denominaciones están reservadas para las instrucciones en BASIC.

Con el fin de clarificar los programas y tener una mayor comodidad para manejar las variables, AMSTRAD permite definir, por intervalos, el tipo de las variables que vamos a emplear. Para ello hay tres instrucciones que vamos a estudiar ahora:

DEFINT

Con ella se consideran las variables como enteras sin necesidad de utilizar el símbolo %. Admite tres versiones:

```
DEFINT p
DEFINT p,q,r
DEFINT c-t
```

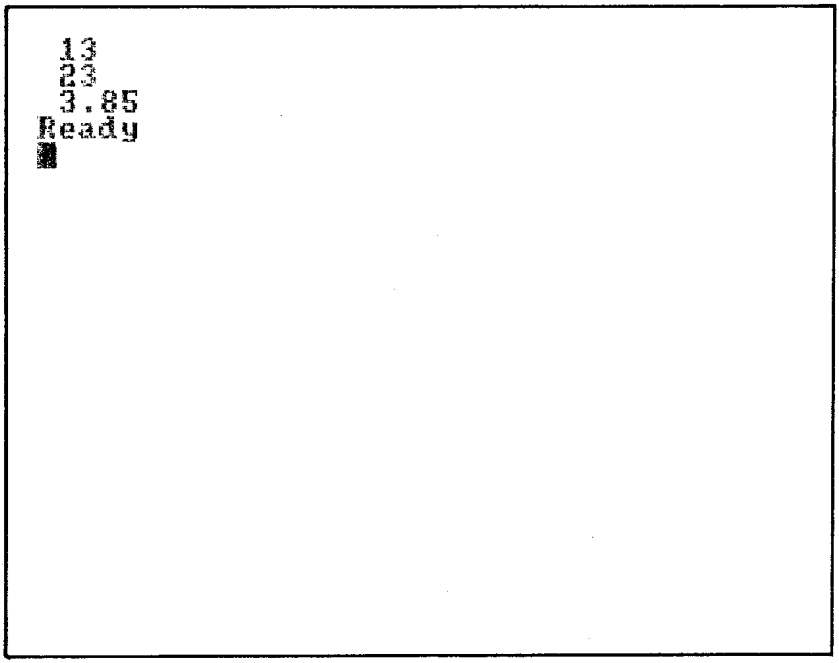
en el primer caso, se consideran como enteras todas las variables cuyo nombre empieza por "p". De igual modo, en el segundo caso, tomará como enteras las variables cuyo nombre empieza por p, q ó r. La última versión, considera como enteras todas las variables cuya primera letra esté comprendida entre las dos letras indicadas (incluidas ambas).

Además, DEFINT, transforma toda variable real en entera. Esto se consigue redondeando al entero más próximo.

Por ejemplo, con:

```
DEFINT c-t
juan=12.8
pepe=23.2
vicente=3.85
?juan
?pepe
?vicente
```

en la pantalla aparece:



DEFREAL

Es análoga a la anterior, pero considerando la variable como real, o sea un número decimal. Mientras no se diga lo contrario, el ordenador toma siempre una variable numérica como real.

DEFSTR

Del mismo modo que las anteriores, ahora todas las variables cuyos nombres empiecen por las letras señaladas se considerarán alfanuméricas.

Con estas tres instrucciones podremos estructurar cómodamente los programas dedicando un conjunto de letras a variables enteras, otro a reales y otro a alfanuméricas.

3

Modo programa

Existe otra manera, diferente a la estudiada en el capítulo anterior, de trabajar con nuestro ordenador. Hasta ahora, la orden que se tecleaba era ejecutada inmediatamente después de pulsar "INTRO", "ENTER" o "RETURN". A esta forma de trabajar se le llama "MODO DIRECTO".

Otra posibilidad, mucho más interesante, consiste en teclear las instrucciones precedidas de un número, al que llamaremos "número de línea". En este caso, al pulsar "INTRO" la instrucción no es obedecida inmediatamente sino que es almacenada en la memoria del ordenador y queda a la espera de la posterior orden de ejecución. A esta forma de trabajar se le llama "MODO PROGRAMA".

Así, tecleando:

```
30 PRINT "CPC-6128"          e "INTRO"
```

después

```
10 CLS          e "INTRO"
```

y, por fin

```
20 PRINT "AMSTRAD"          e "INTRO"
```

habremos conseguido una lista de instrucciones sin que, de momento, ninguna de ellas se haya ejecutado.

NOTAS

1) El orden en que introduzcamos las líneas no tiene importancia. El ordenador, llegado el momento, las ejecutará secuencialmente de menor a mayor numeración.

2) El número de línea tiene que ser un entero comprendido entre 1 y 65535.

3) Cada línea puede tener hasta un máximo de 255 caracteres. Si pretendemos introducir una línea con más caracteres, el ordenador no lo permitirá y nos avisará con un pitido.

4) Es una práctica habitual entre programadores el numerar las líneas a saltos como, por ejemplo, de 10 en 10 o de 5 en 5; esto facilita el poder insertar nuevas líneas que por error u olvido hubieramos dejado de colocar.

5) Si queremos borrar una línea de la memoria del ordenador, no tenemos más que teclear su número y presionar "INTRO".

   La línea en cuestión no habrá desaparecido de la pantalla pero si de la memoria.

6) Si tenemos que modificar una línea tendremos, de momento, que volverla a escribir.

7) En una misma línea pueden ir varias instrucciones siempre que se las separe por dos puntos ":". Por ejemplo:

```
5 A=5 : PRINT A
```

8) Otra particularidad del "MODO PROGRAMA" es que las instrucciones, una vez ejecutadas, no son olvidadas por el ordenador, como ocurría en el "MODO DIRECTO", y por tanto se pueden volver a repetir varias veces sin necesidad de teclearlas.

9) Cuando se dice que debemos presionar la tecla "INTRO", nos estamos refiriendo al teclado en español; en los modelos anteriores se trata de la tecla "ENTER" o de "RETURN".

Se llama PROGRAMA a un conjunto de instrucciones precedidas de un número, llamado número de línea, y que el ordenador ejecuta secuencialmente de menor a mayor numeración. En capítulos posteriores veremos que esta secuencialidad en la ejecución no es rígida y que se puede alterar.

NUEVAS INSTRUCCIONES

RUN (ejecuta)

Esta es la orden que hará que nuestro programa se ejecute. Puede tener dos formas:

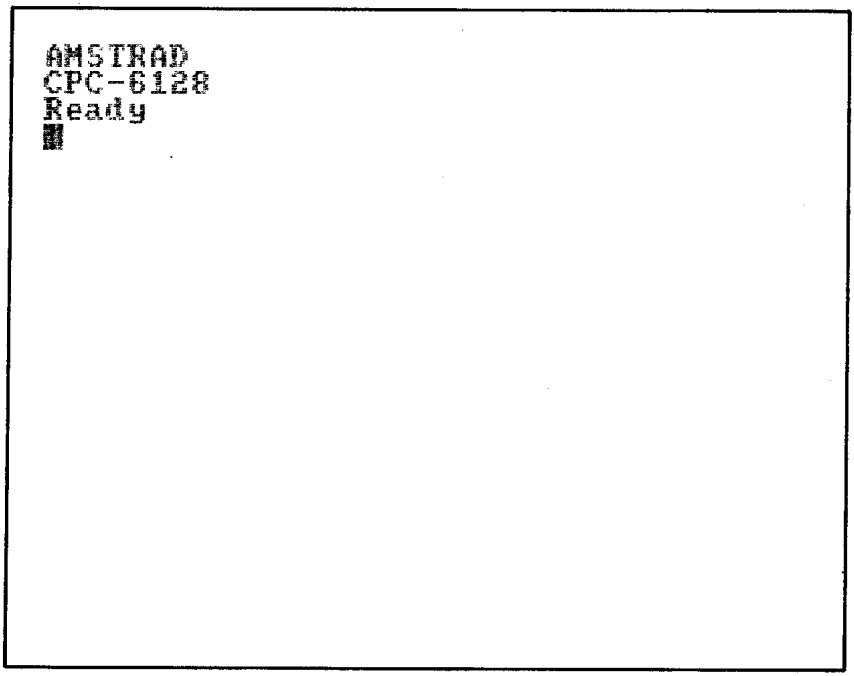
```
RUN

RUN X
```

donde X es un número de línea.

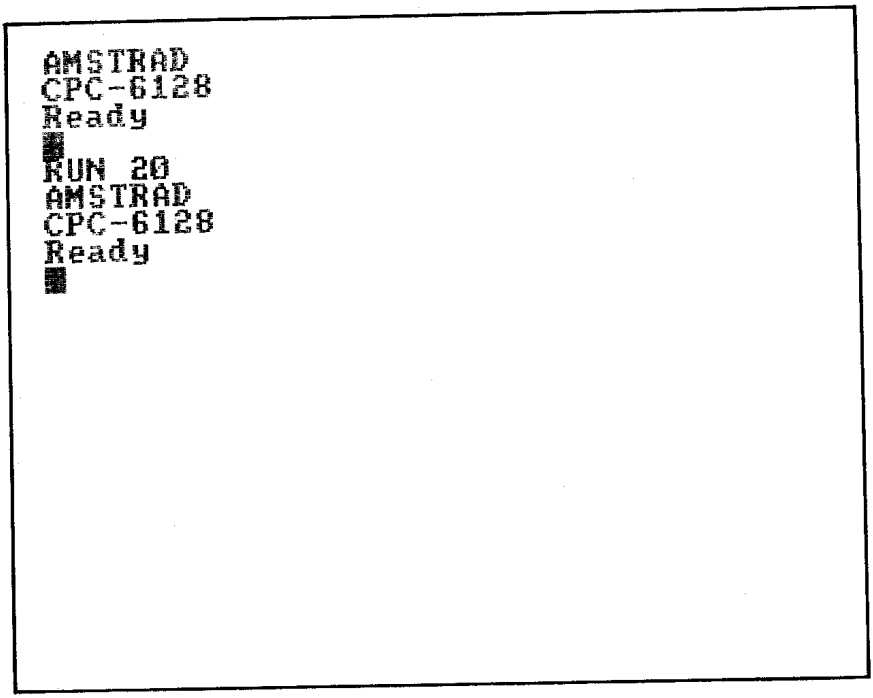
RUN hace que el programa se ejecute a partir de la línea de más baja numeración.

RUN X hace que el programa se ejecute a partir de la línea número X.

Probemos con el programa del apartado anterior y veremos que con la instrucción "RUN" se borra la pantalla y aparece:



Si tecleamos RUN 20, obtendremos:



No se ha borrado previamente la pantalla, ya que no se ejecuta la línea 10.

Esto último demuestra lo dicho anteriormente de que el programa no ha sido borrado de la memoria y que lo podemos ejecutar tantas veces como queramos.

Más adelante, apéndice A, veremos otra aplicación de la instrucción RUN.

Para borrar el programa que hay almacenado en la memoria, y que no interfiera con el nuevo que le vamos a introducir, tenemos varias opciones:

1) Borrar el programa línea a línea, dejando sólo las que podamos necesitar. Esto se consigue tecleando, tras cada número de línea, "INTRO". Por este procedimiento se conservan las variables definidas en el programa.
2) Desconectando con el interruptor del ordenador (no con el del monitor).

3) Pulsando tres teclas simultánemente:

"CONTROL", "MAYS" y "ESC"

o sus equivalentes:

"CONTROL","SHIFT" y "ESC"

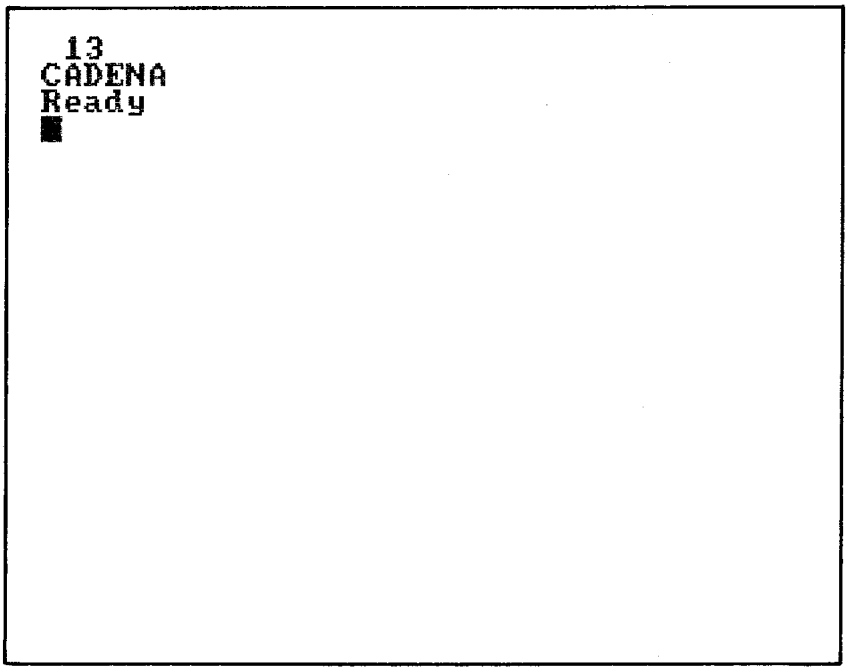
El efecto es el mismo que al desconectar el aparato y, por supuesto, que las variables se borran igual que en el caso anterior.

CLEAR (limpia)

Olvida los valores de las variables, por lo que las numéricas las deja a cero y las alfanuméricas quedan reducidas a la cadena vacía (sin ningún carácter). Teclear el programa:

```
10 CLS
20 A=13
30 A$="CADENA"
40 PRINT A
50 PRINT A$
```

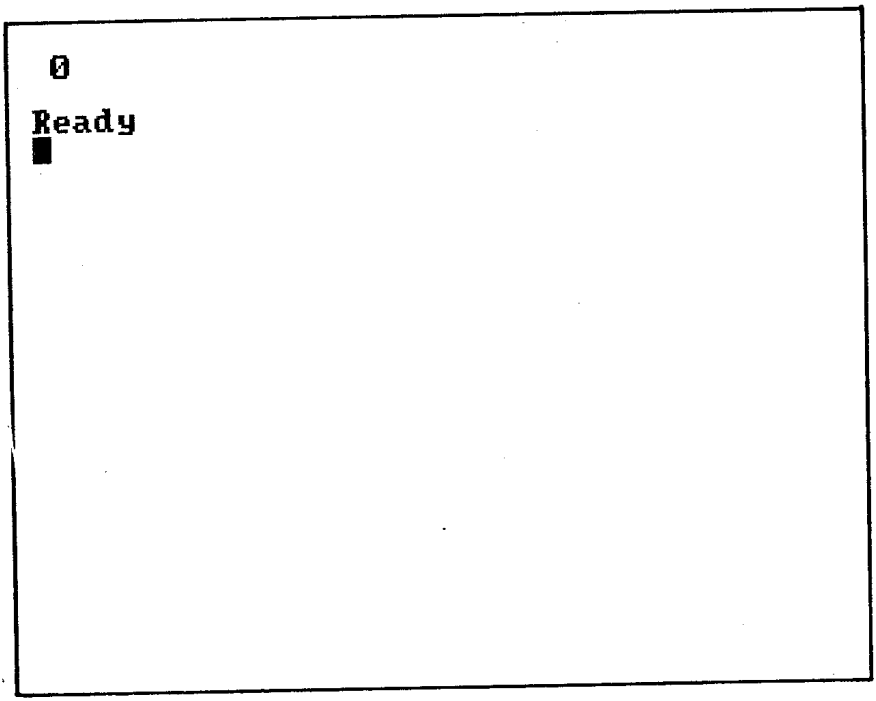
que al ser ejecutado, se obtendrá en la pantalla:



Ahora intercalar:

```
35 CLEAR
```

y así obtendremos:



Si esta instrucción hubiera ido en la línea 25, habría olvidado sólo el valor de A.

Será conveniente borrar el programa para seguir adelante.

NEW (nuevo, renovado)

Borra el programa y las variables que se hubieran definido de la memoria del ordenador, aunque no borra la pantalla. También se seguirá trabajando con el mismo MODE que anteriormente y con los mismos colores que hasta ahora hubieramos elegido (ya se verá cómo).

REM (REMark: comentario)

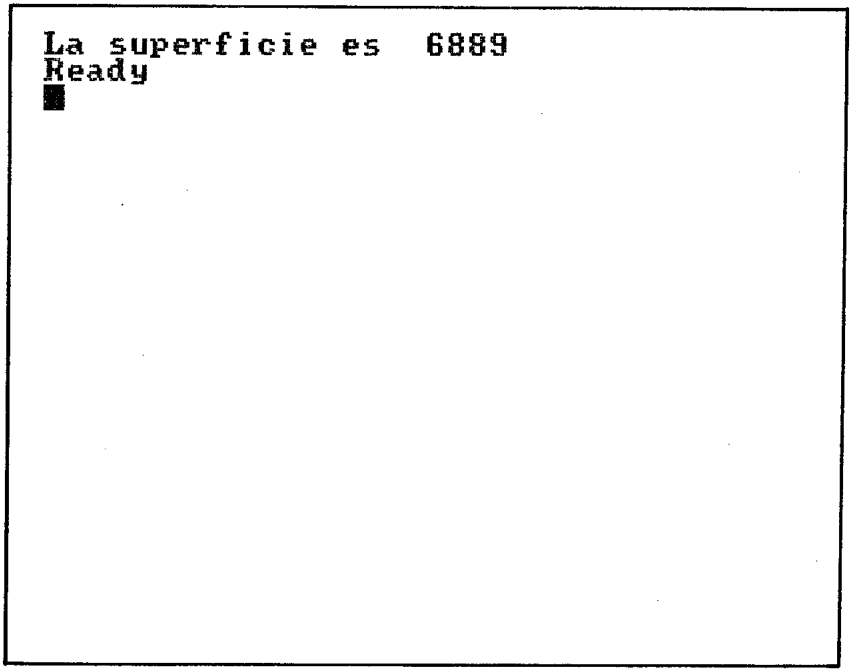
Esta instrucción se utiliza, dentro de un programa, para introducir notas y comentarios con el fin de facilitar su comprensión o, en algunos casos, como simple adorno.

```
10 REM ******************************
20 REM ****************************** *
30 REM SUPERFICIE DE UN CUADRADO * *
40 REM ******************************
50 REM ******************************
60 L=83
70 PRINT "La superficie es ";L*L
```

Esta instrucción, al igual que PRINT, tiene abreviatura: ' ;por tanto, la línea 20 podía quedar de la forma:

```
20 ' SUPERFICIE DE UN CUADRADO **
```

Al ejecutar el programa obtenemos:



Si ahora queremos calcular la superficie de otro cuadrado, por ejemplo de lado 256, tendremos que cambiar la línea 40 por

```
40 L=256
```

Sería conveniente que existiera la posibilidad de introducir datos sin necesidad de modificar el programa. Esto se consigue con la instrucción INPUT que estudiaremos ahora.

INPUT (entrada)

Sirve para introducir datos, a través del teclado, en el ordenador. Así, en el programa anterior, si sustituimos la línea 40 por:

```
40 INPUT L
```

al ejecutar esta instrucción, el ordenador imprime una interrogante en la pantalla y queda a la espera de recibir, por el teclado, un número que asignará a la variable L. De esta manera podremos calcular la superficie de cualquier cuadrado.

Hecha la modificación, el programa anterior queda así:

```
10 ' ***************************
20 ' ***************************
30 ' SUPERFICIE DE UN CUADRADO *
40 ' ***************************
50 ' ***************************
60 INPUT L
70 PRINT "LA SUPERFICIE ES ";L*L
```

Al ponerlo en funcionamiento, nos solicitará el valor de la variable L que nosotros teclearemos y, después de presionar INTRO, nos falicitará la superficie del cuadrado.

Si, en lugar de teclear un número, nos limitamos a presionar INTRO, el ordenador toma el valor de L como 0.

Podemos, incluso, conseguir que se formule la pregunta en la pantalla solicitando el dato:

```
40 INPUT "LONGITUD DEL LADO ";L
```

si el separador empleado es el ";" detrás de la pregunta aparece el signo interrogación. En el caso de emplear como separador la "," no aparecerá dicho signo de interrogación.

También se puede solicitar varios datos, separados por comas, con una sola instrucción INPUT. Así, si queremos calcular la superficie de cualquier triángulo, introduciremos el programa siguiente:

```
1Ø INPUT "BASE,ALTURA ";B,H
2Ø PRINT "LA SUPERFICIE ES ";B*H/2
```

que, al ponerlo en funcionamiento, nos pedirá la base B y la altura H que tendremos que suministrar tal y como se indica en la instrucción, o sea, el valor de la base seguido de una coma y después el valor de la altura.

```
12,3Ø    e INTRO
```

y calculará la superficie de un triángulo de base 12 y de altura 30.

```
BASE,ALTURA? 12,30
LA SUPERFICIE ES 180
Ready
```

Pero no solamente se pueden solicitar, con INPUT, datos numéricos; también es posible solicitar datos alfanuméricos.

Borra, con NEW, el programa anterior y teclea:

```
10 INPUT "nombre ";n$
20 INPUT "primer apellido ";a$
30 CLS
40 PRINT n$;a$
```

al ejecutarlo, se pedirá el nombre y el apellido e inmediatamente después borrará la pantalla e imprimirá esos dos datos.

Es posible que no te haya gustado el resultado ya que el apellido aparece, sin separación, junto al nombre. Esto tiene arreglo:

```
40 PRINT n$;" ";a$
```

ahora se imprimirá el nombre, un espacio en blanco y después el apellido.

Cuando, como en este caso, se nos pide el valor de una variable alfanumérica con INPUT, no es necesario introducir las comillas.

NOTA

Teniendo en cuenta como trabaja la orden INPUT no se podrán introducir comas en una variable alfanumérica, así, por ejemplo, si en

```
2 INPUT B$
```

introducimos la cadena:

```
Adios, querido
```

el ordenador presenta el mensaje de error:

```
Redo from start
```

Para poder introducir comas en una variable, el AMSTRAD, admite una nueva modalidad de la orden INPUT.

```
LINE INPUT
```

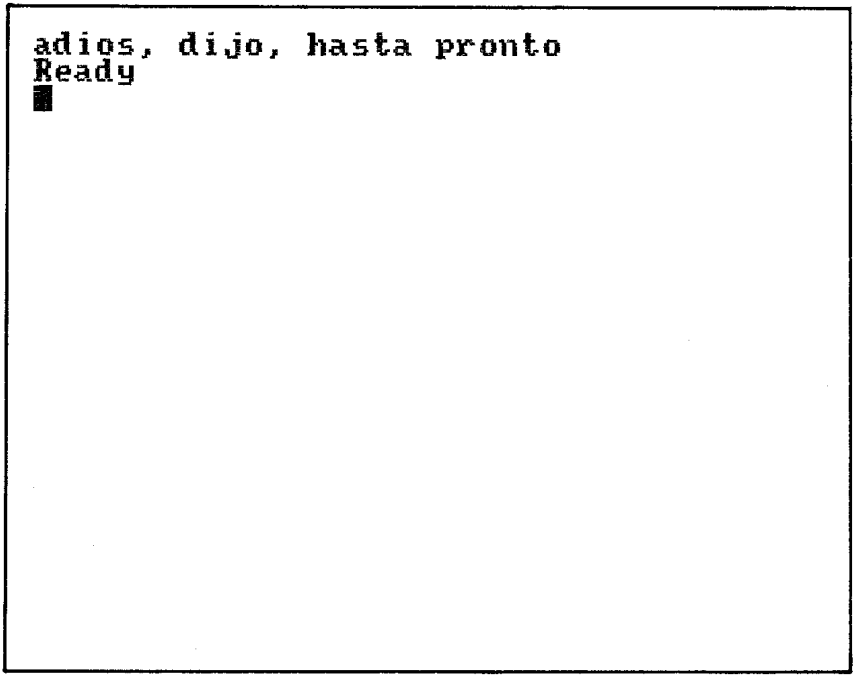
con la que ya puedes introducir comas, por ejemplo:

```
10 LINE INPUT A$
20 PRINT A$
```

al ejecutarlo, si tecleamos:

```
adios, dijo, hasta pronto
```

se tiene:



Veamos otras instrucciones:

STOP (alto)

Al llegar a una instrucción STOP el programa se detiene, apareciendo en la pantalla el mensaje:

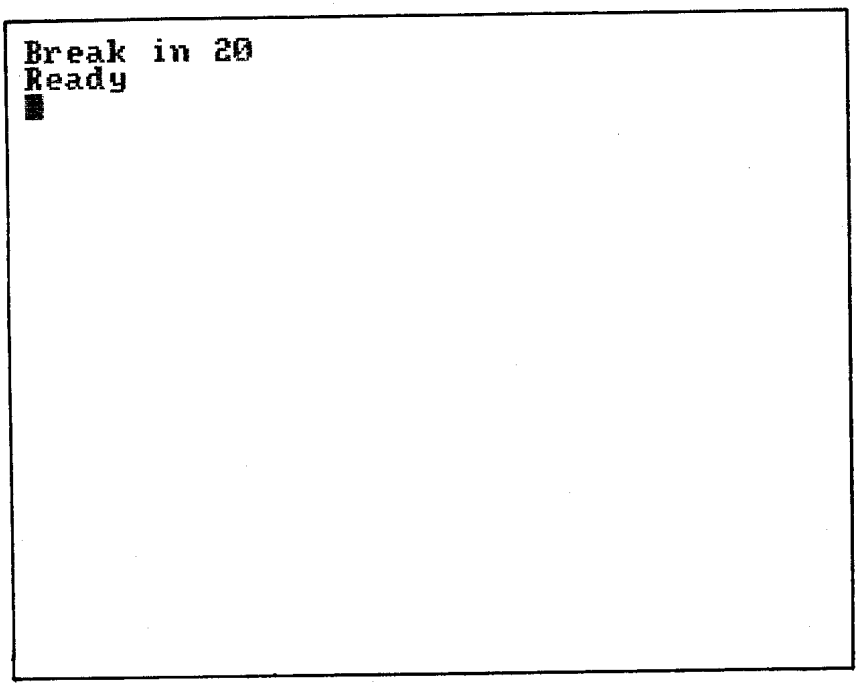
```
Break in n   (BREAK: quebrar,romper)
```

donde n es el número de línea en donde se encuentra esta instrucción.

Así, por ejemplo, con el programa:

```
10 A=255
20 STOP
30 PRINT A
```

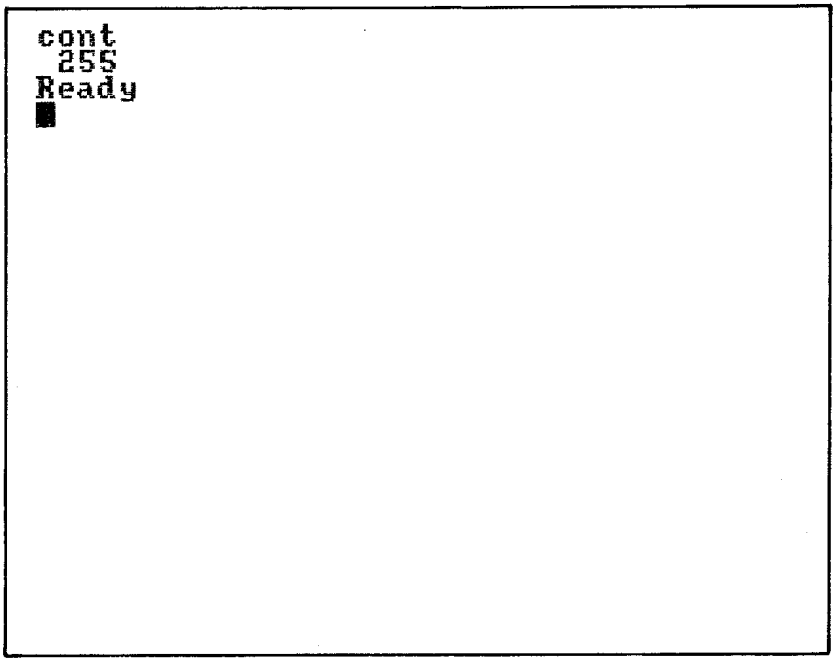
obtenemos:



CONT (CONTinue)

Hace que un programa que ha sido parado con STOP, continue ejecutándose en la instrucción siguiente.

Se emplea como comando directo. Por lo tanto, si en el caso anterior, se teclea CONT (e INTRO) se obtiene:



END (Fin)

Es una instrucción que señala el fin del programa. La instrucción CONT no tiene efecto sobre ella.

4

Modificación de programas

En ocasiones, al ejecutar un programa, la pantalla se borra y el listado desaparece. Si queremos que vuelva a aparecer, se puede conseguir con la instrucción LIST.

LIST (Lista)

Adopta varias formas:

LIST	lista todo el programa.
LIST n	lista la línea n solamente.
LIST n-	lista desde la línea n, incluida, al final.
LIST -n	lista desde la primera línea hasta la n, incluida.
LIST m-n	lista desde la línea m hasta la n, ambas inclusive.

A veces nos encontramos con listados lo suficientemente largos para que no quepan enteros en la pantalla. En este caso, el ordenador, una vez que ésta ha sido llenada, va sacando una línea por abajo por cada una que pierde por arriba (SCROLL, enrrollamiento), por lo que todo el listado pasará ante nosotros.

Para parar, momentáneamente, este efecto, se presiona la tecla "ESC" en una ocasión. Si queremos continuar sacando líneas del listado, presionaremos cualquier otra tecla. Y si, por el contrario queremos parar definitivamente, presionaremos por segunda vez "ESC".

NOTAS

1) Al listar un programa, todas las líneas se ordenan de menor a mayor numeración, cualquiera que fuese el orden en que se han introducido.

2) Si tecleamos las instrucciones con minúsculas, todas aquellas que hayan sido escritas correctamente, aparecerán en mayúsculas al emplear LIST. De no ser así, hay que sospechar que el listado tiene algún error.

3) La tecla "ESC" también es capaz de parar, momentáneamente, la ejecución de un programa si se le pulsa en una ocasión solamente. Presionando cualquier otra tecla después, el programa continúa ejecutándose. Sin embargo, si se presiona dos veces seguidas la tecla "ESC" el programa habrá sido interrumpido, apareciendo el mensaje:

```
Break in n
```

donde n es la línea en donde hemos parado el programa.

BORRADO DE LINEAS

Ya hemos mencionado un procedimiento para suprimir una línea de un programa. No tenemos más que escribir su número y al pulsar "INTRO" desaparecerá de la memoria de nuestro ordenador.

El problema se planteará cuando tengamos que borrar un bloque de varias líneas de un programa. Para ello disponemos de la instrucción:

DELETE

Que puede adoptar una de las formas siguientes:

DELETE	borra todo el programa.
DELETE n	solo borra la línea n.
DELETE n-	borra desde la línea n, incluida, hasta el final.
DELETE -n	borra desde la primera línea hasta la n incluida.
DELETE m-n	borra desde la línea m hasta la n, ambas inclusive.

MODIFICACION DE LINEAS

Pero si lo que queremos es corregir errores de una línea, sin necesidad de borrarla y volverla a escribir de nuevo, debemos acudir a otros procedimientos que estudiaremos ahora.

EDIT (Editar)

La forma general es

```
EDIT n
```

donde n es un número de línea.

El efecto que tiene es el de editar la línea n; ésto significa que nos presenta en la pantalla una copia de dicha línea con el cursor incluido para que efectuemos las correcciones necesarias. Podremos borrar, intercalar o añadir lo que queramos.

Incluso hay ocasiones en que el ordenador emplea este método automáticamente al encontrarse determinados errores en el listado.

Probemos con el programa:

```
10 SUMA
20 MODE1
30 IMPUT "DEME DOS NUMEROS";A,B
40 PRINT "SU SUMA ES";A+B
```

que contiene varios errores.

En la línea 10 falta la instrucción REM, y para corregirla seguiremos los pasos siguientes:

1) Editaremos la línea 10 con EDIT 10.
2) El cursor, en este caso, aparece en el sitio justo en donde lo necesitamos, por lo que sólo nos queda escribir la instrucción REM.
3) Pulsaremos "INTRO" y el error habrá sido subsanado.

Hay que corregir también la línea 20, ya que falta el espacio entre MODE y el 1. Para ello seguiremos los siguientes pasos:

1) Editaremos la línea 20 con EDIT 20.
2) Moveremos el cursor hasta situarlo sobre el "1".
3) Presionaremos la barra espaciadora y luego "INTRO" y el error habrá sido corregido.

El tercer error aparece en la línea 30, ya que en lugar de INPUT se ha escrito IMPUT. Dejamos como ejercicio el modificar dicha línea.

Todavía tenemos a nuestra disposición otro procedimiento para corregir errores:

METODO DEL CURSOR DE COPIA

Partamos del ejercicio anterior. Ya sabemos que para controlar los movimientos del cursor, hay que usar las cuatro teclas que están junto al teclado numérico. Pero si a la vez que presionamos "MAY" también presionamos alguna de dichas teclas, observaremos que un segundo cursor se mueve por la pantalla: es el "CURSOR DE COPIA".

Para corregir errores, el procedimiento general a seguir es el siguiente:

1) Coloquemos el cursor de copia al comienzo de la línea a modificar, usando las teclas de movimiento del cursor junto con "MAYS".
2) Por cada toque que le demos a la tecla "COPIA" ("COPY"), obtendremos la copia de un carácter. Esta copia aparece a partir de la posición en que se encuentra el primer cursor.
3) Si hay que intercalar algo, se escribe directamente.
4) Si hay que dejar de copiar algún caracter de la línea original se mueve el cursor con "MAYS" y la tecla correspondiente de movimiento del cursor.
5) Al final se pulsa "INTRO" y ya está hecha la corrección.

Modifiquemos, por este procedimiento la línea 10.

1) Pulsamos "MAYS" y las teclas correspondientes de movimiento del cursor, hasta colocar el cursor de copia sobre el "1" de la línea 10.
2) Pulsar tres veces la tecla "COPIA" hasta que el cursor de copia se coloque sobre al "S".
3) Teclear la instrucción REM y luego un espacio en blanco.
4) Con el cursor de copia, copiar el resto de la línea, como ya hemos dicho antes.
5) Cuando hayamos acabado, se pulsa "INTRO".

Modifiquemos, ahora, la línea 20.

1) Subamos el cursor hasta dicha línea.
2) Con "COPIA", copiar todos los caracteres hasta que el cursor de copia esté justo encima de la letra a cambiar.
3) Pulsar la barra espaciadora.
4) Volver a pulsar "COPIA" una sola vez.
5) Y, por fin, "INTRO".

Ahora nos toca corregir la línea 30.

1) Colocar el cursor de copia al comienzo de la línea 30.

2) Pulsamos "COPIA" cuatro veces, hasta que el cursor de copia esté justo encima de la letra a cambiar.
3) Escribamos la "n".
4) Movamos el cursor de copia un lugar a la derecha para saltar la "m". Ya hemos dicho que esto se consigue con "MAYS" y la tecla que mueve el cursor en este sentido.
5) Seguiremos copiando la línea hasta el final. También la tecla "COPIA" tiene autorrepetición.
6) Pulsamos "INTRO".

Ahora veremos dos nuevas instrucciones que nos permitirán trabajar con mayor comodidad:

AUTO

Su forma general es:

```
AUTO a,b
```

y nos suministra, de forma automática, los números de línea a la hora de introducir un listado; por lo que no tendremos que teclearlos nosotros.

a: es la numeración de la primera línea.
b: es el salto entre dos líneas consecutivas.

Por ejemplo:

```
AUTO 10,5
```

irá suministrando los números de línea, empezando por el 10, y de 5 en 5. O sea, 10,15,20,25,.....

Pueden faltar uno o los dos parámetros a y b y entonces el ordenador los toma como 10.

Así

AUTO 5 empezará la numeración en la línea 5 y se incrementará de 10 en 10.

AUTO ,1 empezará en la línea 10 y se incrementará de 1 en 1.

AUTO empezará en la línea 10 y se incrementará de 10 en 10.

Si se está trabajando con AUTO, se debe teclear la instrucción que va en la línea cuya numeración ya la suministra el ordenador, y cuando se haya acabado se pulsa "INTRO". El ordenador suministrará entonces la numeración de la siguiente línea etc., etc.....

Cuando hayamos acabado, presionaremos "ESC" y el proceso se termina.

Podría ocurrir que al emplear AUTO la numeración de una línea suministrada automáticamente coincida con alguna línea guardada en la memoria. Entonces dicha línea aparece en la pantalla y tenemos dos opciones:

1) Guardarla tal y como está, para lo que sólo tendrá que pulsar "INTRO", o bien:
2) Borrarla, escribiendo una nueva línea.

Si con AUTO, se sobrepasa el número máximo permitido para la numeración de una línea, margen 1 – 65535, el proceso se interrumpe.

RENUM (REeNUMerar)

Su forma general es:

```
RENUM a,b,c
```

El efecto que tiene es el de reenumerar las líneas de un programa.

a: es la numeración que llevará la primera línea reenumerada.
b: le suministra al ordenador la línea a partir de la que comienza la reenumeración.
c: incremento en la numeración entre dos líneas consecutivas.

Así, con RENUM 110,20,5 conseguiremos que:

1) La línea 20 pase a tener el número 110.
2) Y las restantes tendrán como numeración: 115, 120, 125,.....

Pueden faltar uno, dos o los tres parámetros de la instrucción RENUM. Siempre que falte uno se tomará como 10, en el caso de a y c, pero si falta b se supone que es la primera línea del programa.

EJEMPLO 1

Introducir, con AUTO, el siguiente programa:

```
10 PRINT "JUAN"
20 PRINT "ANA"
30 PRINT "CARLOS"
40 PRINT "INES"
```

Ahora, con RENUM 1,10,1 se reenumera a partir de la línea 10 empezando por darle el número 1 a la primera línea y a saltos de uno a uno, con lo que queda:

```
1 PRINT "JUAN"
2 PRINT "ANA"
3 PRINT "CARLOS"
4 PRINT "INES"
```

El mismo efecto se hubiera conseguido con RENUM 1,,1.
Partiendo del anterior programa, con RENUM 15, se obtendrá:

```
15 PRINT "JUAN"
25 PRINT "ANA"
35 PRINT "CARLOS"
4Ø PRINT "INES"
```

Si ahora empleamos RENUM sin ningún parámetro, el resultado será:

```
1Ø PRINT "JUAN"
2Ø PRINT "ANA"
3Ø PRINT "CARLOS"
4Ø PRINT "INES"
```

Con RENUM ,,2 reenumera a partir de la primera línea del programa, que ahora tendrá de numeración 10, siendo los saltos de 2 en 2. Queda:

```
1Ø PRINT "JUAN"
12 PRINT "ANA"
14 PRINT "CARLOS"
16 PRINT "INES"
```

La secuencia del programa original no puede ser cambiada con RENUM, es decir, si pretendemos reenumerar el programa anterior con la instrucción RENUM 1,14,1, estaremos intentando colocar por delante de las dos primeras líneas las dos últimas. Esto no está permitido, y el ordenador nos dará el mensaje de error:

```
Improper argument
```

Se presenta, también, el mismo mensaje de error, cuando al reenumerar se obtiene un número de línea fuera del margen permitido, que recordemos era de 1 a 65535.

5

Más cosas sobre PRINT

Como ya se dijo en el capítulo 2, en la pantalla se pueden escribir hasta 25 líneas de texto dependiendo el número de caracteres de cada línea del MODE elegido.

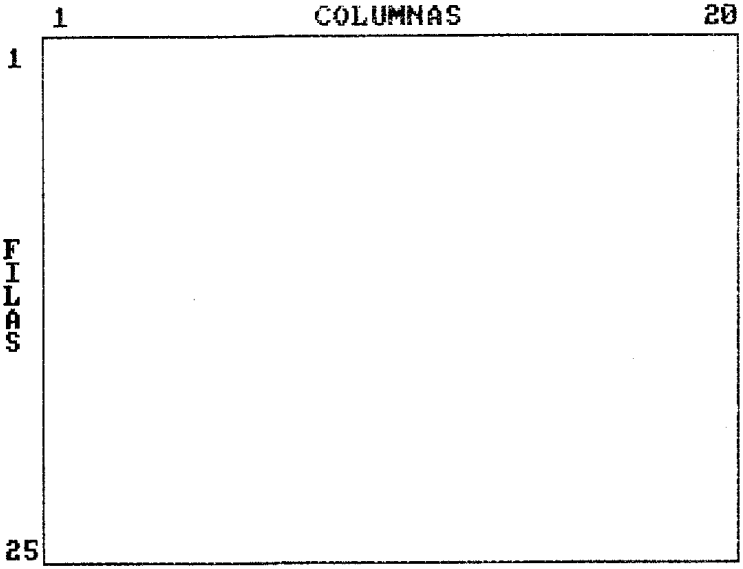
Para las siguientes instrucciones que vamos a ver ahora, tendremos que numerar las columnas y las filas. Como si de una cuartilla de papel se tratase, el origen a partir del cual numeramos estas columnas y filas, estará situado en la parte superior izquierda de la pantalla.

Las columnas se numerarán de izquierda a derecha:

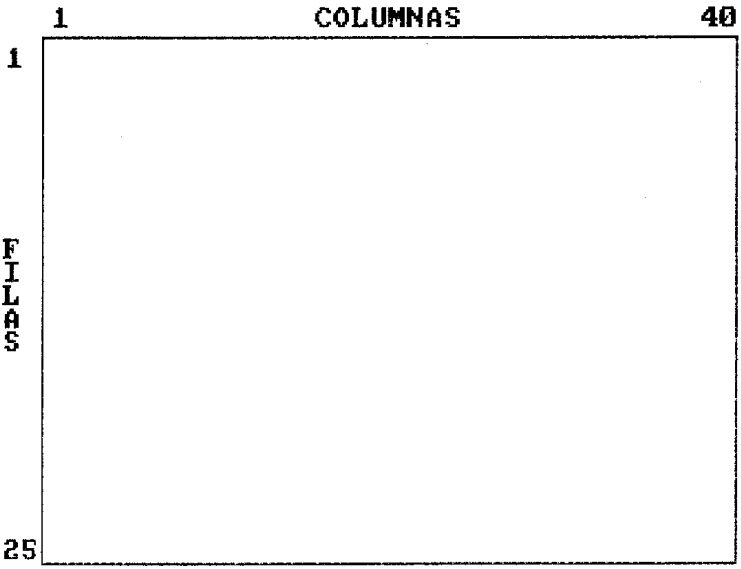
```
en  MODE Ø  de 1 a 2Ø

en  MODE 1  de 1 a 4Ø

en  MODE 2  de 1 a 8Ø
```

Las filas se numerarán de arriba–abajo y de 1 a 25 en cada uno de los tres modos de trabajo.

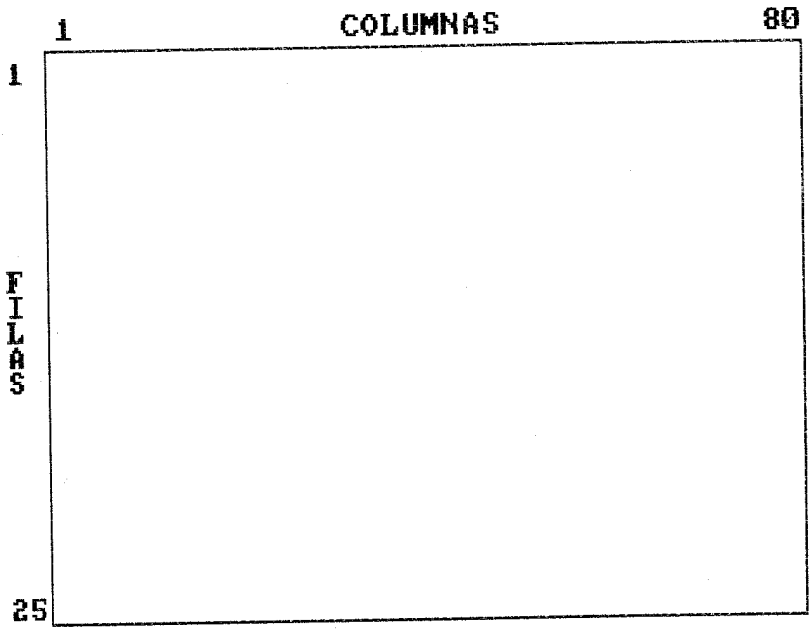
En MODE 0:



En MODE 1:



En MODE 2:



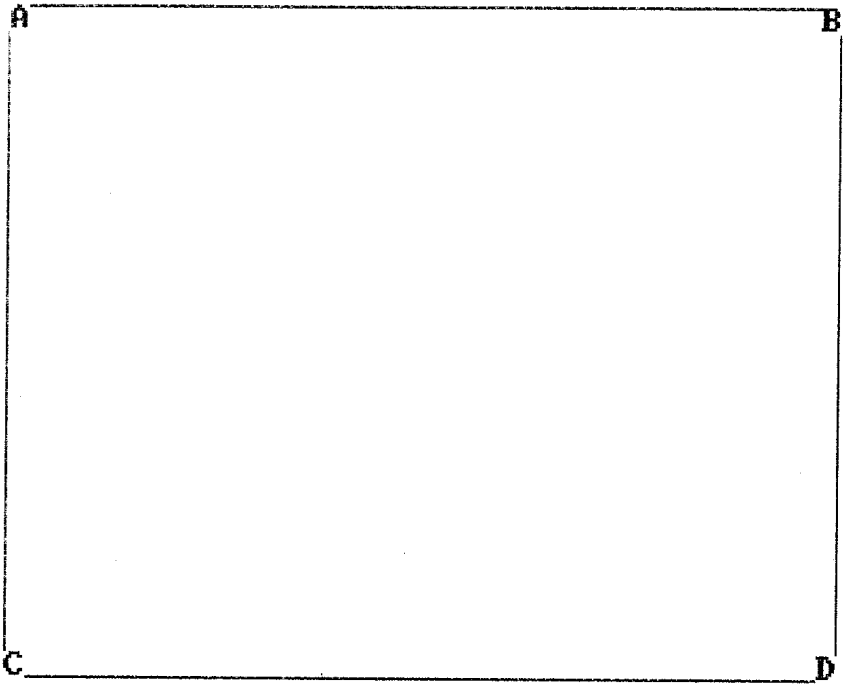
Vamos a ver ahora las coordenadas de los cuatro vértices de la zona de pantalla en donde es posible escribir. Escribamos una letra en cada uno de los vértices.

La letra "A" tendrá de coordenadas 1,1 para la columna y fila independientemente del MODE con el que se trabaje.

La letra "C" tendrá de coordenadas 1,25 para la columna y fila respectivamente.

La letra "B" tendrá siempre la coordenada 1 para la fila y 20,40 u 80 para la columna dependiendo de si está trabajando en MODE 0,1 ó 2.

La letra "D" tendrá siempre la coordenada 25 para la fila y 20,40 u 80 para la columna, dependiendo también del MODE 0,1 ó 2 elegido.



LOCATE (Colocar)

Su efecto es colocar el cursor de texto en la posición que le indiquemos, por lo que si ahora le pedimos al ordenador que imprima, lo hará a partir de esa posición.

Su forma general es:

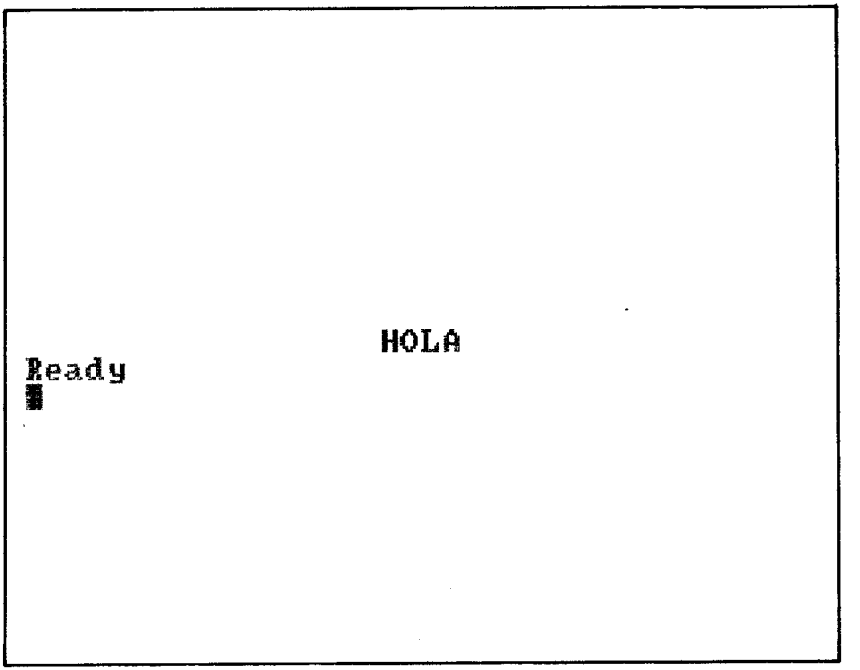
```
LOCATE C,F
```

donde C es la columna y F es la fila.

Así, si queremos imprimir el mensaje "HOLA" en el centro de la pantalla, lo lograremos con:

```
1Ø MODE 1
2Ø LOCATE 18,13:PRINT "HOLA"
```

con lo que en la pantalla aparecerá:



NOTAS

1) Si hubiéramos pretendido imprimir el mensaje anterior a partir de la columna 39 y en la fila 13 con:

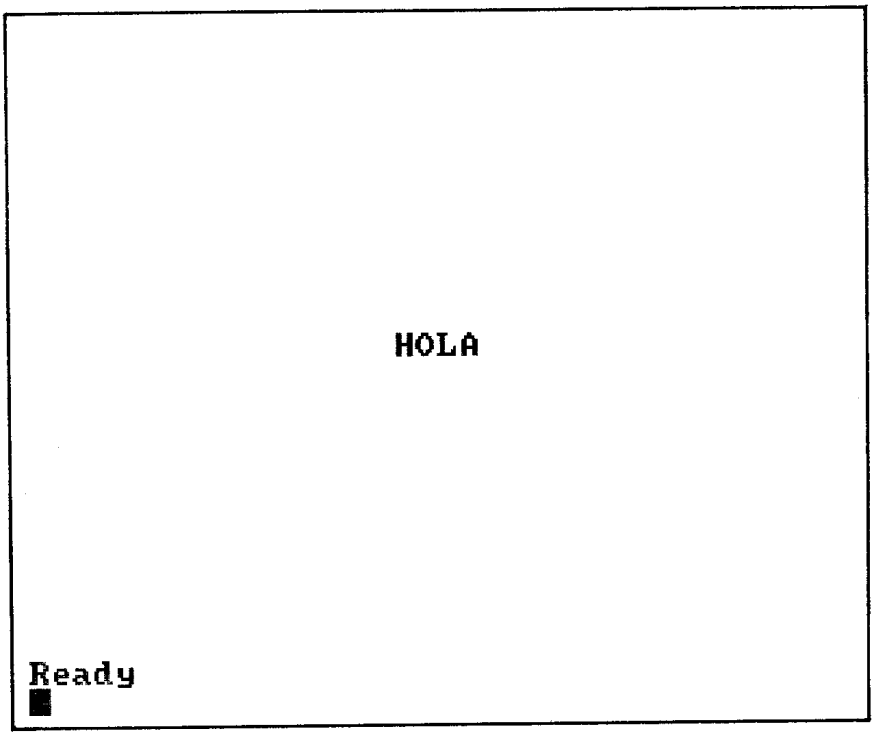
```
2Ø LOCATE 39,13:PRINT "HOLA"
```

al no tener espacio suficiente en la misma fila para imprimirse completo, se empezará a imprimir en el comienzo de la línea siguiente.

2) También puede usarse esta instrucción para colocar el mensaje de conformidad, READY, que aparece en numerosas ocasiones, en la posición en que queramos. En realidad, en este caso, sólo actuamos sobre la fila ya que la columna es siempre 1. Así, por ejemplo, si en el programa anterior, añadimos la línea:

```
3Ø LOCATE 1,25
```

en la pantalla aparecerá:



3) Si alguno de los números que colocamos detrás de LOCATE es negativo o cero, el ordenador dará el siguiente mensaje de error:

```
Improper argument in n
```

donde n es el número de línea en donde se encuentra LOCATE.

4) Sin embargo, si los números son positivos pero no enteros, el ordenador los redondea al entero más próximo.

```
20 LOCATE 0.5,3.2
```

será equivalente a

```
20 LOCATE 1,3
```

5) Si el número que indica la fila es mayor que 25, hará desaparecer una línea de arriba y el ordenador imprimirá en la de más abajo.

6) Si el número que indica la columna es mayor que el margen que hemos indicado, según sea el MODE empleado, entonces escribe en la columna 1 de la fila siguiente. Comprobarlo con:

```
10 MODE 1
20 PRINT "A"
30 LOCATE 60,2:PRINT "B"
```

7) CLS, al borrar la pantalla, coloca el cursor en 1,1.

TAB (TABulate: tabular)

Con esta instrucción, que debe ir precedida de PRINT, elegimos la columna donde imprimir. La fila será la que corresponda en ese momento.

La forma general es:

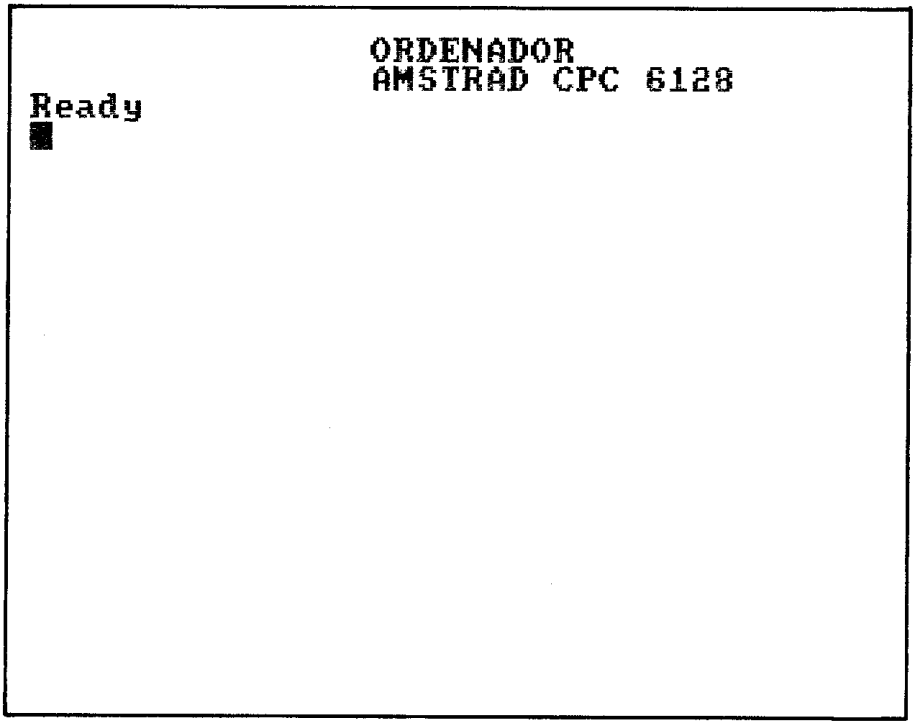
```
PRINT TAB (C);
```

donde C es el número de columna donde empezará a escribir. Detrás del punto y coma, que no es imprescindible ponerlo, irá la variable o la expresión a imprimir.

EJEMPLO 1

```
10 MODE 1
20 LOCATE 16,1:PRINT "ORDENADOR"
30 PRINT TAB(16);"AMSTRAD CPC 6128"
```

ejecutando este programa, obtenemos:



El mensaje de la línea 20 se ha impreso en la fila 1, luego la orden PRINT, de la línea 30, imprime en la fila siguiente. Con TAB (16) lo que hacemos es elegir la columna 16.

Esta instrucción es útil a la hora de presentar tablas en la pantalla.

NOTAS

1) Los valores 0 y negativos, no dan error tomándose como 1.

2) Si pretendemos escribir un mensaje a partir de una columna y no hay espacio suficiente en toda la línea, entonces empezará a imprimirse a partir de la columna primera de la línea siguiente.
Probar con:

```
1Ø MODE 1
2Ø PRINT TAB(39);"BASIC"
```

3) Si se emplean números decimales para la columna, el ordenador los redondea:

```
10 MODE 1
20 PRINT TAB(1.9);"BASIC"
30 PRINT TAB(1.3);"BASIC"
```

4) Si le damos al número de columna un número mayor que el margen permitido, el ordenador lo reduce a un valor posible. Así, en MODE 1, la columna:

   41 es equivalente a la 1
   42 es equivalente a la 2

   y sin modificar la fila.

SPC (SPaCe: espacio)

Su forma general es:

```
PRINT SPC(n)
```

y su efecto es el de dejar n espacios en blanco.

Si, por ejemplo, se introduce en el ordenador el programa:

```
10 MODE 1
20 PRINT "AAAAAAAA"
30 PRINT "B";SPC(6);"B"
```

aparecerá en la pantalla:

```
AAAAAAAA
B      B
Ready
```

NOTAS

1) El separador no es imprescindible usarlo en este caso, por lo que la línea 30 puede quedar:

```
PRINT "B" SPC(6) "B"
```

2) Todos los valores de n menores que 0 se tomarán como 0.

3) Si n es un número decimal positivo, el ordenador lo redondea al entero más próximo.

```
3Ø PRINT "A" SPC(2.3) "B" SPC(3.5) "C"
```

dejará 2 espacios entre la A y la B, y otros 4 espacios entre la B y la C.

4) Los espacios que, como máximo, se pueden dejar con SPC, no pueden llegar al número de caracteres que tiene una línea.

En MODE 0, el máximo es 19 y con SPC(20) escribe a continuación.
En MODE 1, el máximo es 39 y con SPC(40) escribe a continuación.
En MODE 2, el máximo es 79 y con SPC(80) escribe a continuación.

En MODE 1, por ejemplo, 40 espacios equivalen a 0, 41 espacios equivalen a 1, 42 a 2,.....

ZONE

En el capítulo 2 ya vimos el efecto que tenía el separador ","

En MODE 0 produce un salto de una línea.
En MODE 1 produce un salto de 1/3 de línea.
En MODE 2 produce un salto de 1/6 de línea.

Pero incluso sobre este salto se puede actuar, empleando la instrucción ZONE.

Así:

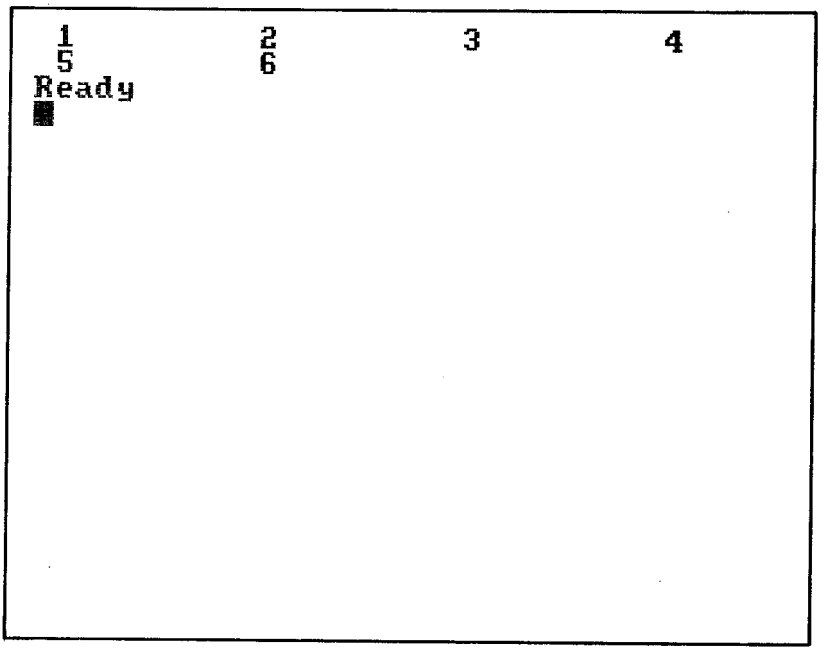
```
ZONE n
```

hace que cada vez que se encuentra una "," como separador, salte n columnas.

Por ejemplo, con el programa:

```
1Ø MODE 1
2Ø ZONE 1Ø
3Ø PRINT 1,2,3,4,5,6
```

obtenemos:



VENTANAS

Nuestro ordenador posee 10 canales diferentes para permitir la introducción y salida de datos. Estos canales se numeran del 0 al 9, ambos inclusive, y se distribuyen de la siguiente manera:

CANALES 0–7 se emplean para presentar el texto en pantalla.

CANAL 8 asignado a la impresora.

CANAL 9 asignado al disco o cassette.

Para utilizar un determinado canal hay que utilizar el símbolo: #

Así, con:

```
PRINT#8,"VENTANA
```

lograremos obtener por impresora la palabra VENTANA.

En este apartado, estudiaremos las diferentes posibilidades que se ofrecen para manejar los 8 primeros canales. Para ello utilizaremos la instrucción:

WINDOW

Cada canal, de los comprendidos entre 0 y 7, tiene asignada una ventana. Inicialmente estas 8 ventanas ocupan toda la pantalla de texto pero, con la instrucción WINDOW, podemos parcelar esta pantalla asignando a cada ventana el espacio que nos interesa.

La expresión general de la instrucción WINDOW es:

```
WINDOW#K,A,B,C,D
```

donde K es el número de la ventana (entre 0 y 7) y A,B,C,D indican las columnas (A y B) y las filas (C y D) que limitan la ventana.

Así, con:

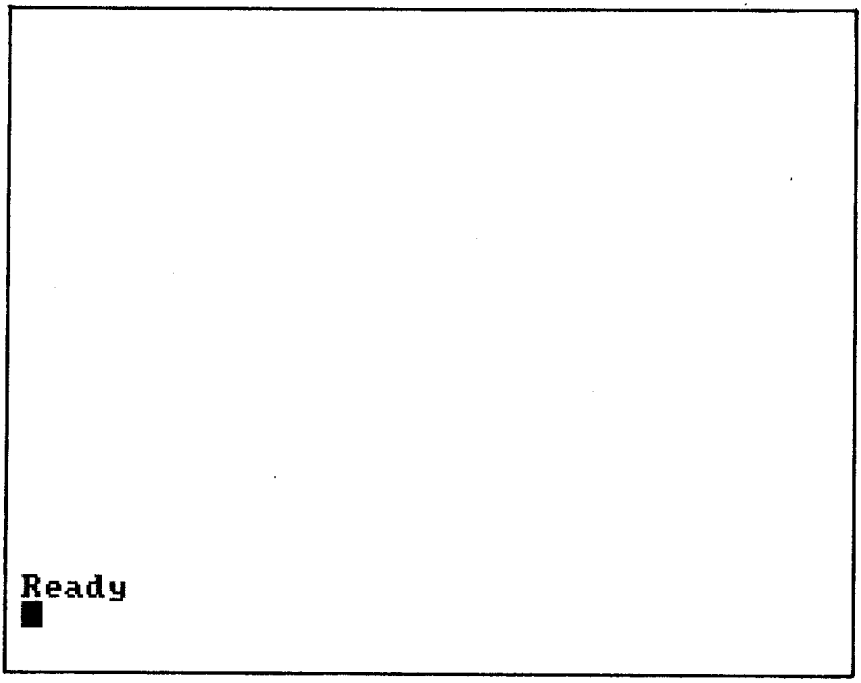
```
10 MODE 1
20 WINDOW# 0,1,40,23,25
```

hemos definido una ventana que ocupa las tres últimas filas de la pantalla. Esto se debe a que, al estar en MODE 1, disponemos de 25 filas y 40 columnas. Con los números 1,40 marcamos los límites de las pantallas en cuanto a columnas y con los números 23,25 hacemos lo mismo con las filas.

Con la línea 20 sólo hemos modificado la ventana 0. Todas las demás siguen ocupando toda la pantalla de texto. Como la ventana 0 es la que utiliza el ordenador para emitir sus mensajes, se habrá observado que

```
Ready
```

aparece ahora en la línea 23, en lugar de la primera.



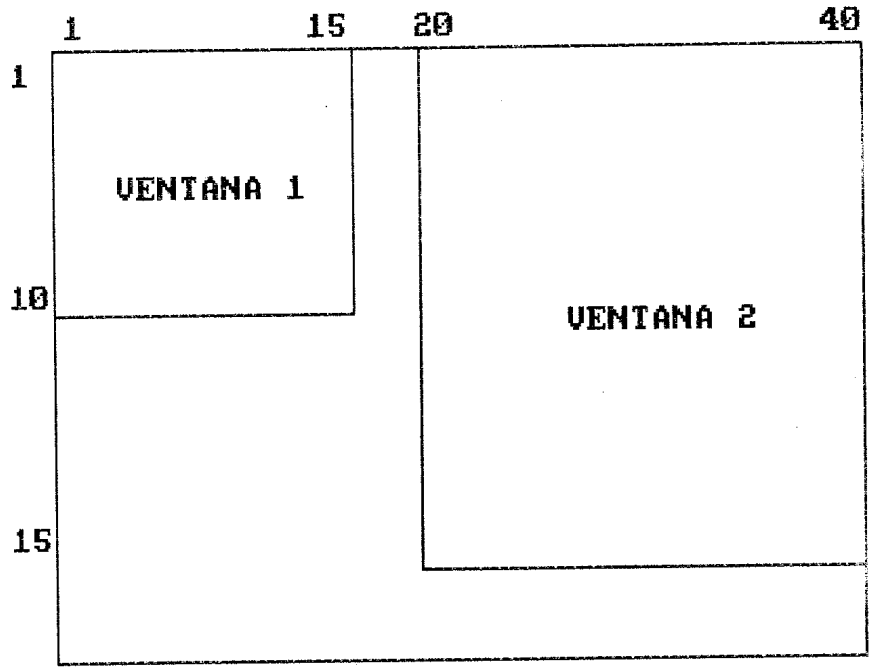
Cuando no se consigna el número de la ventana, el ordenador interpreta que se trata de la número 0; por lo que, la línea 20 anterior se puede escribir también de la forma:

```
20 WINDOW#0,1,40,23,25
```

De la misma forma, se pueden definir nuevas ventanas. Así con:

```
10 WINDOW#1,1,15,1,10
20 WINDOW#2,20,40,1,15
```

definen dos ventanas: la número 1 está limitada por las columnas 1,15 y las filas 1,10; la número 2 por las columnas 20,40 y las filas 1,15.



Siempre que queramos trabajar en una determinada ventana, será necesario indicar su número. Así, si queremos imprimir en cada una de las ventanas anteriores un mensaje, procederemos del siguiente modo:

```
40 PRINT "TAZA
50 PRINT#1,"RELOJ
60 PRINT#2,"REGLA
```

observar que, en la línea 40, al no indicar el número de ventana el ordenador considera que se trata de la número 0.

Para borrar una determinada ventana, hay que emplear la instrucción CLS acompañada del número de la ventana. Así, con:

```
70 CLS#2
```

borramos la palabra REGLA manteniendose en pantalla las otras.

También la instrucción LOCATE puede ir seguida del número de ventana pero, en este caso, hay que tener muy presente sus límites ya que, una vez marcada dicha ventana, el origen a partir del cual contamos las columnas y filas está situado en el extremo superior izquierdo de la ventana.

Con:

```
8Ø LOCATE#2,7,8
9Ø PRINT#2,"REGLA
```

volvemos a escribir la palabra REGLA en la ventana 2, pero ahora centrada.

NOTAS

1) Las diferentes ventanas se pueden solapar entre sí.

2) Existen otras instrucciones que también pueden asociarse a una ventana, como INPUT, LINE INPUT, LIST y otras que se verán más adelante. Por ejemplo, con LIST #2, obtenemos el listado del programa en la ventana número 2.

3) A veces, puede ser interesante intercambiar los números asociados a dos ventanas. Esto puede hacerse con la orden:

   ```
   WINDOW SWAP A,B
   ```

   donde A y B son los números de las ventanas que se cambian.

   Por ejemplo, con:

   ```
   WINDOW SWAP 1,2
   ```

   la ventana 1 pasa a ser la 2 y viceversa.

4) La instrucción MODE borra la configuración de todas las ventanas, quedando la pantalla en la situación inicial.

5) Para conocer la situación del cursor de texto en cada una de las ventanas existen las instrucciones siguientes:

   ```
   POS(#K)
   VPOS(#K)
   ```

   donde K es el número de ventana, que hay que indicar obligatoriamente, incluso en el caso K=0.

   Con POS(#K) obtenemos la columna en la que se halla el cursor en la ventana K.

   Con VPOS(#K) tenemos su fila.

6) Otra instrucción relacionada con ventanas es:

   ```
   COPYCHR$(#K)
   ```

Con ella obtenemos el carácter situado en la posición del cursor correspondiente a la ventana K. Por ejemplo:

```
1Ø MODE 1
2Ø WINDOW#3,1Ø,3Ø,5,2Ø
3Ø LOCATE#3,8,1Ø
4Ø PRINT#3,"MUY FACIL
5Ø LOCATE#3,12,1Ø
6Ø PRINT COPYCHR$(#3)
```

En la pantalla aparece la letra F ya que en la línea 50 se ha colocado el cursor sobre dicha letra. Como en la línea 60 no se especifica el número de ventana, el resultado se imprime en la ventana 0 que, al no estar definida, ocupa toda la pantalla.

En algunos modelos de AMSTRAD, no está definida esta instrucción.

PROGRAMAS PROPUESTOS

5.1) Escribir la palabra "AMSTRAD" centrada en la pantalla.

5.2) Construir un cuadrado de lado 6, formado por asteriscos, y centrado en la pantalla con los tres modos posibles.

5.3) Añadir las dos diagonales al cuadrado del ejercicio anterior.

5.4) Dibujar el siguiente árbol centrado en la pantalla.

```
          *
         ***
        *****
       *******
      *********
     ***********
    *************
   ***************
          * *
          * *
          * *
          * *
          ***
```

5.5) Definir 5 ventanas y colocar su número centrado en cada una de ellas. Ejecutar el programa y después listarlo.

6

Saltos

Ya hemos visto que un programa se realiza según el orden que marcan los números de línea que lo componen. Sin embargo, en muchas ocasiones es necesario alterar este orden para obtener los resultados que deseamos. Veamos por ejemplo:

Si queremos imprimir 10 veces la palabra "AGATA", según lo visto anteriormente, necesitamos 10 instrucciones PRINT:

```
1Ø PRINT "AGATA
2Ø PRINT "AGATA
3Ø PRINT "AGATA
4Ø PRINT "AGATA
5Ø PRINT "AGATA
6Ø PRINT "AGATA
7Ø PRINT "AGATA
8Ø PRINT "AGATA
9Ø PRINT "AGATA
1ØØ PRINT "AGATA
```

Evidentemente, este procedimiento no es muy práctico; sería interesante alguna instrucción que permitiera volver 10 veces a la línea

```
1Ø PRINT "AGATA
```

Esto lo podemos hacer mediante lo que en BASIC se denominan SALTOS, que pueden ser de dos tipos:

a) saltos incondicionales.
b) saltos condicionales.

Este capítulo lo dedicaremos integramente a estudiar en profundidad las instrucciones necesarias para efectuar estos saltos.

SALTOS INCONDICIONALES

Llamamos saltos incondicionales a aquellos que realiza el ordenador siempre que el proceso de desarrollo del programa llega a una determinada instrucción, sin depender de ninguna condición.

Hay varias instrucciones que realizan estos saltos actuando de formas muy distintas. Es interesante conocerlas bien ya que ellas eliminarán gran cantidad de dificultades a la hora de realizar nuestros programas.

GOTO (ir a)

La expresión general de esta instrucción es:

```
GOTO N
```

donde N tiene que coincidir necesariamente con un número de línea del programa.

El efecto de esta instrucción es el de conseguir que la siguiente línea que se ejecute sea la N.

EJEMPLO 1

```
1Ø A=12:B=14
2Ø GOTO 4Ø
3Ø PRINT A
4Ø PRINT B
```

En este caso, después de haber definido A=12 y B=14, al llegar a la línea 20 se efectúa un salto a la línea 40. Como en esta línea se manda imprimir el valor de B, en pantalla aparece el número 14.

En este pequeño ejemplo, hemos visto como se ha realizado un salto hacia delante; pero la instrucción GOTO también realiza saltos hacia atrás.

EJEMPLO 2

```
1Ø PRINT "AGATA";
2Ø GOTO 1Ø
```

En la línea 10 se imprime la palabra AGATA y, en la 20 se efectúa un salto a la 10, por lo que se repetirá la impresión. Este proceso seguirá indefini-

damente hasta que nosotros lo paremos desde el teclado presionando la tecla:

ESC

¿Qué efecto producirá una línea del tipo 10 GOTO 10?

NOTA

En otros ordenadores, si se da instrucción GOTO N y la línea N no existe en el programa, se ejecuta la marcada con el número mayor más próximo a N. En el ordenador AMSTRAD, si N no existe, presenta el mensaje de error:

```
Line does not exist
```

GOSUB

Su forma general es:

```
GOSUB N
```

donde N tiene que ser un número de línea del programa. Esta instrucción nos permite realizar un SALTO CON RETORNO; es decir, se salta a la línea N, a partir de la cual se sigue ejecutando el programa hasta llegar a la instrucción RETURN (retorno), que produce el regreso a la instrucción siguiente a GOSUB N.

A la parte del programa formado por las líneas comprendidas entre la número N y RETURN se le llama SUBRUTINA.

El siguiente gráfico puede ayudarnos a comprender esta instrucción:

```
línea P    GOSUB N   |  línea N    REM SUBRUTINA
línea Q              |
                     |  línea M    RETURN
```

Cuando el ordenador alcanza la línea P, al encontrarse con GOSUB N, salta a la línea N. Ahora ejecuta las líneas comprendidas entre la N y la M y, al encontrarse con RETURN, vuelve a la línea Q, que es la siguiente a la P.

La única forma de acceso a una subrutina es mediante GOSUB ya que, en caso contrario, al llegar a RETURN presenta el mensaje de error:

```
Unexpected RETURN in M
```

Esta instrucción se emplea cuando a lo largo de un programa hay que ejecutar en repetidas ocasiones un mismo proceso.

Operadores de relación

Antes de pasar a estudiar las diferentes instrucciones que permiten realizar saltos condicionales, introduciremos un conjunto de operadores que serán de gran utilidad.

Hay seis operadores que son los siguientes:

```
=  ,  >  ,  <  ,  <=  ,  >=  ,  <>
```

siendo su significado:

```
A = B      A es igual a B.
A > B      A es mayor que B.
A < B      A es menor que B.
A <= B     A es menor o igual que B.
A >= B     A es mayor o igual que B.
A <> B     A es distinto de B.
```

En un principio, parece lógico que estos operadores sólo los podamos emplear con números o variables numéricas; sin embargo, más adelante veremos que también será posible utilizarlos con variables alfanuméricas.

NOTAS

1) El ordenador nos dará el mensaje de error:

```
Syntax error
```

si en lugar de escribir <> , escribimos >< .

2) Si una relación es cierta se le asigna el valor –1 y si es falsa el valor 0. Así,

```
PRINT A<B
```

imprime:

```
-1    si A<B
 Ø    si A>=B
```

OPERADORES LOGICOS

Con los operadores de relación ya podemos establecer ciertas condiciones. Por ejemplo, si queremos expresar que A tiene que ser menor que 3, bastará escribir:

```
A<3
```

En muchos casos no es suficiente con una sola condición y es necesario establecer alguna forma de relacionar varias de ellas. Supongamos que queremos expresar que A sea menor que 3 y además mayor que 0; para conseguirlo tendremos que unir las dos condiciones en una sola.

AND (y)

Si llamamos C1 y C2 a dos condiciones.

```
C1 AND C2
```

es otra condición que únicamente es cierta cuando es cierta C1 y además es cierta C2. En caso contrario será falsa.

OR (o)

La condición:

```
C1 OR C2
```

será cierta siempre que alguna de las dos condiciones, C1 ó C2, lo sea. Unicamente será falsa si ambas condiciones son falsas.

XOR (o exclusiva)

La condición:

```
C1 XOR C2
```

es falsa si C1 y C2 son las dos ciertas o las dos falsas.

NOT (negación)

La condición:

```
NOT C1
```

es verdadera cuando C1 es falsa y falsa cuando C1 es verdadera.

La siguiente tabla, donde la letra V indica verdadero y la letra F falso, nos ayudará a comprender lo dicho anteriormente.

C1	C2	C1 AND C2	C1 OR C2	C1 XOR C2	NOT C1
V	V	V	V	F	F
V	F	F	V	V	F
F	V	F	V	V	V
F	F	F	F	F	V

Igual que sucedía con los operadores aritméticos, si manejamos diferentes operadores lógicos existe una orden de prioridad entre ellos:

1) NOT
2) AND
3) OR
4) XOR

Este orden de prioridad puede alterarse mediante el uso de paréntesis. Así, si suponemos que:

```
A=3          B=5          C=8
```

la relación:

```
A<2  AND  B<3  OR  C>5
```

será verdadera ya que primero se considera A<2 AND B<3 que llamaremos R1 y que es falsa; y depués se analiza R1 OR C>5 que es cierta.

Si hubieramos utilizado paréntesis en la forma:

```
A<2  AND (B<3  OR  C>5)
```

primero se analizará la condición incluida entre paréntesis, que evidentemente es verdadera. Si llamamos a esta relación R2, ahora se consideraría A<2 AND R2 que es falsa.

Después de conocer estos operadores, pasaremos a estudiar las diferentes instrucciones que realizan saltos condicionales.

IF (si)

La forma general de esta instrucción es cualquiera de las siguientes:

```
IF condición THEN n
IF condición THEN GOTO n
IF condición GOTO n
```

siendo n el número de una línea del programa.

Con esta orden lograremos que se efectúe un salto a la línea n del programa siempre que la condición se cumpla; en caso contrario, el programa sigue su curso normal.

EJEMPLO 3

```
10 MODE 1
20 INPUT N
30 PRINT N
40 IF N<10 THEN 20
```

al ejecutar este programa, en la línea 20, se nos pide que introduzcamos desde el teclado un número N. A continuación se imprimirá en pantalla y finalmente, si el número es menor que 10, el ordenador volverá a la línea 20. En caso de que el número sea mayor que 10 el programa finaliza.

EJEMPLO 4

Ya podemos resolver el problema de imprimir en pantalla 10 veces la palabra "AGATA".

```
10 PRINT "AGATA
20 C=C+1
30 IF C<10 THEN 10
```

cada vez que imprimimos la palabra "AGATA", la variable C, que llamamos contador, se incrementa en una unidad. Mientras el valor de C sea menor que 10, al llegar a la línea 30 se efectuará un salto a la 10; cuando C=10 el programa se detiene.

Aunque muchas veces tras la instrucción IF se indica un salto, también pueden colocarse otras instrucciones que se ejecutarán cuando la condición

se verifique. Así, el programa del ejemplo anterior podría escribirse en la forma:

```
10 C=C+1
20 PRINT "AGATA
30 IF C=10 THEN END
40 GOTO 10
```

en la línea 30 exigimos que el desarrollo del programa se pare si C=10.

Hay que tener en cuenta que si en una línea tenemos una condición IF, si es falsa no se ejecuta nada de lo que hay a su derecha. Por ejemplo, una línea como:

```
IF A>0 THEN CLS:PRINT A:STOP
```

obligará a borrar la pantalla, imprimir A y parar el programa si A es positivo; en caso contrario, no se realizará ninguna de estas acciones.

Hasta ahora, las condiciones que hemos puesto detrás de IF han sido simples, pero no hay ningún inconveniente en utilizar varias de ellas con la ayuda de operadores lógicos.

EJEMPLO 5

Vamos a calcular la nota media de las calificaciones de un grupo de alumnos. Como desconocemos el número de alumnos, adoptaremos el convenio de que cuando una calificación esté fuera del margen 0–10, el programa se pare:

```
10 MODE 1
20 INPUT "CALIFICACION";N
30 IF N<0 OR N>10 THEN PRINT "NOTA MEDIA"
;S/C:END
40 C=C+1:S=S+N:GOTO 20
```

en la línea 20 se piden las calificaciones y, siempre que éstas estén comprendidas entre 0 y 10, repite el proceso. La variable C se encarga de contabilizar el número de notas introducidas y la variable S acumula su suma.

EJEMPLO 6

Pretendemos, con el empleo de una subrutina, que el ordenador efectúe pausas de duración determinadas.

```
10 MODE 1
20 INPUT "PAUSA EN SEGUNDOS ";P
30 GOSUB 100
40 PRINT "SUBRUTINA EJECUTADA POR PRIMERA
VEZ"
50 INPUT "PAUSA EN SEGUNDOS ";P
60 GOSUB 100
70 PRINT "Y AHORA POR SEGUNDA VEZ"
80 END
100 REM SUBRUTINA-PAUSA
110 P=P*300
120 P=P-1
130 IF P>0 THEN 120
140 RETURN
```

NOTAS AL PROGRAMA

1) Se ejecuta la misma subrutina en dos ocasiones, aunque la pausa no tiene por que tener la misma duración ya que la variable P puede tomar valores distintos.

2) Para que el ordenador efectúe una pausa de P segundos aproximadamente, hay que multiplicar P por 300 (línea 110).

3) Ya hemos comentado que al acceso a la subrutina debe hacerse siempre con GOSUB; por ello hemos colocado la línea:

```
80 END
```

en caso contrario, después de llamar por segunda vez a la subrutina, se ejecutará la línea 100 y siguientes y, al llegar a RETURN, presentaría el mensaje de error:

```
Unexpected RETURN in 140
```

ELSE

Siempre que se presente esta instrucción debe ir acompañando a IF. Su forma general es:

```
IF condición THEN instrucciones1 ELSE in
strucciones2
```

y su funcionamiento es el siguiente:

Si la condición se cumple entonces se ejecutan las instrucciones1. En caso contrario se ejecutan las instrucciones2.

Por ejemplo:

```
IF A>Ø THEN PRINT A ELSE CLS
```

si A es mayor que cero es imprime A; en caso contrario, se borra la pantalla.

EJEMPLO 7

Mediante el siguiente programa se ordenan de menor a mayor dos números dados:

```
1Ø MODE 1
2Ø INPUT "DAME DOS NUMEROS ";A,B
3Ø IF A<B THEN PRINT A;B ELSE PRINT B;A
```

EJEMPLO 8

Con el siguiente programa el ordenador nos informa de cuantos números hay iguales de entre los tres introducidos:

```
1Ø INPUT A,B,C
2Ø IF A=B AND B=C THEN PRINT 3:END
3Ø IF A=B OR A=C OR B=C THEN PRINT 2 EL
SE PRINT 1
```

ON GOTO

Su expresión general es:

```
ON A GOTO M,N,P,....
```

donde M,N,P,.... son números de línea del programa.

Con esta instrucción logramos que el salto a realizar dependa del valor de A:

```
si A=1 se salta a la línea M
si A=2 se salta a la línea N
si A=3 se salta a la línea P
```

Para ver el funcionamiento de esta orden, introducir:

```
10 INPUT "VALOR DE A ";A
20 PRINT "HA SALTADO A LA LINEA";
30 ON A GOTO 100,150,90,50
40 END
50 PRINT 50:GOTO 10
90 PRINT 90:GOTO 10
100 PRINT 100:GOTO 10
150 PRINT 150:GOTO 10
```

Es necesario colocar la línea 40 ya que si A se encuentra fuera del margen 1–4 no se efectúa ningún salto.

Observar también que las líneas introducidas detrás de ON–GOTO puede colocarse en cualquier orden.

Especialmente útil puede resultar esta orden en aquellos programas que presentan un menú de opciones.

EJEMPLO 9

Un programa fichero que nos lleve el control de clientes, direcciones, teléfonos, etc., podría empezar así:

```
10 MODE 1
20 LOCATE 14,3:PRINT "MENU"
30 LOCATE 8,7:PRINT "1 - NUEVAS FICHAS"
40 LOCATE 8,9:PRINT "2 - BORRAR"
50 LOCATE 8,11:PRINT "3 - LEER"
60 LOCATE 8,13:PRINT "4 - FIN"
70 LOCATE 8,17
80 INPUT "OPCION ";O
90 ON O GOTO 200,400,500,1000
100 GOTO 10
200 CLS:LOCATE 4,13
210 PRINT "HA ELEGIDO LA OPCION 1"
220 GOSUB 2000
230 GOTO 10
400 CLS:LOCATE 4,13
410 PRINT "HA ELEGIDO LA OPCION 2"
420 GOSUB 2000
430 GOTO 10
500 CLS:LOCATE 4,13
```

```
510 PRINT "HA ELEGIDO LA OPCION 3"
520 GOSUB 2000
530 GOTO 10
1000 CLS:LOCATE 4,13
1010 PRINT "HA ELEGIDO LA OPCION 4"
1020 LOCATE 1,23:END
2000 REM PAUSA
2010 C=0
2020 C=C+1
2030 IF C<500 THEN 2020
2040 RETURN
```

NOTAS AL LISTADO

1) En la línea 70 se coloca el cursor en la posición 8,17 porque es ahí donde queremos que aparezca la pregunta.

2) En la línea 90 se realiza el salto en función de la opción elegida.

3) La subrutina realiza una pausa para darnos tiempo a leer el mensaje antes de volver al menú.

ON GOSUB

La expresión general es:

```
ON A GOSUB M,N,P,....
```

donde M,N,P,.... son números de línea del programa.

Su funcionamiento es análogo al de ON–GOTO, efectuándose saltos a las subrutinas que empiezan en las líneas M,N,P, etc.

ON BREAK

Sabido es que se puede parar la ejecución de un programa pulsando dos veces la tecla ESC. Cuando realizamos esta operación en la pantalla aparece el mensaje:

```
Break in N
```

que indica que hemos detenido el programa en la línea N.

Con la instrucción ON BREAK logramos que al pulsar dos veces la tecla ESC el resultado no sea el indicado anteriormente.

Esta orden tiene tres posibles formas diferentes, con

```
ON BREAK GOSUB N
```

conseguiremos que al parar el programa desde el teclado, se realice la subrutina que comienza en la línea N.

EJEMPLO 10

```
10 ON BREAK GOSUB 100
20 PRINT "A";:GOTO 20
100 REM SUBRUTINA
110 PRINT "DIFICIL DE PARAR"
120 RETURN
```

Hay que tener en cuenta, que una vez que se ha ejecutado la lína 10, en cualquier momento que se intente parar el programa se efectuará el salto a la subrutina de la línea 100.

Otra forma posible de utilizar ON BREAK es

```
ON BREAK CONT
```

que hace que el programa no pueda detenerse mediante la pulsación de la tecla ESC. Si el programa no se detiene por sí mismo, no se conseguirá de otra forma.

EJEMPLO 11

```
10 ON BREAK CONT
20 PRINT "A";
30 C=C+1
40 IF C=1000 THEN END ELSE GOTO 20
```

en este caso, el programa no se detendrá hasta que C valga 1000.

Finalmente, la tercera forma es:

```
ON BREAK STOP
```

y cuya función es anular cualquiera de las dos anteriores.

EJEMPLO 12

```
1Ø ON BREAK CONT
2Ø PRINT Ø
3Ø C=C+1
4Ø IF C<=5ØØ THEN 2Ø
5Ø ON BREAK STOP
6Ø PRINT "*";
7Ø D=D+1
8Ø IF D=1ØØØ THEN END ELSE 6Ø
```

se observará que en la primera parte del programa es imposible detenerlo desde el teclado. Sin embargo, en la segunda parte, al pulsar dos veces la tecla ESC, el programa se detiene.

ON ERROR GOTO

Su expresión general es:

```
ON ERROR GOTO N
```

donde N es un número de línea del programa.

Con esta orden logramos que una vez que el ordenador ha pasado por ella la recuerde y, en el momento que se produzca un ERROR, se salte a la línea N.

Es interesante colocar en la línea N las dos variables siguientes:

ERL esta variable indica el número de línea donde se ha producido el error

ERR con esta variable conseguiremos que nos indique el tipo de error.

Por tanto, la forma más usual de utilizar la instrucción ON ERROR GOTO es:

```
1Ø ON ERROR GOTO 1ØØØ
..... PROGRAMA .....
1ØØØ PRINT "ERROR";ERR;"EN LA LINEA";ERL
```

La instrucción ON ERROR GOTO 0 anula la anterior.

RESUME

Esta orden tiene como misión reanudar la ejecución de un programa después de detectarse un error con

```
ON GOTO ERROR
```

admite tres modalidades:

RESUME N reanuda el programa a partir de la línea N.

RESUME reanuda el programa a partir de la instrucción donde se había producido el error.

RESUME NEXT reanuda el programa a partir de la instrucción siguiente a aquella donde se ha producido el error.

Si el último ejemplo le añadimos:

```
1Ø1Ø RESUME
```

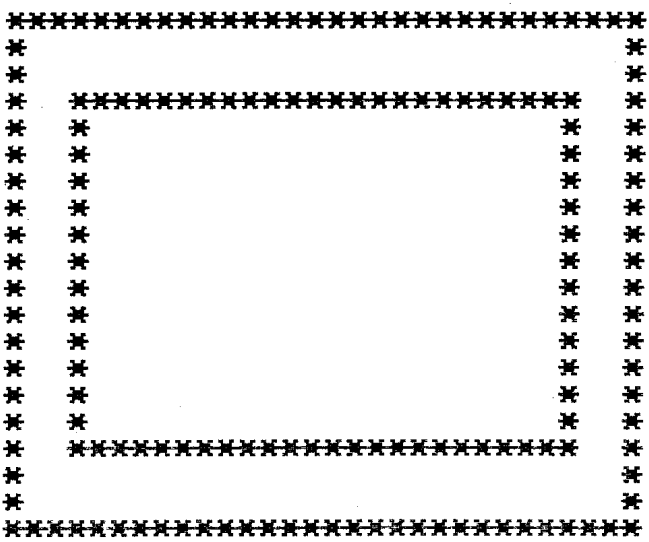
el programa se reanudará después de indicarnos el tipo de error y la línea donde se ha producido.

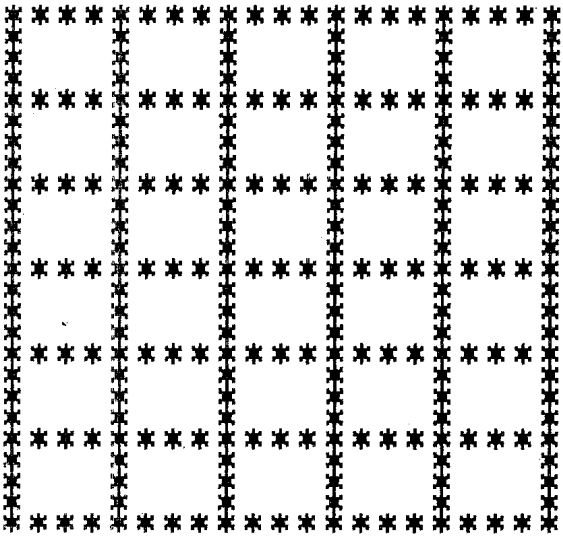
PROGRAMAS PROPUESTOS

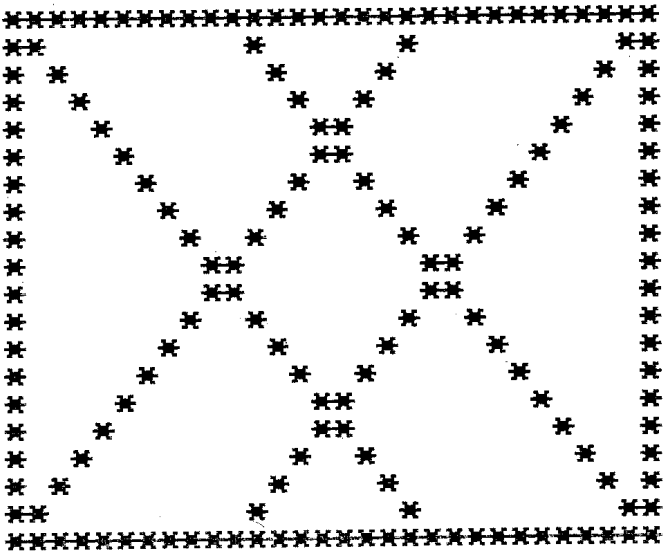
1) Hacer un programa que presente en pantalla la tabla de multiplicar de cualquier número.

2) Hacer un programa para resolver ecuaciones de segundo grado.

3) Dibujar en la pantalla con el símbolo "*" las siguientes gráficas:







4) Dados 10 números, obtener el mayor y el menor de ellos.

5) Dado el número de un mes escribir su nombre (utilizar ON GOSUB)

6) Supongamos que el ordenador no tiene posibilidad de multiplicar ni de calcular potencias. Hacer un programa que permita obtener el valor A elevado a N, siendo ambos números enteros, mediante sumas repetidas. Emplear una subrutina para calcular las diferentes sumas parciales.

7

Bucles

En el capítulo anterior veíamos como repetir un mismo proceso varias veces, utilizando un contador y un condicional; por ejemplo, si se quiere imprimir la palabra "BUCLE" doce veces, introduciremos el programa:

```
1Ø C=C+1
2Ø PRINT "BUCLE"
3Ø IF C=12 THEN END
4Ø GOTO 1Ø
```

En BASIC se puede simplificar este proceso mediante el empleo de los llamados bucles FOR–NEXT.

BUCLES FOR–NEXT

Su forma general es:

```
FOR contador=inicio TO fin
          .
          .
proceso a repetir
          .
          .
NEXT contador
```

que, en castellano, quiere decir:

```
DESDE contador=inicio HASTA fin
          .
          .
proceso a repetir
          .
          .
SIGUIENTE valor del contador
```

con estas nuevas instrucciones, el programa anterior queda así:

```
10 FOR C=1 TO 12
20 PRINT "BUCLE"
30 NEXT C
```

Sigamos paso a paso el desarrollo del programa:

1) En la línea 10, se le asigna al contador C, que ahora llamaremos VARIABLE DE CONTROL DEL BUCLE, un valor de PARTIDA y un valor LIMITE. En nuestro ejemplo, estos valores son 1 y 12 respectivamente.
2) Se imprime la palabra "BUCLE".
3) En la línea 30 se le asigna a C su siguiente valor, el 2, y se compara con el valor LIMITE (12).
4) Mientras el valor C no sea superior al LIMITE, se repite el proceso a partir del paso segundo.
5) Cuando el valor C es mayor que el LIMITE, se dá por finalizado el bucle y el programa continúa su ejecución a partir de la línea siguiente a la 30.

```
 1
 4
 9
 16
 25
 36
 49
 64
 81
 100
Ready
```

La principal ventaja que ofrecen los bucles FOR–NEXT, además de permitirnos una mayor estructuración de los programas, es la de permitir que la variable de control del bucle aparezca también en el proceso a repetirse. Por ejemplo, con el siguiente programa.

```
10 FOR C=1 TO 10
20 PRINT C*C
30 NEXT C
```

se imprimirán en la pantalla, (ver página anterior) los cuadrados de todos los números entre 1 y 10.

Observar que la variable C, además de servir de contador, también se vé involucrada en el proceso de cálculo.

NOTAS

1) El nombre de la variable de control puede constar de más de un carácter

2) Se consigue mayor rapidez en la ejecución de los bucles utilizando, siempre que sea posible, variables enteras.

   Compruébalo con los siguientes programas:

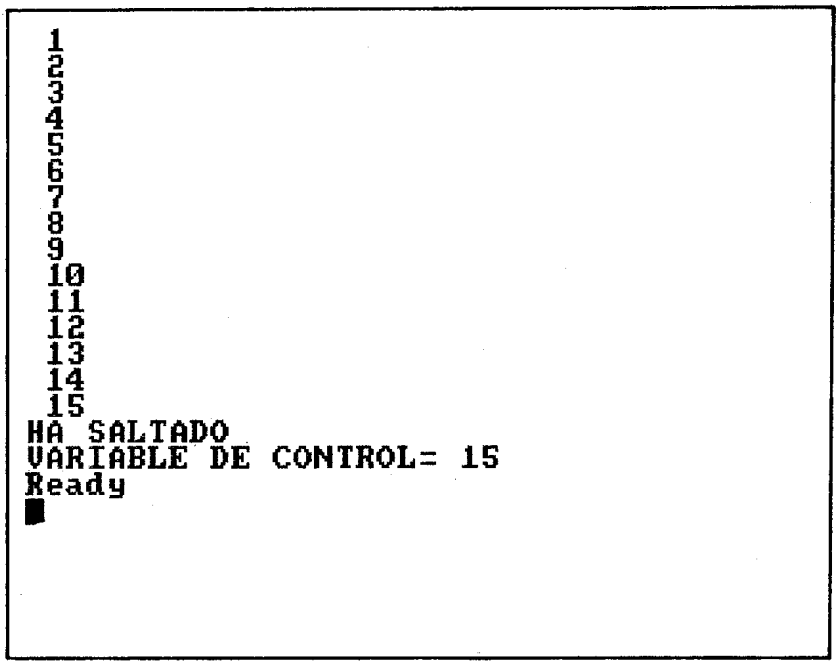
```
10 FOR a=1 TO 10000
20 NEXT a

10 FOR a%=1 TO 10000
20 NEXT a%
```

   Observar que, aproximadamente, tardan 11 y 5 segundos respectivamente.

3) Se puede salir de un bucle conservándose el valor de la variable control, así, por ejemplo, con:

```
10 FOR x=1 TO 100
20 PRINT x
30 IF x=15 THEN 60
40 NEXT x
50 STOP
60 PRINT "HA SALTADO"
70 PRINT "VARIABLE DE CONTROL=";x
```

   obtenemos:



4) No es posible entrar, con GOTO o GOSUB, dentro de un bucle. Por ejemplo, al ejecutar:

```
10 GOTO 30
20 FOR x=1 TO 100
30 PRINT x
40 NEXT x
```

el ordenador presenta el mensaje de error:

```
Unexpected NEXT in 40
```

ya que encuentra un NEXT sin haber pasado antes por FOR. Cuando encuentra un FOR sin el NEXT correspondiente, el mensaje de error es:

```
NEXT missing
```

5) Hasta ahora, en todos los programas hemos escrito NEXT c, NEXT a etc,..., es decir, NEXT acompañado de la variable de control del bucle. Para evitarnos trabajo, nuestro ordenador nos permite prescindir de ella, poniendo simplemente NEXT.

6) Es posible modificar el valor de la variable de control dentro de un bucle. Por ejemplo:

```
10 FOR x=1 TO 100
20 x=x+1: PRINT x
30 NEXT
```

imprime sólo los números pares.

7) Si el número LIMITE es inferior al valor de PARTIDA, y con sólo las instrucciones que hemos visto hasta ahora, no se ejecutará el bucle. Comprobarlo con:

```
10 FOR x=100 TO 1
20 PRINT
30 NEXT
```

8) Con un bucle FOR–NEXT, también podemos efectuar una pausa, por ejemplo con:

```
10 FOR x=1 TO 1000:NEXT
```

conseguimos una pausa de algo más de un segundo.

9) Los valores de PARTIDA y LIMITE pueden ser los valores que, en un determinado momento tomen ciertas variables. Por ejemplo:

```
10 INPUT A
20 INPUT B
30 IF B<A THEN 10
40 FOR X=A TO B
50 PRINT X
60 NEXT
```

10) Los bucles FOR–NEXT también pueden ejecutarse en MODO DIRECTO, es decir, sin número de línea.

```
FOR X=1 TO 10:PRINT X:NEXT
```

11) También podrán ir detrás de una condición para que sólo si esa condición se cumple, el bucle se ejecute. Así, en el programa:

```
10 INPUT x
20 IF x=10 THEN FOR a=1 TO 10000:NEXT
```

sólo se efectuará la pausa si el valor introducido es 10.

STEP (Incremento)

Hasta ahora, la variable de control del bucle se incrementaba siempre en una unidad. Sin embargo, también podemos actuar sobre este incremento haciendo uso de la nueva instrucción STEP, que se introduce tras el valor LIMITE.

```
FOR control=PARTIDA TO LIMITE STEP incremento
```

Por ejemplo, si se desea imprimir todos los números pares inferiores a 100, introduciremos el programa siguiente:

```
10 FOR x=2 TO 98 STEP 2
20 PRINT x
30 NEXT
```

El salto podrá ser negativo, decimal e incluso cero.

Con el programa:

```
10 FOR x=98 TO 2 STEP -2
20 PRINT x
30 NEXT
```

obtendremos también los números pares, pero de mayor a menor.

Con el programa:

```
10 FOR x=1 TO 2 STEP 0.1
20 PRINT x
30 NEXT
```

se imprimirán los números decimales entre el 1 y el 2, de décima en décima.

NOTAS

1) Al igual que ocurría con los parámetros de partida y límite, también el incremento puede ser el valor que toma en ese momento una determinada variable.

2) Si el incremento es negativo y el límite es mayor que el valor de partida, el bucle no se ejecuta. Comprobarlo con:

```
10 FOR x=1 TO 12 STEP -3
20 PRINT x
30 NEXT
```

3) De igual manera tampoco se ejecuta el bucle cuando, siendo el incremento positivo, el valor límite es inferior al de partida.

4) Si se toma como incremento el valor 0, se obtiene un bucle infinito que no terminará nunca, ya que el valor límite nunca se alcanza.

```
10 FOR x=1 TO 3 STEP 0
20 PRINT x
30 NEXT
```

5) La variable de control toma siempre el valor de partida pero no tiene por qué alcanzar exactamente el valor límite. Comprobarlo con:

```
10 FOR x=1 TO 12 STEP 2
20 PRINT x
30 NEXT
```

EJEMPLO 1

Con el siguiente programa vamos a calcular la suma de los N primeros números enteros, partiendo del 1.

```
10 INPUT N
20 IF N<=0 THEN 10
30 FOR X=1 TO N
40 S=S+X
50 NEXT
60 PRINT "LA SUMA ES ";S
```

en la línea 40, la variable S va alcanzando las diversas sumas parciales.

EJEMPLO 2

El problema anterior se puede modificar de forma que se pida de qué número hasta qué otro queremos sumar.

```
10 INPUT A
20 INPUT B
30 IF B<A THEN 10
40 FOR X=A TO B
50 S=S+X
60 NEXT X
70 PRINT "LA SUMA DE";A;"HASTA";B;"ES";S
```

EJEMPLO 3

Vamos a construir un programa para sacar en pantalla la tabla de multiplicar de un número dado.

```
10 MODE 1
20 INPUT "TABLA DEL ";N
30 IF N<1 OR N>9 THEN 10
40 CLS
50 PRINT "TABLA DEL";N
60 PRINT
70 FOR X=1 TO 9
80 PRINT
90 PRINT N;"X";X;"=";N*X
100 NEXT
```

que, por ejemplo, dará como resultado:

```
TABLA DEL  9

 9 X 1 = 9
 9 X 2 = 18
 9 X 3 = 27
 9 X 4 = 36
 9 X 5 = 45
 9 X 6 = 54
 9 X 7 = 63
 9 X 8 = 72
 9 X 9 = 81
Ready
```

La misión de la línea 30 es la de no admitir números fuera del intervalo 1–9. Más adelante veremos como desechar, si no nos interesan, los valores no enteros.

EJEMPLO 4

Si se deja caer libremente un cuerpo, la relación entre el espacio recorrido E y el tiempo transcurrido T es:

```
E=4.9*T↑2
```

Con el siguiente programa se obtiene una tabla espacio–tiempo, en décimas de segundo hasta un tope dado.

```
1Ø INPUT "TIEMPO LIMITE ";TIEMPO
2Ø MODE 1
3Ø FOR T=Ø TO TIEMPO STEP Ø.1
4Ø PRINT T,4.9*T*T
50 NEXT
```

BUCLES ENCAJADOS

Dentro del proceso a repetir dentro de un bucle, pueden aparecer a su vez otro u otros bucles, obteniéndose lo que se conoce como bucles encajados o anidados. Por ejemplo, con el programa:

```
1Ø MODE 1
2Ø FOR X=1 TO 6
3Ø FOR Y=1 TO 6
4Ø PRINT X;"-";Y
5Ø NEXT Y
6Ø NEXT X
```

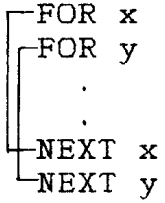
se imprimen todas las parejas de números enteros comprendidos entre 1 y 6.

Para que los bucles encajados funcionen correctamente, el bucle externo debe contener completamente al bucle o bucles internos. Es decir, son correctas estructuras de este tipo:

```
┌─FOR x
│ ┌─FOR y
│ │   .
│ │   .
│ │   .
│ └─NEXT y
└─NEXT x
```

```
┌──FOR x
│  ┌─FOR y
│  │   .
│  │   .
│  │ ┌─FOR z
│  │ │ ┌─FOR n
│  │ │ │   .
│  │ │ │   .
│  │ │ └─NEXT n
│  │ └─NEXT z
│  └─NEXT y
└──NEXT x
```

y, en cambio, son incorrectas, por solaparse, estructuras de la forma:



Por lo dicho anteriormente, el programa siguiente no funcionará correctamente.

```
10 FOR x=1 TO 8
20 FOR y=1 TO 3
30 PRINT x;y,
40 NEXT x
50 NEXT y
```

si intercambiamos entre sí las líneas 40 y 50, el programa funcionará perfectamente.

Aprovechando las ventajas que nos ofrece nuestro ordenador AMSTRAD, podríamos sustituir las líneas 40 y 50 por cualquiera de las siguientes:

```
40 NEXT:NEXT
40 NEXT y:NEXT x
40 NEXT y,x
```

NOTAS

1) Las variables de control de los bucles encajados deben de ser distintas.
2) En caso de tener muchos bucles encajados es aconsejable, por claridad, el especificar las variables en NEXT.

WHILE-WEND

Hasta ahora, con los bucles FOR–NEXT, se podía repetir un proceso un número determinado de veces, ya que era necesario conocer el valor límite. Sin embargo, en muchos problemas se desconoce el número de veces que hay que repetir un proceso; por ejemplo, al buscar un número que cumpla unas determinadas condiciones, al calcular la nota media de un número indeterminado de alumnos, etc... En estos casos, al trabajar con bucles

FOR–NEXT, hay que introducir una condición y un salto para poder salir del bucle.

Pero, para conseguir una buena programación estructurada, hay que restringir al máximo el uso de la sentencia GOTO ya que complica muchísimo el seguimiento y comprensión de los programas, sobre todo si son extensos.

Nuestro ordenador posee una nueva estructura de bucle: WHILE–WEND (mientras que – seguir) que, aunque no es BASIC estandar, facilita enormemente el programar de una forma estructurada.

La forma general de estos bucles es:

```
WHILE condición

 proceso a repetir

WEND
```

mientras la condición sea cierta, se repite el proceso y, cuando sea falsa, continúa en la línea siguiente a WEND.

Por ejemplo, con este nuevo tipo de bucles, el programa que imprime los pares menores que 100 será:

```
10 WHILE A<100
20 A=A+2
30 PRINT A
40 WEND
```

Como inicialmente el valor de A es cero, en la línea 20 se van obteniendo los diversos números pares, aumentándose en 2 el valor de A.

Se recomienda realizar todos los ejemplos vistos anteriormente con FOR–NEXT, con los nuevos bucles WHILE–WEND. Así, la suma de todos los enteros entre A y B se conseguirá con:

```
10 WHILE A>=B
20 INPUT A,B
30 WEND
40 CLS
50 WHILE A<=B
60 S=S+A
70 A=A+1
80 WEND
90 PRINT "la suma es";S
```

En el primer bucle (líneas 10–30) ponemos la condición de que A no sea superior a B, mientras esto no ocurra, el ordenador pedirá dos números. En el segundo bucle (líneas 50–80) la variable S almacena las diversas sumas parciales.

EJEMPLO 5

El más famoso y antiguo algoritmo conocido con este nombre, es el algoritmo de Euclides, que permite calcular el máximo común divisor de dos números enteros a y b. El método es el siguiente:

1) Sean a y b el mayor y el menor de los dos números.
2) Sea r el resto de dividir a entre b.
3) Si r no es cero, se hace a=b, b=r y se vuelve a 2)
4) Si r es cero, el máximo común divisor es r. Fin.

El programa que desarrolla este algoritmo es:

```
10 INPUT "escribe el número mayor";a
20 INPUT "escribe el número menor";b
30 IF a<b THEN 10
40 r=a MOD b
50 WHILE r<>0
60 a=b
70 b=r
80 r=a MOD b
90 WEND
100 PRINT "el mcd es";b
```

Observar que en la línea 40 hemos obtenido el resto de la división utilizando la instrucción MOD.

NOTAS

1) WEND es un juego de palabras. Por un lado es una contracción de WHILE y END, por lo que podría traducirse como el final del WHILE; por otra lado, su traducción literal es "seguir", es decir, que se siga repitiendo el bucle.

2) Si la condición inicial es falsa, no se ejecuta el bucle. Comprobarlo con:

```
10 a=5
20 WHILE a<0
30 PRINT "negativo"
40 WEND
```

3) La condición que acompaña a WHILE puede constar, a su vez, de varias condiciones unidas por los operadores lógicos AND, OR, XOR y NOT. Por ejemplo las siguientes líneas permiten que, al introducir calificaciones, no se admitan las que están fuera del intervalo 0–10.

```
10 n=-1
20 WHILE n<0 OR n>10
30 INPUT "calificación";n
40 WEND
```

¿Cuál es el cometido de la línea 10?

4) También se puede efectuar pausas con estos bucles; así, con:

```
10 WHILE a<1000
20 a=a+1
30 WEND
```

se consigue una pausa de unos 3 segundos aproximadamente.

5) Las órdenes WHILE, WEND tienen que ir unidas. Así, aparece el mensaje de error:

```
WEND missing
```

cuando a un WHILE no le corresponde ningún WEND, y el mensaje será:

```
Unexpected WEND
```

cuando encuentre un WEND sin el correspondiente WHILE.

BUCLES WHILE–WEND ENCAJADOS

De igual manera que sucedía con los bucles FOR–NEXT, también podemos introducir un bucle WHILE–WEND dentro de otro, con la ventaja adicional de que ahora no es posible solaparlos.

Por ejemplo, vamos a obtener todas las parejas de números en las que el primero varía de 1 a 8 y el segundo de 1 a 3.

```
10 a=1:b=1
20 WHILE a<9
30 WHILE b<4
40 PRINT a;b
50 b=b+1
60 WEND
70 b=1:a=a+1
80 WEND
```

¿Qué sucederá si quitamos la orden b=1 en la línea 70?. ¿Por qué?.

PROGRAMAS PROPUESTOS

1) Dibujar, con asteriscos y en los tres modos, un rectángulo lo mayor posible.
2) Realizar, con bucles, el dibujo del ejercicio 5–4
3) Se llama FACTORIAL de un número entero positivo n (y se representa por n!) al producto de los n primeros números enteros

   ```
   n!=1.2.3.4.5.....n
   ```

   Encontrar un programa que calcule el factorial de un número dado empleando bucles FOR–NEXT y WHILE–WEND.
4) Imprimir en la pantalla una tabla con las raíces cuadradas y cúbicas de los números enteros y positivos pertenecientes a un intervalo dado.
5) Sacar por la pantalla la numeración de las fichas de dominó.
6) En una carrera participan 6 corredores, numerados del 1 al 6. Se pide un programa que saque en pantalla todas las posibles llegadas, considerando solo los tres primeros.
7) Construir un programa, similar al de la tabla de multiplicar, pero de forma que, con cada número entre 1 y 9 nos interrogue sobre el producto y nos diga si hemos acertado o no.
8) Se llaman números pitagóricos a aquellos números enteros positivos x,y,z que cumplen:

   ```
   z*z=x*x+y*y
   ```

   Son números pitagóricos, por ejemplo: 3–4–5, 5–12–13, 7–24–25....., etc.

   Hacer un programa que encuentre todos los tripletes pitagóricos inferiores a un número dado.

8

Funciones

FUNCIONES NUMERICAS

Nuestro ordenador, al igual que muchas calculadoras, lleva incorporadas una serie de funciones numéricas estandar para facilitar la programación, especialmente en cuestiones de índole científica. Estas funciones, que actúan sobre un número o variable numérica devolviendo un resultado numérico, son:

ABS(X)

Calcula el valor absoluto de X; es decir, su valor es:

```
- X   si X negativo.
  X   si X es negativo o cero.
```

Así:

```
ABS(-3.4)=3.4     ABS(4.1)=4.1     ABS(Ø)=Ø
```

SGN(X)

Obtiene el signo de X, su valor será:

```
-1    si X es negativo.
 1    si X es positivo.
 Ø    si X es cero.
```

Así:

```
SGN(-3.4)=-1      SGN(4.1)=1      SGN(Ø)=Ø
```

SQR(X)

Calcula la raiz cuadrada (SQuare Root) de X; por tanto, X no podrá ser negativo. Así, por ejemplo, con:

```
PRINT SQR(29)
```

se imprime 5.38516481, y con

```
PRINT SQR(-29)
```

se imprime el mensaje de error:

```
Improper argument
```

EXP(X)

Es la función exponencial. Su valor es e^x, donde e = 2.7182818... Comprobarlo con:

```
1Ø MODE 1
2Ø PRINT "NUMERO","FUNCION"
3Ø FOR X=1 TO 1ØØ
4Ø PRINT X,EXP(X)
5Ø NEXT
```

Se observará que se obtiene el mensaje de error:

```
Overflow (rebosar)
```

cuando X es superior a 88.0296918.

LOG(X)

Es la función logaritmo neperiano, inversa de la anterior, y calcula el nú-

mero al cual hay que elevar e para obtener X; por consiguiente X deberá ser mayor que cero, presentándose el mensaje de error:

```
Overflow
```

en caso contrario.

LOG10(X)

Calcula el logaritmo decimal de X (número mayor que cero). El logaritmo decimal de X es el número al cual hay que elevar el 10 para obtener X. Comprobar el efecto de estas dos funciones logarítmicas con el programa siguiente.

```
10 MODE 1
20 PRINT "NUMERO","LOG(X)","LOG10(X)"
30 FOR X=0.5 TO 10 STEP 0.5
40 PRINT X,LOG(X),LOG10(X)
50 NEXT
```

MAX(X,Y,Z,.....)

Obtiene el valor máximo de los valores X,Y,Z,..... Comprobarlo con:

```
10 INPUT "INTRODUCIR 4 NUMEROS"; A,B,C,D
20 PRINT "EL MAYOR DE ELLOS ES "; MAX(A,
B,C,D)
```

MIN(X,Y,Z,.....)

Devuelve el mínimo número incluido en la lista. Así, por ejemplo,

```
PRINT MIN(2,-12,17)
```

imprimirá: −12.

FUNCIONES TRIGONOMETRICAS

Entre las funciones numéricas destacan por su importancia sobre todo en el terreno geométrico, las funciones trigonométricas.

SIN(X)

Es la función seno (SINe). Obtiene el valor del seno del ángulo X, expresado en radianes si no se indica lo contrario.

COS(X)

Es la función coseno. Da el coseno del ángulo X, expresado en radianes mientras no se indique lo contrario.

TAN(X)

Es la función tangente. Calcula el valor de la tangente del ángulo X, que se supone en radianes si no se indica lo contrario.

$$TAN(X) = \frac{SIN(X)}{COS(X)}$$

por lo que se imprimirá el mensaje:

```
Division by zero
```

cuando COS(X)=0

ATN(X)

Es la función ArcoTaNgente, inversa de la anterior. Calcula el ángulo, en radianes mientras no se indique lo contrario, cuya tangente es X.

PI

Es el valor del número PI, que es la relación constante existente entre la longitud de la circunferencia y su diámetro. Así, con

```
PRINT PI
```

se obtiene 3.14159265.

Evidentemente PI tiene será una palabra reservada.

Como se ha indicado, inicialmente nuestro ordenador maneja los ángulos en radianes. Sin embargo, se puede conseguir que lo haga en grados sexagesimales empleando la instrucción:

DEG

Que hace que el ordenador trabaje en grados (DEGrees) sexagesimales en lugar de en radianes.

Si queremos volver a manejar los ángulos en radianes, existe otra instrucción: **RAD** de efectos contrarios a DEG.

EJEMPLO 1

Se consigue en pantalla una tabla de la función seno desde 0 hasta 360 grados de intervalos de 15 grados, con el programa:

```
10 MODE 1
20 DEG
30 FOR X=0 TO 360 STEP 15
40 PRINT X,SIN (X)
50 NEXT
60 GOTO 60
```

que al ejecutarlo obtendremos:

```
0              0
15             0.258819045
30             0.5
45             0.707106781
60             0.866025404
75             0.965925827
90             1
105            0.965925826
120            0.866025404
135            0.707106781
150            0.5
165            0.258819045
180            0
195           -0.258819046
210           -0.5
225           -0.707106781
240           -0.866025404
255           -0.965925826
270           -1
285           -0.965925826
300           -0.866025404
315           -0.707106781
330           -0.5
345           -0.258819046
360            0
```

como la tabla ocupa toda la pantalla, hemos colocado la línea 60 para que el ordenador no muestre el mensaje "READY", y así lograr que los primeros ángulos no desaparezcan.

Trabajando en radianes el programa sería así:

```
10 MODE 1
20 RAD
30 FOR x=0 TO 2*PI STEP PI/12
40 PRINT x*180/PI,SIN(x)
50 NEXT
60 GOTO 60
```

NOTAS

1) Un radian es, en una circunferencia, el ángulo central que abarca un arco cuya longitud es igual al radio. Por lo tanto 2*PI radianes serán 360 grados y un radian es aproximadamente:

```
57 grados 17 minutos y 44 segundos
```

2) La función ATN (X), devuelve un ángulo comprendido entre 90 y −90 grados, si se trabaja en grados y entre −PI/2 y PI/2, cuando se trabaja en radianes.

3) Si se introduce la orden DEG como comando directo (sin número de línea), su efecto desaparece cuando se ejecuta cualquiera de las órdenes siguientes: CLEAR, LOAD, NEW , RUN además de, evidentemente, RAD.

4) Si G es un ángulo expresado en grados, su equivalente en radianes es:

$$\frac{PI*G}{180}$$

y a la inversa, R radianes son:

$$\frac{180*R}{PI}$$

grados sexagesimales:

FUNCIONES ENTERAS

Vamos a estudiar cuatro funciones numéricas que presentan la particula-

ridad de que su resultado es un número entero (en realidad con la instrucción ROUND pueden también obtenerse resultados decimales).

INT(X)

Es la función parte entera (INTeger). Todo número X está comprendido entre dos enteros:

```
A<=X<=B
```

INT(X) es el menor de estos dos enteros. Así:

```
INT(2.67)=2      INT(-2.67)=-3
```

ya que:

```
2<=2.67<=3   y   -3<=-2.67<=-2
```

Con el consiguiente programa se podrá investigar con los números deseados:

```
10 INPUT X
20 PRINT INT(X)
30 GOTO 10
```

FIX(X)

Obtiene el entero resultante de suprimir la parte decimal de X (FIX = ajustar). Así, por ejemplo:

```
FIX(2.67)=2   y   FIX(-2.67)=-2
```

NOTAS

1) Observar que si el número X es entero, se tiene:

```
X=INT(X)=FIX(X)
```

2) De acuerdo con la definición, si X es entero o positivo, se tiene que

```
INT(X)=FIX(X)
```

en caso contrario, es decir si X es decimal negativo, se cumple que:

```
FIX(X)=INT(X)+1
```

CINT(X)

Devuelve el entero más cercano a X, siempre que esté comprendido entre −32768 y 32767. Por ejemplo:

```
CINT(12.45)=12    CINT(-12.45)=-12    CINT(1.8)=2
      CINT(32767.4)=32767      CINT(2.5)=3
```

Y se emite el mensaje:

```
Overflow
```

cuando el resultado sobrepasa los límites indicados. Comprobarlo con:

```
PRINT CINT(-32768)    PRINT CINT(32767.5)
```

ROUND(X)

Es la función redondear (ROUND). Obtiene el entero más cercano a X pero, a diferencia de CINT, no tiene la limitación anterior. Así, por ejemplo:

```
ROUND(12.45) =12        ROUND(-Ø.8)=-1
          ROUND(39483.7)=39484
```

Esta instrucción admite una generalización muy interesante:

```
ROUND(X,N)
```

que redondea el número X a N decimales. Así:

```
ROUND(123.455,2) = 123.46
ROUND(34.5692,3)=34.569
```

Si N=0, estamos en el primer caso. Es decir,

```
ROUND(X,Ø)=ROUND(X)
```

Si N es negativo, se redondea a la potencia de 10, con exponente −N, más cercana. Así,

```
ROUND(1247.4621,-2)=12ØØ

 ROUND(12.474621,-1)=1Ø

 ROUND(16Ø.62,-2)=2ØØ
```

A modo de ejercicio recapitulador, completar sin el ordenador el siguiente cuadro:

VALOR DE X	INT(X)	FIX(X)	CINT(X)	ROUND(X)	ROUND(X,2)
1.876	1	1	2	2	1.88
−1.876					
−1.2791					
12.5					
548.3794					
−548.3794					
32767.613					
32767.413					
−32768.4					
−32768.7					
−12.5					

EJEMPLO 2

Recordemos que un número se dice primo cuando sólo es divisible por la unidad y por sí mismo. Veamos un programa, con la función INT que averigua si un número entero y positivo es primo o no.

```
1Ø INPUT "DAME UN NUMERO";N
2Ø IF N<=Ø OR N<>INT(N) THEN 1Ø
3Ø IF N=1 OR N=2 THEN PRINT N;"ES PRIMO":
END
4Ø FOR X=2 TO SQR(N)
5Ø IF N/X=INT(N/X) THEN PRINT N;"NO ES
PRIMO":END
6Ø NEXT
7Ø PRINT N;"ES PRIMO"
```

observar que el valor límite de la variable de control del bucle es SQR(N) en lugar de N−1, ya que si un número no es divisible por ninguno inferior a su raiz cuadrada, tampoco será divisible por ninguno mayor.

FUNCIONES DEFINIDAS POR EL USUARIO

En BASIC se permite que el programador defina sus propias funciones. Esto se suele hacer cuando una misma expresión aparece varias veces a lo largo de un programa, consiguiéndose así ahorrar memoria y evitar errores al tener que teclearla sólo una vez.

La orden para definir nuestras propias funciones es:

DEF FN

Cuya expresión general es:

```
DEF FN nombre (variable/s) = expresión
```

Posteriormente, cuando necesitamos obtener el valor de la función para un valor concreto de las variable, introduciremos:

```
FN nombre (valor de la/s variable/s)
```

Por ejemplo, con:

```
1Ø DEF FN A(X)=X^2+Y^3
```

hemos definido una función, de nombre A, dependiente de una sola variable y que calcula la suma de su cuadrado y de su cubo. Si ahora deseamos saber que valor tiene esta función para un valor determinado k, podremos hacerlo con:

```
2Ø INPUT K
3Ø PRINT FN A(K)
```

EJEMPLO 3

En la definición de una función pueden aparecer otras funciones. Así, en el programa:

```
1Ø DEF FN B(X,Y)=3*SIN(X)+4*COS(X)
2Ø INPUT X,Y
3Ø PRINT FN B(X,Y)
```

hemos definido la función B, que depende de dos variables, partiendo de las funciones seno y coseno.

EJEMPLO 4

A veces, en el cálculo científico, se trabaja con logaritmos cuya base no es ni 10 ni el número e; en este caso es conveniente disponer de una función que nos calcule el logaritmo en una base A cualquiera. Para ello recordemos la expresión:

$$\log_A(X) = \frac{\log(X)}{\log(A)}$$

con lo que el programa de cambio de base será:

```
1Ø INPUT "NUEVA BASE";A
2Ø DEF FN LG(X)=LOG(X)/LOG(A)
3Ø INPUT "NUMERO";N
4Ø PRINT "EL LOGARITMO DE"N"EN BASE"A"ES"
FN LG(N)
5Ø PRINT
6Ø GOTO 1Ø
```

NOTAS

1) En nuestro ordenador, a diferencia de otros, el nombre de la función no tiene porque constar de una sola letra. En general se admite la misma denominación que en el caso de las variables.

2) Si se pretende obtener el valor de una función sin haberla definido anteriormente, se presenta el mensaje de error:

```
Unknown user function in n
```

donde n es el número de línea donde se ha efectuado el intento.

PROGRAMAS PROPUESTOS

1) Obtener en pantalla una tabla de las funciones seno, coseno y tangente desde 0 grados hasta 90 grados a intervalos de 5 y con cuatro decimales.

2) Dado un número, presentar en pantalla el valor de todas sus funciones numéricas siempre que sea posible calcularlas (por ejemplo, si el número es negativo, no tendrá logaritmo)

3) Encontrar todos los números pares mayores que dos y menores que 50000 tales que al dividirlos por 3,5,7,9, y 11 den siempre de resto 2.

4) Conseguir que el ordenador imprima todos los números primos menores que un número entero dado mayor que 2.

5) Descomponer cualquier número entero positivo en sus factores primos.

6) Se dice que un número entero positivo es perfecto si es igual a la suma de todos sus divisores excepto él mismo. Elaborar un programa que compruebe si un número es perfecto o no. Probar con los números 8128 y 130816.

7) Dos números enteros positivos son amigos cuando cada uno de ellos es igual a la suma de todos los divisores del otro. Construir un programa que nos diga si dos números dados son amigos o no. Probar con las parejas: 1184, 1210 y 17296, 18416.

9

Cadenas

Una cadena es un conjunto de caracteres de cualquiera de los tres tipos: alfabéticos, numéricos o especiales. Recordemos que el nombre de una cadena finaliza con el símbolo $ y sus caracteres se deben incluir entrecomillados. Por ejemplo:

```
A$ = "AMSTRAD"
```

Existe una función predefinida entre cadenas: la concatenación representada con el símbolo +. Esta función obtiene la cadena que resulta de unir todos los caracteres de las cadenas dadas. Así, por ejemplo, con

```
10 MODE 1
20 A$="PROGRAMANDO CON"
30 B$=" AMSTRAD"
40 PRINT A$+B$
```

se obtiene:

Recordar también que la longitud máxima de una cadena es de 255 caracteres; si se intenta trabajar con una longitud mayor aparece el mensaje de error:

```
String too long in (número de línea)
```

esto se puede comprobar ejecutando el siguiente programa:

```
10 FOR X=1 TO 256
20 A$=A$+"A"
30 NEXT
```

que presenta el mensaje de error:

```
String too long in 20
```

Para calcular la longitud que tiene la cadena A$, existe en BASIC la función:

LEN

Que adopta la forma general:

```
LEN(A$)
```

Así, por ejemplo, si A$="EVEREST", entonces

```
PRINT LEN(A$)
```

presentará en la pantalla el número 7. Lo mismo hubiera sucedido si hubieramos introducido:

```
PRINT LEN("EVEREST")
```

Por supuesto, al calcular la longitud de una cadena también se contabilizan los espacios en blanco. Así,

```
PRINT LEN("AMSTRAD          ")
```

dará como resultado: 13.

EJEMPLO 1

En ocasiones nos encontramos ante la necesidad de centrar en la pantalla una cabecera o un mensaje en general. Los cálculos que hay que efectuar no son difíciles pero aquí exponemos un programa que realiza el trabajo por nosotros.

```
10 INPUT "MODE";M
20 INPUT "MENSAJE ";M$
30 IF LEN(M$)>2^M*20 THEN 10
40 MODE M
50 C=2^M*10+1-LEN(M$)/2
60 PRINT TAB(C);M$
```

nos pide el modo de trabajo M y el mensaje M$ y luego lo presenta centrado en la pantalla. Por la línea 30 no admite mensajes más largos que una línea ya que 2↑M*20 toma los valores 20, 40 y 80 según M valga 0, 1 ó 2.

TROCEADO DE CADENAS

Existen en BASIC tres funciones predefinidas que nos permiten obtener subcadenas de una dada. Son las siguientes:

LEFT$ (LEFT: izquierda)

Su forma general es:

```
LEFT$(A$,N)
```

donde A$ es una variable alfanumérica y N es un número entero.

Con esta función se obtiene como resultado la cadena formada por los N primeros caracteres de A$ empezando a contar por la izquierda.

Si A$="UNIVERSO", entonces con:

```
LEFT$(A$,1) es la cadena "U"
LEFT$(A$,2) es la cadena "UN"
LEFT$(A$,3) es la cadena "UNI"
```

Si le pedimos más caracteres que los que tiene A$, no nos dará error sino que suministrará hasta el último caracter. Así

```
LEFT$(A$,1Ø) = "UNIVERSO"
```

Con N=0, dá como resultado la cadena vacía.

EJEMPLO 2

Dada una cadena A$, escribir sus n primeros caracteres con n variando desde 1 hasta LEN(A$).

```
1Ø INPUT A$
2Ø FOR X=1 TO LEN(A$)
3Ø PRINT LEFT$(A$,X)
4Ø NEXT
```

RIGHT$ (RIGHT: derecha)

Su forma general es:

```
RIGHT$(A$,N)
```

donde A$ es una variable alfanumérica y N es un número entero. El resultado es una cadena que contiene los N primeros caracteres de A$ empezando a contar por la derecha.

Si A$="UNIVERSO", entonces:

```
RIGHT$(A$,1) es la cadena "O"
RIGHT$(A$,2) es la cadena "SO"
RIGHT$(A$,3) es la cadena "RSO"
```

y, como sucedía en la función anterior, se cumple que:

```
RIGHT$(A$,10) = "UNIVERSO" y   RIGHT$(A$,0) = ""
```

EJEMPLO 3

Este programa es análogo al anterior con la diferencia de que ahora las subcadenas se toman por la derecha.

```
10 INPUT A$
20 FOR X=1 TO LEN(A$)
30 PRINT RIGHT$(A$,X)
40 NEXT
```

MID$ (MIDdle: en medio)

Su forma general es:

```
MID$(A$,A,B)
```

donde A$ es una cadena y A y B dos números enteros.

El resultado es una cadena formada por los B caracteres de A$ que hay a partir del que ocupa el lugar A. Así, si A$="UNIVERSO" se tiene que:

```
MID$(A$,1,3) es la cadena "UNI"
MID$(A$,3,4) es la cadena "IVER"
MID$(A$,5,3) es la cadena "ERS"
```

Si B es mayor que el número de caracteres disponibles o no se especifica su valor, se obtiene la cadena formada desde la posición A hasta el final. Por ejemplo:

```
MID$(A$,4,10) es la cadena "VERSO"
MID$(A$,4) es la cadena "VERSO"
```

Si B=0 ó A>LEN(A$) se obtiene la cadena vacía. Por ejemplo:

```
MID$(A$,4,0) es la cadena vacia
MID$(A$,10,1) es la cadena vacia
```

NOTA

De acuerdo con las definiciones anteriores, es evidente que se cumplen las siguientes igualdades:

```
LEFT$(A$,X)=MID$(A$,1,X)
RIGHT$(A$,Y)=MID$(A$,LEN(A$)-Y+1)
```

NUMEROS Y CADENAS

Existen dos funciones predefinidas que nos permiten trabajar los números como cadenas o viceversa. Son las siguientes:

STR$ (STRing: cadena)

Obtiene una cadena formada por los caracteres de X, que debe ser un número o una variable numérica, incluyendo el espacio reservado para el signo. Es decir:

```
STR$(12)=" 12"
STR$(-123)="-123"
```

esta instrucción se empleará en todos aquellos casos en que sea más fácil trabajar con cadenas que con números. Así, por ejemplo, si queremos obtener las cifras de las decenas de un número, procederíamos así:

```
10 INPUT N
20 A$=STR$(N)
30 PRINT MID$(A$,LEN(A$)-1,1)
```

empleando la nueva orden. Si se trabajase directamente con números el programa sería:

```
10 INPUT N
20 X=N MOD 100
30 Y=X\10
40 PRINT Y
```

EJEMPLO 4

El siguiente programa averigua si un número entero positivo es capicúa o no.

```
10 INPUT N
20 IF N<0 OR N<>INT(N) THEN 10
30 A$=STR$(N)
40 A$=MID$(A$,2)
50 IF LEN(A$)>9 THEN 10
60 FOR X=1 TO LEN(A$)
70 B$=MID$(A$,X,1)+B$
80 NEXT
90 IF A$=B$ THEN PRINT N;"ES  CAPICUA":END
100 PRINT N;"NO ES CAPICUA"
```

en las líneas 20 y 50 se comprueba que el número es realmente un entero positivo. Se considera, línea 40, la cadena A$ formada por todas las cifras del número y se obtiene, bucle 60–80, la cadena inversa B$. Por último, en las líneas 90 y 100, se analiza si el número dado es capicúa o no.

VAL (VALue: valor)

Su forma general es:

```
VAL(A$)
```

y su efecto es el contrario de STR$: Si A$ es una cadena formada por dígitos, incluso precedidos de signo y/o con punto decimal, el resultado será el número representado por dicha cadena.

```
Si A$="-42.5", entonces  VAL(A$)=-42.5.
```

Con el consiguiente programa:

```
10 INPUT A$
20 PRINT "EL VALOR DE ";A$;" ES";VAL(A$)
30 GOTO 10
```

comprobar que:

1) Si el primer carácter de A$ es alfabético, el resultado es 0.
2) Si el primer carácter es A$ es especial, salvo algunas excepciones, el resultado es 0.
3) Si los primeros caracteres son dígitos seguidos de caracteres alfanuméricos, el resultado es el número formado por los primeros.

4) Si el primer carácter es el signo "+" o el signo "–" y el siguiente es un carácter alfabético, dará el error:

```
Type mismatch in n
```

siendo n el número de línea, 20 en nuestro caso.

NOTA

Observar el diferente significado que tiene la operación "+" entre cadenas y entre números.

```
10 A$="12"
20 B$="18"
30 PRINT A$+B$
40 PRINT VAL(A$)+VAL(B$)
```

EJEMPLO 5

Dado un número entero positivo, vamos a calcular la suma de todas sus cifras.

```
10 INPUT N
20 IF N<=0 OR N<>INT(N) THEN 10
30 A$=MID$(STR$(N),2)
40 FOR X=1 TO LEN(A$)
50 B$=MID$(A$,X,1)
60 S=S+VAL(B$)
70 NEXT
80 PRINT S
```

Aquí es preferible trabajar con cadenas por la facilidad que tenemos de trocearlas.

CODIGOS ASCII

Cada caracter de nuestro ordenador, alfabético, numérico o especial, tiene asociado un código que le identifica. Este código es un número entero comprendido entre 0 y 255.

La mayoría de los ordenadores coinciden en asignar, al menos en el intervalor 32–126, los mismos códigos a los mismos caracteres. Estos códigos son los llamados códigos ASCII (American Standard Code for Information

Interchange). En la siguiente tabla se muestran todos los caracteres asignados por AMSTRAD, en el modelo CPC 6128 y teclado en español, a los códigos comprendidos entre el 32 y 255.

32		33	!	34	"	35	#	36	$	37	%	38	&	39	'	40	(	41	)	42	*
43	+	44	,	45	-	46	.	47	/	48	0	49	1	50	2	51	3	52	4	53	5
54	6	55	7	56	8	57	9	58	:	59	;	60	<	61	=	62	>	63	?	64	@
65	A	66	B	67	C	68	D	69	E	70	F	71	G	72	H	73	I	74	J	75	K
76	L	77	M	78	N	79	O	80	P	81	Q	82	R	83	S	84	T	85	U	86	V
87	W	88	X	89	Y	90	Z	91	[	92	\	93	]	94	↑	95	_	96	`	97	a
98	b	99	c	100	d	101	e	102	f	103	g	104	h	105	i	106	j	107	k	108	l
109	m	110	n	111	o	112	p	113	q	114	r	115	s	116	t	117	u	118	v	119	w
120	x	121	y	122	z	123	{	124	¡	125	}	126	~	127	▒	128		129	▘	130	▝
131	▀	132	▖	133	▌	134	▗	135	▐	136	▂	137	▚	138	▜	139	▞	140	▟	141	▙
142	▄	143	▆	144	·	145	¹	146	ª	147	⅄	148	¡	149	¡	150	¿	151	⊦	152	▪
153	⁄	154	▬	155	▲	156	¬	157	¶	158	⊤	159	⊥	160	^	161	ñ	162	"	163	Ñ
164	@	165	¶	166	∮	167	'	168	⅂	169	½	170	¾	171	ñ	172	÷	173	¬	174	ó
175	¡	176	∝	177	β	178	∀	179	δ	180	ε	181	θ	182	λ	183	μ	184	π	185	σ
186	∮	187	∮	188	χ	189	ω	190	Σ	191	Ω	192	/	193	\	194	/	195	\	196	^
197	)	198	∨	199	(	200	∕	201	\	202	◊	203	X	204	/	205	\	206	▒	207	▒
208	▔	209	|	210	_	211	|	212	▛	213	▜	214	▟	215	▙	216	▒	217	▒	218	▒
219	▒	220	▒	221	▒	222	▒	223	▒	224	♚	225	♛	226	♜	227	♦	228	♥	229	♠
230	o	231	♠	232	□	233	▮	234	♂	235	♀	236	♩	237	♪	238	♫	239	♣	240	♯
241	♭	242	♯	243	♯	244	▲	245	▼	246	▶	247	◀	248	▒	249	▒	250	▒	251	▒
252	▒	253	▒	254	▒	255	▒														

Los códigos comprendidos entre 0 y 31, que no aparecen en la tabla anterior, corresponden a caracteres de control y los estudiaremos más adelante.

Para poder trabajar cómodamente con los códigos ASCII, existen las funciones:

ASC

Su expresión general es:

```
ASC(A$)
```

y devuelve el código ASCII del primer caracter de la cadena.

Así:

```
ASC(" ")=32
ASC("A")=65
ASC("Aa")=65
```

CHR$ (Character: caracter)

Su forma general es:

```
CHR$(A)
```

donde A es un número entre 0 y 255.

Esta función devuelve el caracter cuyo código es A.

Con el siguiente programa, obtendremos todos los caracteres cuyos códigos estén entre 32 y 255 (tabla anterior).

```
10 MODE 2
20 ZONE 8
30 FOR X=32 TO 255
40 PRINT X;CHR$(X),
50 NEXT
```

EJEMPLO 6

Haciendo uso de algunos de los conocimientos adquiridos en este capítulo, se puede conseguir que el ordenador cifre mensajes.

```
10 INPUT "MENSAJE";M$
20 FOR X=1 TO LEN(M$)
30 C=ASC(MID$(M$,X,1))
40 C=C+2
50 A$=A$+CHR$(C)
60 NEXT
70 PRINT "EL MENSAJE CIFRADO ES "
80 PRINT A$
```

en la línea 40 se va cambiando los códigos ASCII de cada uno de los caracteres, con lo que el mensaje quedará irreconocible.

Así, el mensaje:

```
EL NOMBRE DE LA ROSA
```

se cifra como

```
GN"PQODTGFG"NC"TQUC
```

El conseguir un programa que descifre lo anterior es cosa fácil, sólo tendríamos que cambiar la línea 40 por:

```
40 C=C-2
```

Más consideraciones sobre el código ASCII

Los códigos comprendidos entre 0 y 31 corresponden a caracteres de control y, en general, no se pueden imprimir directamente en pantalla. Sin embargo, con CHR$(1) si que es posible esta impresión:

```
10 FOR X=0 TO 31
20 PRINT X;CHR$(1);CHR$(X)
30 NEXT
```

Mencionaremos el efecto de alguno de estos caracteres de control:

- PRINT CHR$(7) hace que el ordenador emita un pitido.
- PRINT CHR$(8) retrasa el cursor un lugar. Así:

```
PRINT "A";CHR$(8);"B"
```

  presenta en pantalla una "B" ya que al retrasarse el cursor un lugar ésta letra borra la anterior.
- PRINT CHR$(9) avanza el cursor un lugar. Así, con:

```
PRINT "A";CHR$(9);"B"
```

  se imprime:"A B".
- PRINT CHR$(10) hace bajar el cursor una línea conservando la misma columna que tenía anteriormente.
- PRINT CHR$(24) imprime en inverso. Observaremos el efecto que tiene con:

```
PRINT CHR$(24);"MENU";CHR$(24)
```

  se debe colocar CHR$(24) delante y detrás de la expresión a imprimir para que el efecto no perdure.
- También es posible imprimir un caracter en el espacio que ocupa otro sin que el primero se borre. Veamos como:

```
10 PRINT CHR$(22)+CHR$(1)
20 LOCATE 16,13:PRINT "AMSTRAD"
30 LOCATE 16,13:PRINT "________"
40 PRINT CHR$(22)+CHR$(0)
```

  En la línea 10 se activa el efecto de sobreimpresión o transparencia y en la línea 40 se desactiva.

En las líneas 20 y 30 se imprime dos veces en el mismo lugar de la pantalla.

Los códigos ASCII también permiten emplear los operadores de relación con caracteres de cualquier tipo.

Así, podremos comparar el caracter "A" con el caracter "a" y diremos que A<a pues el código ASCII del primer caracter es menor que el del segundo.

De igual manera, diremos que el caracter "+" es mayor que el caracter "*" ya que el código ASCII del primero es 43 y el del segundo es 42.

El ordenador también puede comparar cadenas, y lo hace comparando caracter a caracter. Dos cadenas serán iguales si sus caracteres, uno a uno, son iguales y en caso contrario una de ellas será menor que la otra.

Con el siguiente programa podremos ejercitarnos con todo esto:

```
10 INPUT A$,B$
20 IF A$<B$ THEN PRINT A$"<"B$
30 IF A$>B$ THEN PRINT A$">"B$
40 IF A$=B$ THEN PRINT A$"="B$
50 GOTO 10
```

La aplicación inmediata de todo esto es la ordenación alfabética.

OTRAS FUNCIONES PARA TRABAJAR CON CADENAS

LOWER$(A$)

Obtiene la cadena que resulta de sustituir en A$ las letras mayúsculas por las correspondientes minúsculas (LOWERS), permaneciendo inalterables los caracteres de A$ no alfabéticos. Por ejemplo:

```
LOWER$("aAbB12$%")= "aabb12$%"
```

UPPER$(A$)

Es la función inversa de la anterior. Devuelve la cadena formada por los caracteres de A$ habiendose sustituido los caracteres alfabéticos en minúsculas por sus correspondientes en mayúsculas (UPPER), permaneciendo igual los demás.

SPACE$(N)

Obtiene la cadena formada por N espacios (space) en blanco, por lo que N deberá estar entre 0 y 255. Así, con:

```
A$=SPACE$(3):PRINT "A";A$;"B"
```

se tiene: A B

INSTR(N,A$,B$)

El resultado de esta función es un número que nos indica a partir de qué caracter de A$ está contenida B$; en el caso de que B$ no esté contenida en A$ nos dará el valor 0.

El número N, entre 1 y 255, indica a partir de qué caracter de A$ se tiene que empezar a buscar; si se omite:

```
INSTR(A$,B$)
```

empezará por el primero.

EJEMPLO 7

Sea A$="caracola" y B$="col", se tiene:

```
INSTR(1,A$,B$)=5
INSTR(6,A$,B$)=Ø
INSTR(A$,B$)=5
```

si B$="COL", el resultado de esta función será 0 siempre.

STRING$(N,"caracter")

Forma una cadena compuesta por N caracteres como el especificado en segundo lugar. El valor de N debe estar entre 0 y 255. Así:

```
STRING$(6,"$")="$$$$$$"
```

Existe otra manera de usar esta función:

```
STRING$(6,36)
```

que tendrá el mismo resultado que la anterior, ya que 36 es el código ASCII del caracter "$".

BIN$(X,Y)

Pasa el número entero X a binario, siendo el resultado una cadena de ceros y unos. El parámetro "Y" es el número de cifras binarias que como máximo, podrá ser de 16; por lo que el mayor número que es posible pasar a binario es el 65535, dando el error "Overflow" para valores mayores.

Con el programa siguiente, se le suministra al ordenador los parámetros X e Y obteniendo como respuesta el número binario.

```
10 INPUT X,Y
20 PRINT BIN$(X,Y)
30 GOTO 10
```

si X no se puede expresar con solo Y cifras binarias, se emplearán todas las necesarias, siendo el tope máximo el indicado anteriormente de 16. Así, si X=12 e Y=2, como el número 12 en binario no se puede expresar con dos cifras únicamente, se emplean las cuatro que son necesarias:1100.

Observar que también se admiten números negativos, siendo el tope para estos el −32768.

HEX$(X,Y)

Pasa el número entero X a hexadecimal empleando Y cifras hexadecimales, con un máximo de cuatro.

Al igual que en el caso anterior, el valor de X debe estar entre −32768 y 65535.

Si el número X no puede ser expresado en hexadecimal con sólo Y cifras, se emplearán todas las que sean necesarias. Comprobarlo con:

```
10 INPUT X,Y
20 PRINT HEX$(X,Y)
30 GOTO 10
```

DEF FN

También podremos definir nuestras propias funciones alfanuméricas de forma análoga a como lo haríamos en el capítulo anterior; por ejemplo,

la siguiente función depende de dos variables alfanuméricas, X$ y Y$, y nos devuelve la cadena que se obtiene concatenando (uniendo) las dos cadenas X$ y Y$.

```
10 DEF FN A$(X$,Y$)=X$+Y$
20 INPUT X$,Y$
30 PRINT FN A$(X$,Y$)
```

e incluso podemos definir funciones cuyas variables sean alfanuméricas pero el resultado que sea un número:

```
10 DEF FN A(X$,Y$,Z$)=LEN(X$)+LEN(Y$)+
LEN(Z$)
20 INPUT X$,Y$,Z$
30 PRINT FN A(X$,Y$,Z$)
```

en este caso la función es numérica ya que el resultado es un número, por lo que se le denomina con A y sin el signo $. Si transgredimos esta norma, el ordenador dará el mensaje:

```
Type mismatch
```

PROGRAMAS PROPUESTOS

1) Dado un verbo en castellano, obtener a qué conjugación pertenece.
2) Construir el presente de indicativo de cualquier verbo regular de la primera o segunda conjugación.
3) Dada una frase, contabilizar el número de veces en que aparece un caracter dado.
4) Dada una frase, sin signos de puntuación, decir cuantas veces aparece una palabra dada en ella empleando INSTR
5) Hacer que el ordenador escriba literalmente cualquier número entero positivo de 4 cifras (si tiene menos de 4 completar con ceros). Así, introduciendo el número 3256 tendrá que aparecer en pantalla: tres mil doscientos cincuenta y seis.

10

Azar

RND

En gran cantidad de programas, sobretodo en juegos y simulaciones, es necesario obtener números al azar. Nuestro ordenador tiene incorporada una lista fija de números decimales comprendidos entre 0 y 1. Para tomar valores de esta lista se utiliza la función azar (RaNDom).

```
RND (X)
```

que tiene tres modalidades distintas, dependiendo del signo de X no de su valor.

Si X es positivo RND(X) toma como valor el primer número de la lista. Cuando volvemos a utilizar esta orden obtenemos el siguiente término de la lista, y así sucesivamente.

Veamos con un ejemplo como funciona en esta modalidad; para ello desconectar el ordenador, volverlo a conectar e introducir el programa

```
10 CLS
20 FOR X=1 TO 20
30 PRINT RND(1)
40 NEXT
```

al ejecutarlo se obtiene en pantalla los veinte primeros números de la lista.

```
 0.271940658
 0.528612386
 0.021330127
 0.175138616
 0.657773343
 0.653729687
 0.601031218
 0.934601486
 0.545995176
 6.34897E-02
 0.989510193
 0.140522114
 3.67999E-02
 0.182864718
 0.352758596
 0.375612436
 0.342460861
 0.335665084
 0.525869688
 0.595680852
Ready
```

Si ahora se ejecuta de nuevo el programa, se obtendrán los siguientes 20 números de la lista, etc.

Se puede comprobar que la secuencia es fija desconectando el ordenador, volviéndolo a conectar e introduciendo de nuevo el programa anterior. Al ejecutarlo se tendrá de nuevo los veinte primeros números de la lista.

El valor de la variable X es indiferente; habríamos obtenido los mismo números si en lugar de poner en la línea 30 RND(1) hubieramos colocado, por ejemplo, RND(35). Por esta razón, nuestro ordenador nos permite no tener que indicar éste. Así, en lugar de:

```
RND(número positivo)
```

podemos introducir:

```
RND
```

Si X es negativo la función RND toma un valor fijo de la lista. Se puede comprobar con el siguiente programa:

```
10 FOR X=1 TO 20
20 PRINT RND(-1)
30 NEXT
```

Con este programa, se obtienen los números fijos de la lista correspondientes a los trece primeros enteros negativos.

```
10 FOR X=-1 TO -13 STEP -1
20 PRINT X,RND(X)
30 NEXT
```

Analizar el siguiente programa para acabar de comprender el funcionamiento de la orden RND con parámetro negativo. Comparar su resultado con los veinte primeros números de la lista fija obtenida al principio del capítulo.

```
10 PRINT RND(-1)
20 FOR X=1 TO 10:PRINT RND:NEXT
30 PRINT
40 PRINT RND(-13)
50 FOR X=1 TO 10:PRINT RND:NEXT
```

Si X es nulo la función RND toma como valor el anterior número obtenido al azar. Comprobarlo con:

```
10 CLS
20 PRINT "RND";TAB(18);"RND(0)":PRINT
30 FOR X=1 TO 10
40 PRINT RND;TAB(18);RND(0)
50 NEXT
```

SIMULACIONES Y JUEGOS

Vamos a utilizar la función RND para simular diversos experimentos. Así, por ejemplo, lanzar una moneda y obtener cara o cruz es un experimento análogo al de tomar un número y ver si es mayor o menor que 0.5.

El siguiente programa simula el lanzamiento de una moneda cien veces:

```
10 FOR X=1 TO 100
20 A=RND
30 IF A<0.5 THEN PRINT "CARA "; ELSE
PRINT "CRUZ ";
40 NEXT
```

En la línea 30 asociamos al suceso "obtener cara" al hecho de que el número sea menor que 0.5 y "obtener cruz" en caso contrario.

Si hubiesemos querido contabilizar el número de caras y cruces bastaría con haber añadido al programa las líneas:

```
25 IF A<Ø.5 THEN CA=CA+1
5Ø PRINT:PRINT "CARAS:";CA,"CRUCES:";1ØØ-CA
```

EJEMPLO 1

En la mayoría de los juegos (lotería, dados, ruleta, etc.) lo que se busca al azar es un número entero. Esto también se puede conseguir con **RND**. Si queremos obtener números enteros aleatorios comprendidos entre P y Q, ambos inclusive, hay que utilizar la expresión:

```
INT((Q-P+1)*RND)+P
```

Así, por ejemplo, el siguiente programa obtiene 20 números enteros aleatorios comprendidos entre dos números dados:

```
1Ø INPUT "NUMERO MENOR";P
2Ø INPUT "NUMERO MAYOR";Q
3Ø FOR X=1 TO 2Ø
4Ø PRINT INT((Q-P+1)*RND)+P
5Ø NEXT
```

EJEMPLO 2

Basándonos en lo anterior, vamos a simular el lanzamiento de un dado, es decir, se elige un número entero comprendido entre 1 y 6, por lo que la orden será:

```
INT(6*RND)+1
```

Con el siguiente programa lanzaremos seis mil veces un dado y se contabilizará el número de veces que aparece cada puntuación:

```
10 FOR X=1 TO 6000
20 A=INT(RND*6)+1
30 IF A=1 THEN UN=UN+1
40 IF A=2 THEN DO=DO+1
50 IF A=3 THEN TR=TR+1
60 IF A=4 THEN CU=CU+1
70 IF A=5 THEN CI=CI+1
```

```
80 IF A=6 THEN SE=SE+1
90 NEXT
100 PRINT:PRINT
110 PRINT "NUMERO","FRECUENCIA"
120 PRINT TAB(3);1,UN
130 PRINT TAB(3);2,DO
140 PRINT TAB(3);3,TR
150 PRINT TAB(3);4,CU
160 PRINT TAB(3);5,CI
170 PRINT TAB(3);6,SE
```

Se puede comprobar que, como era de prever, las frecuencias de las seis puntuaciones son aproximadamente mil.

EJEMPLO 3

Con la orden RND vamos a simular el sorteo de la lotería: se presentarán en pantalla los tres primeros premios de cinco cifras cada uno y las 16 terminaciones de tres cifras; por ejemplo:

```
PRIMER PREMIO          8  4  4  6  4

SEGUNDO PREMIO         7  0  2  0  3

TERCER PREMIO          4  0  6  6  0


TERMINACIONES

 9  4  5        2  7  8        3  3  2
 0  5  9        5  6  9        7  4  1
 7  9  6        9  1  4        6  7  8
 5  0  0        6  1  8        2  7  4
 1  5  9        8  6  5        1  8  2
 5  4  9

Ready
```

El programa será el siguiente:

```
10 MODE 1
20 PRINT "PRIMER PREMIO";TAB(20)
30 FOR X=1 TO 5:PRINT INT(RND*10);:NEXT
40 PRINT:PRINT
50 PRINT "SEGUNDO PREMIO";TAB(20)
60 FOR X=1 TO 5:PRINT INT(RND*10);:NEXT
70 PRINT:PRINT
80 PRINT "TERCER PREMIO";TAB(20)
90 FOR X=1 TO 5:PRINT INT(RND*10);:NEXT
100 PRINT:PRINT
110 PRINT:PRINT "TERMINACIONES":PRINT
120 FOR X=1 TO 16
130 FOR Y=1 TO 3
140 PRINT INT(10*RND);
150 NEXT Y
160 PRINT,
170 NEXT X
180 LOCATE 1,24
```

EJEMPLO 4

El ordenador elige al azar un número entero comprendido entre uno y cien y nosotros debemos adivinarlo en un máximo de seis intentos. Tras cada intento, el ordenador nos irá indicando si nuestro número es mayor, menor o igual que el elegido por él.

El programa es el siguiente:

```
10 N=INT(100*RND)+1
20 FOR X=1 TO 6
30 INPUT "NUMERO ELEGIDO";A
40 IF A>N THEN PRINT A;"ES MAYOR"
50 IF A<N THEN PRINT A;"ES MENOR"
60 IF A=N THEN PRINT "ACERTO EN";X;"JUGADAS"
:END
70 PRINT
80 NEXT
90 PRINT "EL NUMERO BUSCADO ES";N
```

EJEMPLO 5

Uno de los juegos de dados más sencillo y antiguo es el conocido con el nombre de "la jaula". En su origen se introducían tres dados en una jaula de alambre y se lanzaba ésta.

Si se apuestan A pesetas por el número N y éste no aparece se pierde lo apostado; si aparece sólo una vez, ni perdemos ni ganamos; si aparece dos veces, ganamos lo apostado y, por fin, si aparece tres veces, el doble de lo apostado.

Con el siguiente programa podemos jugar a dicho juego. Se comienza con 100 pesetas y se puede continuar jugando mientras se tenga dinero.

```
10 D=100
20 LOCATE 1,1:INPUT "¿CUANTO APUESTA?";A
30 IF A<0 OR A>D THEN 20
40 INPUT "¿NUMERO ELEGIDO?";N
50 X=INT(6*RND)+1:Y=INT(6*RND)+1:Z=INT(6*
RND)+1
60 D=D-A
70 LOCATE 1,5:PRINT "NUMEROS:";X;Y;Z
80 IF X=N THEN D=D+A
90 IF Y=N THEN D=D+A
100 IF Z=N THEN D=D+A
110 LOCATE 1,10
120 IF D>0 THEN PRINT "TIENE";D;"PESETAS":
GOTO 20
130 PRINT "ESTA ARRUINADO"
```

RANDOMIZE

Como se ha visto hasta ahora, la función RND no es realmente aleatoria; siempre que se conecta el ordenador se repite la misma secuencia rígida de números aleatorios. De esta forma es fácil, en un juego, conocer los números que van a salir.

Para solucionar el problema, nuestro ordenador posee una nueva orden:

```
RANDOMIZE X
```

Con esta orden, en lugar de comenzar a tomar números desde el principio de la lista, se comienza a partir de la posición número X.

Así, con el siguiente programa:

```
10 RANDOMIZE 100
20 FOR X=1 TO 20
30 PRINT RND
40 NEXT
```

se obtienen los 20 primeros números de la lista a partir de la posición número 100. Con:

```
10 RANDOMIZE 1
```

conseguiríamos los 20 primeros números de la lista.

Si no se indica el valor de X, es decir, si se introduce únicamente:

```
RANDOMIZE
```

al ejecutarse el programa aparecerá el mensaje:

```
Random number seed?
```

con lo que el ordenador nos pide que le indiquemos la posición de partida de la lista; es decir, el valor del parámetro X en lugar de introducirlo en el listado se introduce durante la ejecución del programa.

Realmente con esta modalidad de la función RANDOMIZE no solucionaremos nuestro problema, ya que el programador puede conocer los números que van a ir apareciendo. Para evitar esta posibilidad RANDOMIZE puede adoptar otra modalidad.

```
RANDOMIZE TIME
```

Ahora comienza la secuencia a partir de una posición que depende del tiempo que lleva funcionando nuestro ordenador, por lo que ya es prácticamente imposible prever los números que se van obteniendo.

PROGRAMAS PROPUESTOS

1) Obtener en la pantalla una apuesta de la lotería primitiva en la que aparezcan señalados los seis números premiados. Por ejemplo:

```
LOTERIA PRIMITIVA

 [ ]    8    15    22    29    36    43
  2     9    16    23    30    37   [ ]
  3    10    17    24    31    38   [ ]
  4    11    18   [ ]   [ ]    39    46
  5    12    19    26    33    40    47
  6    13    20   [ ]    34    41    48
  7    14    21    28    35    42    49
Ready
 [ ]
```

2) El ordenador tomará al azar dos números enteros positivos inferiores a 100 y nos preguntará cual es su suma. Si la respuesta es incorrecta se repetirá la pregunta, pero sólo una sola vez. Si la respuesta es correcta se repite el mismo proceso para la multiplicación.

3) Construir el programa inverso del ejemplo 4, nosotros elegimos un número del 1 al 100 y el ordenador debe encontrarlo, tomando aproximaciones sucesivas, al azar.

4) Hacer un programa para jugar al MASTER–MIND. El ordenador tomará un número de 4 cifras distintas (la primera no nula) y contabilizará los intentos efectuados. En cada intento, hay un máximo de 9 oportunidades, informará del número de muertos (cifras acertadas en la misma posición) y heridos (cifras acertadas en distinta posición).

5) El siguiente resultado numérico le puede permitir impresionar a sus amistades por sus dotes de adivino:
Tomar un número entero cualquiera, por ejemplo el 85, multiplicarlo por 3 y sumar el número obtenido y los dos anteriores a él, en nuestro caso 255+254+253=762. Sumar ahora todas las cifras de la suma anterior, 7+6+2=15. Si la suma tiene varias cifras, volvemos a sumar, 1+5=6. Repetir el proceso hasta tener un número de una sola cifra; ésta será siempre 6.

Comprobar con números enteros tomados al azar, menores que un millón, que este resultado es cierto.

6) El siguiente juego es una mezcla de MASTERMIND y de AHORCADO:
Un jugador introduce en el ordenador una palabra (para no complicarlo demasiado de 3, 4 ó 5 letras y sin repetirse ninguna), y el otro jugador debe adivinarla. Este jugador debe introducir palabras que tengan el mismo número de letras que la buscada y el ordenador irá informando de si el número de letras que coinciden en ambas palabras, sin importar el orden, es par o impar.
Por ejemplo, el desarrollo de una partida podría ser:

```
LA PALABRA A ENCONTRAR TIENE 5 LETRAS

PALABRA? CAMPO
impar

PALABRA? SALON
par

PALABRA? JUEGO
impar

PALABRA? CLASE
impar

PALABRA? LIBRO

HAS ACERTADO EN 5 INTENTOS
```

11

Variables dimensionales

En bastantes ocasiones nos encontraremos con el problema de asimilar valores a un número elevado de variables. Por el procedimiento que hemos visto hasta ahora, esta asignación resulta costosa e incluso, en algunos casos, imposible de llevar a cabo en la práctica.

Si tenemos que asignar por el teclado valores de 11 variables distintas en un programa, podemos hacerlo con 11 instrucciones como éstas:

```
10 INPUT a
20 INPUT b
...
```

ó bien con:

```
10 INPUT a,b,c,.....
```

Este procedimiento no parece muy adecuado sobre todo si el número de variables es todavía mayor. Tampoco es muy útil si posteriormente necesitamos manejar el valor introducido con un número de orden dado.

Sin embargo, hay otra manera más cómoda de trabajar cuando el número de variables es grande. Para éstos casos se emplean variables con subíndice:

a(0),a(1),a(2),...

Todas ellas llevan la misma letra, pero se distingue una de otra en el número que la acompaña entre paréntesis.

Así, en el ejemplo anterior el problema se resuelve empleando las variables:

a(0),a(1),a(2),...,a(10)

y el bucle:

```
10 FOR X=1 TO 10
20 INPUT A(X)
30 NEXT
```

Si ahora queremos imprimir el octavo número introducido bastará con:

```
PRINT a(7)
```

Es importante señalar que cuando queremos cursar un número de variables mayor que 11, será preciso dimensionarlas, empleando la instrucción:

DIM

que puede adoptar las formas siguientes:

```
DIM a(N)
DIM a%(N)
DIM a$(N)
```

según sea el tipo de variables que empleemos.

Con esta instrucción se reserva en memoria espacio para almacenar N+1 variables:

```
desde a(0) hasta a(N) con DIM a(N)
desde a%(0) hasta a%(N) con DIM a%(N)
desde a$(0) hasta a$(N) con DIM a$(N)
```

Así, con:

```
DIM a(100)
```

tendremos la posibilidad de usar las variables:

```
a(0),a(1),a(2),...a(100)
```

que, inicialmente, una vez dimensionadas, tomarán todas ellas el valor 0.

Con:

```
DIM a$(40)
```

tendremos la posibilidad de usar 41 variables, desde:

a$(0) hasta a$(40)

que, por ejemplo, podrán usarse para que el ordenador recuerde el nombre de los 40 alumnos de una clase y los asocie a un número de orden. En este caso podemos dejar sin asignar ningún valor a la variable a$(0).

Inmediatamente después de la anterior instrucción el valor que tomaría cada una de las variables dimensionadas, en esta ocasión, sería la cadena vacía.

NOTAS

1) En el caso de que el número de variables a emplear sea 11 ó menor que 11, no se necesita dimensionarlas, ya que el ordenador reserva espacio suficiente en la memoria automáticamente.

2) Si pretendemos dimensionar por segunda vez una misma variable, o sea:

```
10 DIM A(20)
20 DIM A(40)
```

el ordenador nos suministra el mensaje de error:

Array already dimensioned

advirtiéndonos que ya había sido dimensionada anteriormente.

3) Si, sin dimensionar la variable, empleamos:

PRINT b(24)

se obtiene el mensaje de error:

Subscript out of range

sin embargo no habrá aparecido con:

PRINT b(2)

por la razón apuntada en la nota 1.

4) El programa:

```
10 A(0)=12
20 DIM A(100)
```

también nos dará el error:

```
Array already dimensioned in 20
```

ya que, con la línea 10 se supone implícitamente que se ha dado la instrucción:

```
DIM a(10)
```

5) Hay que tener presente que con:

```
DIM a(100)
```

no dimensionamos 100 variables sino 101, ya que empezamos con a(0). Aunque hay ocasiones en que por comodidad esta variable no se emplea.

ERASE

En cierto modo ésta instrucción efectúa la operación inversa a la instrucción DIM.

ERASE sirve para liberar al ordenador de la zona de memoria reservada para las variables dimensionadas anteriormente.

Así:

```
ERASE a$,b
```

anula los espacios en la memoria que ocupaban las variables dimensionadas a$ y b, quedando sin efecto las instrucciones:

```
DIM a$(M)
DIM b(N)
```

y olvidando todos los valores que tomaban las variables a$(0),...,a$(M) y b(0),...,b(N).

Con el programa:

```
10 DIM A$(12):DIM B$(20)
20 A$(6)="ERASE":B(10)=10
30 PRINT A$(6);B(10)
40 ERASE A$,B
50 PRINT A$(6);B(10)
```

se pone claramente de manifiesto lo dicho anteriormente.

ORDENACIONES

El problema de ordenar un conjunto de variables numéricas o alfanuméricas ha tenido desde siempre gran importancia en Informática.

Nuestro propósito es el de iniciar el tema sin intención, ni mucho menos de agotarlo.

Primero veremos el método de la burbuja que es uno de los más sencillos, aunque no de los más rápidos.

Imaginemos que tenemos 100 variables numéricas y que hay que ordenarlas de menor a mayor. Primero, y de forma aleatoria, vamos a asignar valores a las 100 variables con el programa:

```
20 MODE 2
30 DIM A(100)
40 FOR X=1 TO 100
50 A(X)=INT(RND*1000)
60 NEXT
```

cada variable tomará como valor un número entero menor que 1000 (línea 30).

No emplearemos la variable a(0).

El programa que ordenará dichas variables será el siguiente:

```
70 N=100
80 C=0
90 FOR X=1 TO N-1
100 IF A(X)<=A(X+1) THEN 140
110 B=A(X):C=C+1
120 A(X)=A(X+1)
130 A(X+1)=B
140 NEXT
150 IF C<>0 THEN N=N-1:GOTO 80
```

NOTAS

1) Con este programa se compara el primer elemento de la lista con el segundo y si no están ordenados se intercambian.

2) Después de la primera vuelta el mayor de los números estaría colocado en último lugar.

3) Volveremos a dar varias pasadas dejando el último elemento de la pasada anterior fuera de las comparaciones ya que es el mayor de ellos.

4) En cada pasada controlamos el número de cambios efectuados con la variable C. Si C=0 quiere decir ésto que la tarea ya se ha terminado.

5) Observar que introduciendo:

```
10 DEFINT A-Z
```

el programa gana en rapidez al usar variables enteras.

6) Para comprobar que realmente se han ordenado, añadir:

```
160 FOR X=1 TO 100
170 PRINT A(x)
180 NEXT
```

Otro método sencillo y más rápido es el llamado de INSERCION. Primero se ordenan los dos primeros números de la lista y, depués se van colocando en el lugar que les corresponde los restantes; este método es análogo al empleado al introducir una nueva ficha en un fichero ya ordenado.

```
10 DIM A(100)
20 FOR X=1 TO 100
30 A(X)=INT(RND*1000)
40 NEXT
50 FOR I=1 TO 100
60 FOR J=1 TO I-1
70 IF A(J)<=A(I) THEN 120
80 A=A(I)
90 FOR K=I TO J+1 STEP -1
100 A(K)=A(K-1):NEXT K
110 A(J)=A:GOTO 130
120 NEXT j
130 NEXT i
140 FOR X=1 TO 100
150 PRINT A(X)
160 NEXT
```

Vamos a emplear el método de la burbuja para ordenar alfabéticamente diez palabras introducidas con el programa:

```
10 FOR X=1 TO 10
20 INPUT A$(X)
30 A$(X)=UPPER$(A$(X))
40 NEXT
```

Ya que la comparación se hace por medio de los códigos ASCII de los caracteres que forman cada palabra y las minúsculas tienen códigos menores que las mayúsculas, es necesario introducir la línea 30 que convierte todas las letras en mayúsculas, ya que de no introducirla, la letra b estaría ordenada alfabéticamente ántes que la A.

```
50 MODE 1
60 N=10
70 C=0
80 FOR X=1 TO N-1
90 IF A$(X)<=A$(X+1) THEN 120
100 B$=A$(X):C=C+1
110 A$(X)=A$(X+1):A$(X+1)=B$
120 NEXT
130 IF C<>0 THEN N=N-1:GOTO 70
```

El procedimiento es análogo al anterior, por lo que no son necesarios más comentarios.

En los modelos de AMSTRAD con teclado en español, este procedimiento falla con las letras "ñ" y "Ñ". El motivo es que se les asigna los códigos 171 y 161 respectivamente, con lo que el ordenador considera estas letras como las últimas del alfabeto.

MATRICES

Hasta ahora sólo hemos trabajado con variables dimensionadas con un sólo número, las llamadas listas. Pero también existe la posibilidad de unas variables con dos ó más subíndices. Por ejemplo, podremos usar variables como las siguientes:

a(1,1) a(1,2) a(1,3) a(1,4)

a(2,1) a(2,2) a(2,3) a(2,4)

a(3,1) a(3,2) a(3,3) a(3,4)

Se trata de una tabla de variables llamada matriz, de tal manera que para referirnos a una variable determinada, habrá que indicar la fila y la columna en que se encuentra situada.

Así, la variable a(2,3) es la que ocupa la fila segunda y la columna tercera.

Una tabla de este tipo podría almacenar, por ejemplo las calificaciones de tres alumnos en cuatro asignaturas diferentes de tal manera que,

```
a(i,j)
```

es la calificación del alumno número i en la asignatura número j.

También se pueden emplear variables con tres, cuatro o más subíndices. Ahora vamos a ocuparnos de los casos en que será necesario dimensionarlas.

Si poseen dos ó tres subíndices sólo habrá que dimensionar si uno de los dos es mayor que 10. Por ejemplo con DIM a(5,12) reservamos en la memoria espacio suficiente para almacenar una tabla de 6 filas con trece columnas; en totai 78 variables:

```
a(0,0) a(0,1).... a(0,12)
a(1,0) a(1,1).... a(1,12)
...
a(5,0) a(5,1).... a(5,12)
```

con DIM a(3,2,20) podemos emplear 4*3*21=252 variables.

También se pueden usar variables con más de tres subíndices pero en este caso, siempre habrá que dimensionarlas:

```
DIM a(4,3,2,5)
```

reserva espacio para 5*4*3*6=360 variables identificándose cada una por cuatro índices.

En todos estos últimos ejemplos hemos visto variables numéricas, pero también se pueden emplear variables alfanuméricas.

EJEMPLO

Con el empleo de variables con cuatro subíndices podríamos llevar el control del dinero recaudado por una de las doce tiendas de una cadena que poseen cada una de ellas seis secciones y reseñando el día y el mes.

En este caso se dimensionaría así:

```
DIM a(12,6,31,12)
```

y el valor de la variable a(7,2,5,9) será el dinero recaudado por el establecimiento número 7 en su sección dos el día 5 del mes 9 (Septiembre).

Introducidos los datos en el ordenador, día a día, podríamos calcular por ejemplo todo el dinero recaudado en el establecimiento 7 sección segunda durante el mes noveno:

```
FOR X=1 TO 30
S=S+A(7,2,X,9)
NEXT
```

o bien el dinero recaudado por todas las secciones de este establecimiento en el mes de Septiembre:

```
FOR X=1 TO 30
FOR Y=1 TO 6
S=S+A(7,Y,X,9)
NEXT:NEXT
```

EJERCICIOS PROPUESTOS

1) Simular con el ordenador el juego del BINGO. En la pantalla van apareciendo a intervalos regulares de tiempo los diversos números, destacándose en inverso el último aparecido.

2) Construir cuadrados mágicos de orden n impar siguiendo el siguiente algoritmo:

   a) Se sitúa el 1 encima de la posición central; es decir, si n=3 se coloca en (1,2), si n=5 en (2,3), etc.

   b) Los siguientes números se colocan en el orden natural siguiendo la dirección noreste. Si la posición de llegada ya está ocupada, seguimos la dirección noroeste.

   c) En caso de salir de los límites del cuadrado:
      - si sobrepasamos la columna por la izquierda (derecha) colocamos el siguiente número en la última (primera) columna de la fila que le corresponde.
      - Si sobrepasamos las filas colocamos el número siguiente en la última fila de la columna que le corresponde.

3) Construir un programa para jugar a la lotería primitiva. El ordenador solicitará cuántos números se desean elegir y los obtendrá al azar, presentando en pantalla el boleto ya rellenado.

4) Un álbum de cromos consta de doscientos cromos. Calcular por simulación cuántos cromos se necesita comprar para rellenar el album. Presentar en una tabla 10x20 el número de veces que nos ha aparecido cada cromo.

12

Introducción de datos

Habitualmente en todos los programas necesitamos almacenar un conjunto de datos, a veces elevado, para posteriormente trabajar con ellos. Esta introducción de datos puede hacerse de dos maneras:

a) Mediante la utilización de las instrucciones LET o INPUT.

b) Con las sentencias READ (leer) y DATA (datos).

En el primer caso, se definen los datos uno a uno. Por ejemplo supongamos que queremos introducir en el ordenador los datos:

2,0.5,–3,"JUAN","CASA"

Podemos hacerlo con:

```
10 A=2
20 B=0.5
30 C=-3
40 A$="JUAN"
50 B$="CASA"
```

En el segundo caso, podemos introducir todos los datos agrupados precedidos de la instrucción DATA y, después con la instrucción READ, los podremos leer.

El ejemplo anterior, con estas dos nuevas sentencias quedará en la forma:

```
10 READ A,B,C,A$,B$
20 DATA 2,0.5,-3,JUAN,CASA
```

En este caso a la primera variable que se lee (A) se le asocia el primer dato (2), a la segunda (B) el segundo dato 0.5, y así sucesivamente. Es decir, READ lee el nombre de la variable y, por medio de DATA, se le asigna valores de una forma ordenada.

Es evidente que este segundo método es mucho más corto, sobre todo cuando el número de datos introducidos es muy grande.

NOTAS

1) Siempre que exista una sentencia READ en un programa, tiene que haber una sentencia DATA.

2) Es indiferente el lugar donde situemos DATA, ya que cuando aparece en el desarrollo de un programa una instrucción READ, se analiza desde la primera línea hasta hallar el DATA correspondiente. Así conseguiremos el mismo efecto del ejemplo anterior si escribimos:

```
10 DATA 2,0.5,-3,JUAN,CASA
20 READ A,B,C,A$,B$
```

3) DATA puede ir incluso detrás de una sentencia STOP o END. Por ejemplo:

```
10 READ A,B
20 PRINT A+B
30 STOP
40 DATA 2,3
```

presenta en pantalla el número 5, ya que es el resultado de sumar A que vale 2, con B que vale 3.

Aunque la instrucción DATA puede colocarse en cualquier parte, es conveniente ponerla al principio o al final del programa, con objeto de clarificarlo.

Lógicamente, el número de variables que se leen, debe coincidir con el número de datos que se introducen. Sin embargo, podemos introducir más datos que nombres de variables, y el ordenador ignorará las restantes.

```
10 READ A,B,C
20 PRINT A,B,C
30 DATA 4,-6,3,8,6
```

Ahora bien, si existen más nombres de variables que datos, el ordenador presentará un mensaje de error. Así, al escribir:

```
10 READ A$,B$,C$
20 PRINT A$,B$,C$
30 DATA SILLA,MESA
```

en pantalla aparecerá:

```
DATA exhausted in 10
```

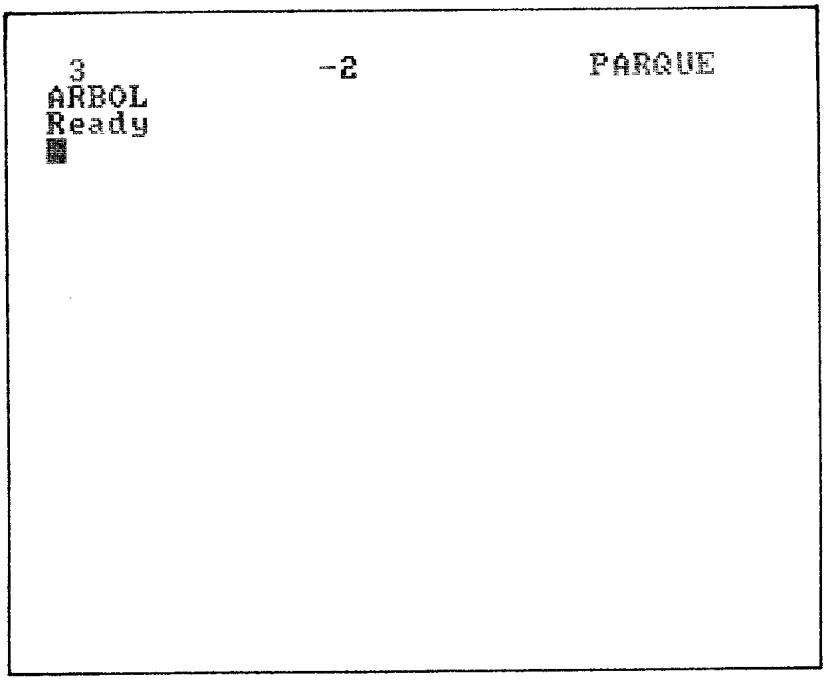
READ y DATA podemos utilizarlas en muchas situaciones, lo cual, amplia enormemente las posibilidades de estas instrucciones cuando realizamos

cualquier programa. Veamos algunas de las características más sobresalientes:

1– Pueden funcionar más de una sentencia READ con un sólo DATA. Por ejemplo con:

```
10 READ A,A$
20 READ B,B$
30 PRINT A,B,A$,B$
40 DATA 3,PARQUE,-2,ARBOL
```

en pantalla aparece:



ya que una vez asociado las variables correspondientes al primer READ, al segundo le asociará los datos que faltan, es decir, –2 y ARBOL.

2– También con un sólo READ pueden funcionar varios DATA. En este caso cuando se acaban los datos asociados al primer DATA, se continúa con los del siguiente, y así sucesivamente.

Por ejemplo con:

```
10 READ A$,B$,C$,D$
20 PRINT A$,B$,C$,D$
30 DATA LUZ,DIA
40 DATA SOMBRA, NOCHE
```

en pantalla aparece:

```
LUZ           DIA           SOMBRA
NOCHE
Ready
```

3- READ también puede ir dentro de un bucle. Por ejemplo:

```
10 DATA 1,-1,2,-2,3,-3
20 FOR X=1 TO 6
30 READ A
40 PRINT A
50 NEXT
```

este programa tiene el inconveniente de que los seis valores están asociados a la variable A y, por tanto, no podemos recuperarlos después

de ejecutarse el bucle. Este problema se soluciona trabajando con variables dimensionadas.
Así, con el programa:

```
10 DATA 1,-1,2,-2,3,-3
20 FOR X=1 TO 6
30 READ A(X)
40 NEXT
50 FOR X=6 TO 1 STEP -1
60 PRINT A(X)
70 NEXT
```

logramos que se escriban los seis números en orden inverso.

4– También podemos leer únicamente el dato que ocupa un determinado lugar con el siguiente programa:

```
10 INPUT "QUE DATO QUIERES";N
20 DATA 1,2,3,4,5,6,7,8,9,10
30 X=X+1:READ A:IF N<>X THEN 30
40 PRINT"EL DATO";N;"ES";A
```

Esto puede ser interesante para resolver problemas como el siguiente:

– Dado el número de un mes, imprimir su nombre.

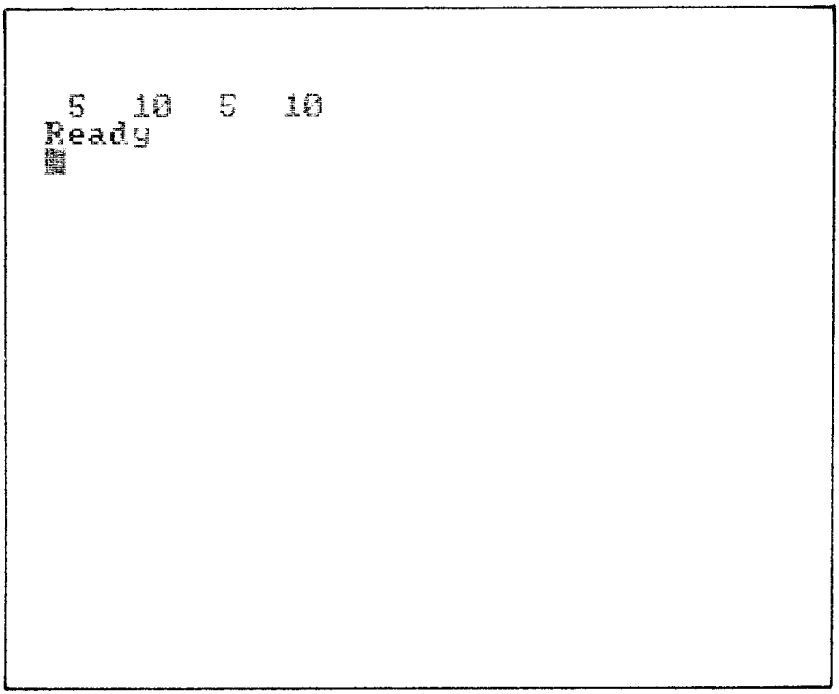
```
10 INPUT "NUMERO DEL MES";N
20 CLS
30 DATA ENERO,FEBRERO,MARZO,ABRIL,MAYO,
JUNIO
40 DATA JULIO,AGOSTO,SETIEMBRE,OCTUBRE,
NOVIEMBRE,DICIEMBRE
50 X=X+1:READ A$:IF X<>N THEN 50
60 PRINT"EL MES NUMERO";N;"ES ";A$
```

RESTORE

Esta instrucción tiene como misión restaurar los valores almacenados en DATA, una vez que ya han sido utilizados. Para comprender cómo funciona veamos el siguiente programa:

```
10 DATA 5,10
20 READ A,B
30 RESTORE
40 READ C,D
50 PRINT A;B;C;D
```

en pantalla aparecerá:



Esto es debido a que con la línea 20 asociamos a la variable A y B los valores 5 y 10 respectivamente. Con la línea 30 volvemos a restaurar éstos valores en DATA aunque hayan sido asignados a A y B. Por eso al ejecutarse la sentencia READ de la línea 40 se da a C el valor 5 y a D el valor 10. Si la línea 30 no existiera, el ordenador presentaría el mensaje de error:

```
DATA exhausted in 40
```

ya que C y D no pueden coger valores de ningún DATA.

Si se tienen varias líneas DATA podemos consignar detrás de la sentencia RESTORE el número de línea correspondiente al DATA que queremos restaurar:

```
10 DATA 1,2,3
20 DATA 4,5,6
30 DATA 7,8,9
40 READ A,B,C,D,E,F,G,H
50 RESTORE 30
60 READ I,J,K
70 PRINT A,B,C,D,E,F,G,H,I,J,K
```

Con la línea 40 obtenemos:

A=1 B=2 C=3 D=4 E=5 F=6 G=7 H=8

Después (línea 50) restauramos los datos 7 y 8, por tanto, con la línea 60 tenemos:

I=7 J=8 K=9

EJERCICIOS PROPUESTOS

1) Construir un pequeño listín telefónico.
2) Hacer un breve diccionario castellano–inglés.
3) Dada una frase, escribirla en morse. Tener presente únicamente los números y las letras.

13
El tiempo

Dos son las cuestiones importantes que se van a estudiar en este capítulo: la medición del tiempo y las interrupciones.

Haciendo uso del reloj interno del ordenador, podremos medir el tiempo que éste emplea en realizar una determinada tarea.

Con las sentencias de interrupción podremos conseguir que el ordenador abandone una tarea para dedicarse a otra y regresar después a la primera.

TIME

Podría considerarse como una variable que toma el valor 0 en el preciso instante de conectar el ordenador y que va incrementándose en una unidad cada 1/300 de segundo transcurrido.

Visualicemos, en la pantalla, los valores que va tomando TIME:

```
10 MODE 1
20 LOCATE 15,12
30 PRINT TIME
40 GOTO 20
```

solamente las operaciones de carga y grabación en disco o cassette hacen que este ordenador de tiempos se interrumpa.

Ahora es facíl conocer, por ejemplo, el tiempo en segundos que tarda en ejecutarse un bucle:

```
10 A=TIME
20 FOR X=1 TO 10000
30 NEXT
40 B=TIME
50 PRINT INT((B-A)/300)
```

en las líneas 10 y 40 se memorizan los valores que tiene TIME antes y después del bucle. Por lo que B–A será el tiempo, en unidades de 1/300 segundos, que ha tardado el ordenador en ejecutar el bucle. Para que el resultado venga en segundos, dividiremos por 300.

Las dos últimas líneas del programa se podrían haber resumindo en una:

```
40 PRINT INT((TIME-A)/300)
```

Ahora se puede comprobar la veracidad de la afirmación que se hizo en un capítulo anterior de que ganábamos en rapidez usando variables enteras. Cambiemos la línea 20 por:

```
20 FOR X%=1 TO 10000
```

y observaremos que el tiempo empleado es sensiblemente menor.

INTERRUPCIONES

Es grande la facilidad que tiene AMSTRAD para interrumpir una tarea y dedicarse a otra. Las instrucciones que usaremos para ello son AFTER y EVERY.

AFTER (después)

Puede tomar una de estas dos formas:

```
AFTER a GOSUB n
AFTER a,b GOSUB n
```

Veamos la primera: Al cabo de a unidades de tiempo, cada unidad equivale a 1/50 de segundo, salta a la subrutina de la línea n, cualquiera que fuese la tarea que en ese momento estuviese ejecutando.

Así:

```
AFTER 100 GOSUB n
```

hace que al cabo de 2 segundos, salte a la subrutina de la línea n.

El valor máximo que puede tomar a es 32767.

Con:

```
AFTER a,b GOSUB n
```

hace lo mismo que la instrucción anterior, pero especificando el temporizador b que lleva la cuenta de cuando ha llegado el momento de saltar a la subrutina de la línea n.

El ordenador tiene 4 temporizadores que están numerados de 0 a 3. Así, con:

```
AFTER 5Ø,1 GOSUB n
```

al cabo de un segundo, controlado por el temporizador número 1, saltará a la subrutina de la línea n.

Si no se especifica ningún temporizador, el encargado de realizar la tarea será el temporizador número 0.

Es importante hacer notar que no se trata de una simple pausa, sino que el ordenador, después de AFTER, sigue trabajando normalmente hasta que se cumple el tiempo y es entonces cuando salta a la subrutina indicada para luego volver a la tarea que estaba realizando anteriormente.

El temporizador b toma el valor a y a partir de ese momento se va decrementando hasta que llega a cero y es en ese momento cuando salta a la subrutina.

EJEMPLO

Así, en el programa:

```
1Ø AFTER 1ØØ GOSUB 6Ø
2Ø FOR X=1 TO 1ØØ
3Ø PRINT X;
4Ø NEXT
5Ø STOP
6Ø PRINT "HA SALTADO"
7Ø FOR X=1 TO 3ØØØ:NEXT
8Ø RETURN
```

se interrumpe el bucle 20–40 cuando van transcurridos 2 segundos y el ordenador salta a la subrutina de la línea 60, imprime un mensaje, ejecuta otro bucle para perder un poco de tiempo y, por fin, vuelve al bucle primero.

Con AFTER se pueden controlar hasta 4 subrutinas, una por cada temporizador, siendo el número 3 el de máxima y el 0 el de mínima prioridad. Esto quiere decir que sí, en un momento determinado, dos instrucciones AFTER pretenden hacer saltar a dos subrutinas diferentes, será la subrutina correspondiente al temporizador de máxima prioridad la que se ejecute primero.

EVERY (cada)

También puede tomar una de las dos formas siguientes:

```
EVERY a GOSUB n
EVERY a,b GOSUB n
```

donde los parámetros a,b y n tienen el mismo significado que en el caso anterior.

La diferencia entre AFTER (después) y EVERY (cada) queda bien patente en la traducción al castellano.

AFTER, transcurrido el tiempo especificado, salta una única vez a la subrutina de la línea n.

EVERY, salta a intervalos regulares de a/50 segundos, y tantas veces como sea posible, a la subrutina de la línea n.

Con el siguiente programa se verá mejor la diferencia:

```
10 MODE 1
20 EVERY  50 GOSUB 100
30 GOTO 30
100 N=N+1
110 LOCATE 1,13
120 PRINT"HA SALTADO";N;"VECES"
130 RETURN
```

probarlo tal y como está y luego cambiar AFTER por EVERY para notar la diferencia entre una y otra instrucción.

Para parar el programa habrá que hacer un BREAK pues en caso contrario no saldrá nunca de la línea 30.

También en EVERY, el parámetro a puede tomar, como máximo, el valor 32767.

La cuestión de los temporizadores funcionan igual en ambas instrucciones.

Una instrucción EVERY anula una AFTER anterior en el caso de que ambas estén controladas por el mismo temporizador.

REMAIN

Su forma general es:

```
REMAIN (T)
```

donde T es el número de un temporizador.

Con:

```
PRINT REMAIN (T)
```

se imprime en pantalla el tiempo que faltaba para que el temporizador número T quede a cero y además desactiva ese temporizador.

DI e EI

En ocasiones nos puede interesar prescindir de las interrupciones, y para ello existe la instrucción DI.

Si en un momento determinado queremos volver a activarlas, lo lograremos con la instrucción EI.

14

Algo más sobre el teclado

En muchos programas y sobre todo en los referentes a juegos, es necesario detectar si una determinada tecla ha sido presionada o no. Esto puede hacerse mediante la instrucción INPUT, pero en ese caso, tenemos el inconveniente de que el programa se detiene hasta que introducimos el dato. Supongamos que queremos confeccionar un programa para que un móvil se mueva horizontalmente por la pantalla en la dirección que le señalemos presionando determinadas teclas (Z para la izquierda y X para la derecha). Con lo ya visto hasta ahora, el programa sería:

```
10 MODE 0
20 A$=" "+CHR$(248)+" "
30 C=1
40 LOCATE C,12:PRINT A$
50 LOCATE 1,25:INPUT B$
60 B$=UPPER$ (B$)
70 IF B$<>"Z" AND B$<>"X" THEN 50
80 IF B$="Z" THEN C=C-1
90 IF B$="X" THEN C=C+1
100 IF C>18 THEN C=18
110 IF C<1 THEN C=1
120 GOTO 40
```

En la línea 10 hemos definido una variable A$ con el carácter 248 que corresponde a un muñeco precedido y seguido de un espacio en blanco con el fin de que, al moverse, borre el muñeco anterior.

En la línea 55 utilizaremos la instrucción UPPER$ para trabajar solo con mayúsculas.

El programa mejoraría mucho si el ordenador fuera capaz de detectar directamente si se han pulsado las teclas Z o X. Esto puede conseguirse con:

INKEY$

Cuando el ordenador se encuentra con la instrucción INKEY$ investiga si en ese preciso momento hay una tecla pulsada. En caso de que ésto suceda, asocia a la variable alfanumérica INKEY$ el valor del carácter de dicha tecla. Si no hay ninguna tecla pulsada INKEY$ tomará el valor de la cadena vacía.

Veamos unos pequeños ejemplos que clarificarán lo dicho anteriormente:

EJEMPLO 1:

```
10 MODE 1
20 IF INKEY$="Z" THEN PRINT"TECLA PULSADA:Z"
30 GOTO 20
```

Este sencillo programa escribe en pantalla cuando se ha pulsado la tecla Z. Es necesario colocar la línea 30 ya que, la rapidez con la que se desarrolla la lectura del teclado, es tan alta que pasa por la línea 20 antes de que podamos presionar la tecla Z.

EJEMPLO 2:

Con el siguiente programa conseguimos que el ordenador imprima en pantalla el carácter correspondiente a la tecla pulsada.

```
10 MODE 1
20 PRINT INKEY$;
30 GOTO 20
```

EJEMPLO 3:

Puede ser interesante que un determinado programa se detenga hasta que se presione cualquier tecla. Esto se consigue fácilmente con:

```
10 MODE 1
20 LOCATE 4,12
30 PRINT"PARA CONTINUAR, PULSAR UNA TECLA"
40 IF INKEY$="" THEN 40
50 CLS
60 PRINT"CONTINUAMOS...."
```

Con la línea 40 obligamos al ordenador a que espere hasta que se pulse una tecla.

Ahora ya podemos mejorar el programa incial. Bastará con cambiar las líneas 50 y 60 por:

```
50 LOCATE 1,25
60 B$=INKEY$
```

Una instrucción parecida a INKEY$ pero todavía con más posibilidades es:

INKEY

Con INKEY no solamente se investiga si se ha presionado una tecla o no sino que también detecta si, simultáneamente a una tecla determinada, hemos pulsado MAYS o CONTROL.

La forma general de la función INKEY es:

```
INKEY (N)
```

donde N es un número asociado a la tecla en cuestión. Este número se encontrará en los modelos 664 y 6128 en una tabla situada encima de la unidad de disco. Para los modelos 464 y 472 deberá consultarse el manual de instrucciones. Por ejemplo, la tecla A tiene asignado el número 69. El valor numérico que toma INKEY (69) viene dado por la siguiente tabla:

A pulsada MAYS pulsada CONTROL pulsada INKEY (69)

A pulsada	MAYS pulsada	CONTROL pulsada	INKEY (69)
NO	NO	NO	-1
SI	NO	NO	0
SI	SI	NO	32
SI	NO	SI	128
SI	SI	SI	160

Esto sucede con cualquier tecla. Para comprobar el valor que toma INKEY (69) podemos hacer el siguiente programa:

```
10 MODE 1
20 X=INKEY(69)
30 LOCATE 1,12
```

```
40 IF X=-1 THEN PRINT"NO ESTAS PULSANDO
LA A";SPC(16) ELSE PRINT"ESTAS PULSANDO
LA A ";
50 IF X=0 THEN PRINT"Y NADA MAS"
60 IF X=32 THEN PRINT"Y MAYS"
65 IF X=128 THEN PRINT"Y CONTROL"
70 IF X=160 THEN PRINT",MAYS Y CONTROL"
90 GOTO 20
```

NOTA

1) Observar que, mientras INKEY$ toma valores alfanuméricos, INKEY (N) toma valores numéricos.

2) Una de las aplicaciones más interesantes de esta función, es la posibilidad de detectar si dos teclas son presionadas. Veamos como podemos hacerlo para el caso de las teclas correspondientes a los números 1 y 2.

   El código de la tecla 1 es 64 y el de la tecla 2, 65. Con estos dos números construimos la siguiente expresión:

   ABS (INKEY (64) * 2 + INKEY (65))

   Esta expresión toma los siguientes valores:

   3 si no se presionan ninguna de las dos teclas
   1 si únicamente presionamos la tecla 1.
   2 si únicamente presionamos la tecla 2.
   0 si presionamos a la vez las teclas 1 y 2.

   Podemos verificar estos resultados con el programa:

```
10 MODE 1
20 A=ABS(INKEY(64)*2+INKEY(65))
30 PRINT A;
40 GOTO 20
```

A continuación se estudiará una instrucción que nos facilitará, de una forma considerable, la introducción en el ordenador de expresiones habituales en la mayoría de nuestros trabajos. Esta instrucción es:

KEY

Normalmente, al presionar una tecla, aparece un único carácter en la pantalla. Se ganaría en rapidez si algunas de las expresiones que utilizamos más

a menudo se obtuvieran presionando una única tecla. Supongamos una expresión como:

```
CLS : LIST
```

que se utiliza muchas veces. Para introducirla en el ordenador es necesario escribir cada uno de los caracteres que la componen. Con KEY lograremos almacenar toda la expresión en una tecla solamente; de tal forma que, al pulsar dicha tecla nuestro ordenador admitirá las dos instrucciones que componen la expresión.

Si utilizamos KEY, las únicas teclas donde podemos almacenar cualquier expresión son las correspondientes al teclado numérico. Lo primero que debe conocerse son los códigos, llamados expandibles, que están asociados a cada una de dichas teclas. Estos códigos son:

Tecla	Cógigo
f–0	128 ó 0
f–1	129 ó 1
f–2	130 ó 2
f–3	131 ó 3
f–4	132 ó 4
f–5	133 ó 5
f–6	134 ó 6
f–7	135 ó 7
f–8	136 ó 8
f–9	137 ó 9
	138 ó 10

También la tecla ENTER tiene asociado un código expandible que será:

```
139 ó 11
```

si se presiona sola, y el

```
140 ó 12
```

si la presionamos junto con CONTROL

Una vez visto ésto, ya podemos obtener la expresión CLS:LIST al pulsar la tecla f–0 sin más que introducir:

```
KEY 128, "CLS:LIST"
```

El mismo efecto se conseguirá con:

```
KEY 0 , "CLS:LIST"
```

Ahora con f-0 y ENTER se borra la pantalla y se lista el programa que haya en la memoria. Podemos hacer que se ejecuten las dos instrucciones directamente sin pulsar ENTER con:

```
KEY 0 , "CLS:LIST" + CHR$ (13)
```

ya que CHR$ (13) hace las veces de ENTER.

NOTAS

1) Podemos definir las teclas con un programa grabado previamente en disco o cinta.

2) NEW no borra las expresiones definidas y asignadas de antemano. Para borrarlas será necesario desconectar el aparato o hacer un RESET.

También podemos redefinir teclas fuera del teclado numérico. Para hacerlo utilizaremos la instrucción

KEY DEF

Veamos un ejemplo: la tecla

vamos a reconvertirla para que, al pulsarla, se introduzcan en el ordenador las instrucciones

```
"CLS:LIST"
```

Procederemos del siguiente modo:

- Elegimos un código expandible. Estos códigos son valores comprendidos entre 128 y 159. Como hasta el 140 se utilizan para las teclas del teclado numérico, tomaremos un código mayor que 140. Por ejemplo, 150.
- Asociaremos al código elegido la expresión que deseamos. En nuestro caso

```
KEY 150 , "CLS:LIST" + CHR$ (13)
```

Añadimos CHR$ (13) para que CLS:LIST se ejecute automáticamente.

- Necesitaremos conocer el número asignado a nuestra tecla. Recordar que ese número se encuentra en la parte superior de la unidad de disco. En nuestro caso, éste es 39.
- Con

```
KEY DEF 39,1,150
```

asignamos a la tecla 39 la expresión asociada al código expandible 150.

El número situado entre 39 y 150 puede ser 1 ó 0 para indicar si existe o no repetición al mantener presionada la tecla.

A una misma tecla le podemos asignar tres expresiones según se presione sola, con MAYS o con CONTROL. Por ejemplo, a la tecla número 39 vamos a asignarle tres expresiones:

```
"CLS:LIST"
"MODE 1"
"MODE 2"
```

Esto se logra con

```
KEY 150, "CLS:LIST" + CHR$ (13)
KEY 151, "MODE 1" + CHR$ (13)
KEY 152, "MODE 2" + CHR$ (13)
KEY DEF 39,0,150,151,152
```

donde los tres números últimos son los códigos expandibles correspondientes a los tres casos posibles siguientes:

1.- Presionando solo la tecla 39
2.- Presionando la tecla 39 junto con MAYS
3.- Presionando la tecla 39 junto con CONTROL

Una de las utilidades más inmediatas de esta instrucción es el poder asignar a alguna tecla funciones distintas de las que tiene inicialmente. Así podemos hacer que se borre el programa pulsando la tecla ESC de la siguiente forma:

```
10 KEY 159,"NEW"+CHR$(13)
20 KEY DEF 66,1,159,159
```

Pero todavía, con KEY DEF podemos resolver otros problemas. En la versión en castellano de nuestro ordenador AMSTRAD se han cambiado los caracteres correspondientes a los códigos ASCII 161,163,171. Estos caracteres eran antes " ","£" y "±" mientras que ahora son Ñ, Pt y ñ respectivamente.

Sin embargo, en muchas impresoras el código asignado a la ñ es el 124 y a la Ñ el 92.

Para conseguir que la impresora saque la letra ñ tanto en mayúscula como en minúscula, emplearemos KEY DEF de la siguiente manera

```
KEY DEF 29,1,124,92
```

pues 92 es la tecla marcada con "ñ". El número 1 se coloca con el fin de obtener repetición en caso de mantener presionada la tecla. Y, como hemos visto antes, 124, 92 son los códigos ASCII de la ñ y Ñ respectivamente.

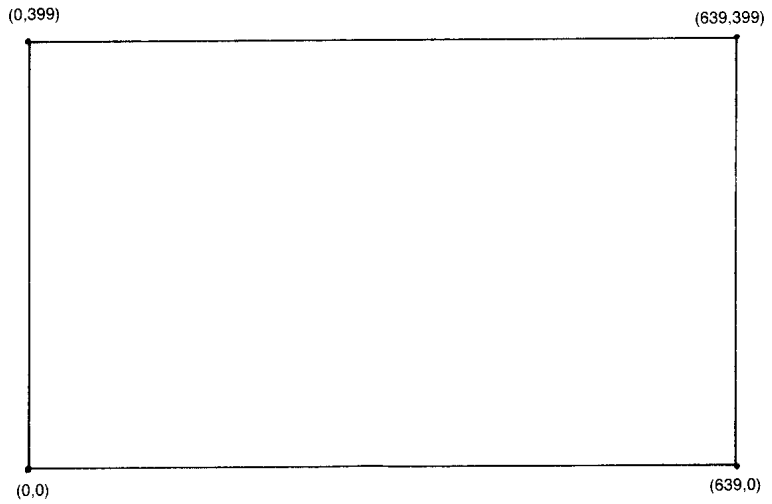
15

Gráficos

Cuando utilicemos las posibilidades gráficas de nuestro ordenador, en lugar de la pantalla de texto empleada hasta ahora, diremos que usamos la pantalla gráfica. Esta tiene como ventaja el que admite una mayor resolución ya que trabaja con puntos (pixels) que son más pequeños que los caracteres que usamos hasta ahora.

En la pantalla gráfica, el origen de coordenadas está situado en la esquina inferior izquierda de PAPER, a diferencia de la pantalla de texto en la que se encuentra en la esquina superior izquierda. Cada punto de la pantalla vendrá determinado por dos números que llamaremos coordenadas. La primera de ellas indica el desplazamiento horizontal respecto del origen y toma valores entre 0 y 639 ambos inclusive; la segunda mide el desplazamiento vertical respecto del origen y viene dada por un número entre 0 y 399 ambos inclusive.

Así, los cuatro vértices de la pantalla gráfica tendrán de coordenadas:



y el centro de la pantalla será, aproximadamente, el (319,199).

En la pantalla gráfica también tienen influencia los diferentes modos de trabajo utilizado. Según se trabaje en MODE 0, MODE 1 ó MODE 2 los puntos tendrán distintas anchuras, siendo el grosor (altura) siempre el mismo: 2 unidades. En MODE 0, MODE 1 y MODE 2, la anchura es respectivamente de 1,2 ó 4 unidades. Esto lo podemos expresar gráficamente así:

MODE 2 
MODE 1 
MODE 0 

Por todo lo dicho, podremos colocar horizontalmente 160, 320 ó 640 puntos (pixels) según se trabaje con MODE 0, MODE 1 ó MODE 2.

Verticalmente, cualquiera que sea el modo de trabajo siempre podremos colocar 200 puntos.

PLOT

Esta instrucción permite representar un punto en la pantalla. Su forma general es:

```
PLOT x,y
```

donde x e y indican las coordenadas del punto.

Se puede comprobar el funcionamiento de esta instrucción con:

```
10 MODE 0
20 PLOT 10,10:PLOT 630,10
30 PLOT 10,390:PLOT 630,390
40 PLOT 319,199
50 GOTO 50
```

Como ya hemos dicho anteriormente, en MODE 1, la anchura de cada punto es de dos unidades. Esto quiere decir que con PLOT 0,0 y con PLOT 0,1 no dibujaremos dos puntos distintos sino uno solo.

En MODE 0 ocurre algo parecido pero los puntos ahora tienen una anchura de 4 unidades y por lo tanto se dibuja el mismo punto con:

PLOT 0,0 PLOT 0,1 PLOT 0,2 y con PLOT 0,3

En MODE 2 esto no ocurre ya que, como se decía anteriormente, la anchura en este caso es de una unidad.

Para apreciar todo esto en los diferentes modos, se puede ejecutar el programa:

```
10 INPUT "MODO";M
20 MODE M
30 FOR X=0 TO 639 STEP 2^(2-M)
40 PLOT X,0
50 NEXT
```

con él se obtiene el eje horizontal inferior. Observar que se dibuja el mismo eje en cualquiera de los tres modos a pesar de que el salto del bucle (línea 30) es distinto según el modo empleado.

Por otra parte, para ver que la altura de cada punto es 2 se puede introducir el programa:

```
10 INPUT "MODO";M
20 MODE M
30 FOR X=0 TO 399
40 PLOT 0,X
50 NEXT
```

EJEMPLO

Un programa que representa una lluvia de estrellas puede ser el siguiente:

```
10 MODE 2
20 FOR X=1 TO 1000
30 PLOT RND*640,RND*400
40 NEXT
```

en donde los puntos van apareciendo de forma aleatoria en la pantalla.

EJEMPLO

Con PLOT se puede dibujar un cuadrado con el programa:

```
10 INPUT "LADO";L
15 MODE 2
20 FOR X=0 TO L-1
30 PLOT 0,X:PLOT X,0
40 PLOT L-1,X:PLOT X,L-1
50 NEXT
```

si también se quiere dibujar las diagonales se debe añadir:

```
45 PLOT X,X:PLOT L-X-1,X
```

PLOTR

Existe una instrucción que es una variación de PLOT. En muchos casos, una vez que hemos situado un punto en la pantalla, deseamos dibujar otro que esté desplazado un número determinado de pasos horizontalmente y verticalmente respecto del anterior. Lógicamente, con PLOT lo podemos hacer dando las coordenadas del nuevo punto; sin embargo, si conocemos los desplazamientos horizontal y vertical, podemos usar la instrucción:

```
PLOTR X,Y
```

Así, por ejemplo, si acabamos de marcar un punto en las coordenadas 138,325 y queremos colocar otro 121 pasos a la derecha y 15 hacia arriba, basta con:

```
PLOTR 121,15
```

Hay que observar que el mismo resultado se obtendría con:

```
PLOTR 259,340
```

Si la primera coordenada de PLOTR es negativa, significa que el desplazamiento es hacia la izquierda. En el caso de que la segunda coordenada sea negativa, esto quiere decir que el desplazamiento es hacia abajo.

ORIGIN

Ya hemos dicho que el origen de coordenadas para dibujar está situado inicialmente en la esquina inferior izquierda del papel. Pero este origen puede cambiarse con la instrucción:

```
ORIGIN X,Y
```

donde X e Y son las coordenadas del nuevo origen.

Por ejemplo, con:

```
ORIGIN 319,199
```

trasladamos el origen al centro de la pantalla.

Puede comprobarse el efecto de esta instrucción con el programa:

```
10 MODE 1
20 ORIGIN 319,199
30 PLOT 50,50:PLOT -50,50
40 PLOT 50,-50:PLOT -50,-50
```

NOTAS

1) Observar, que en este caso, tienen sentido las coordenadas negativas.

2) Al cambiar de modo, el origen vuelve a su posición inicial.

3) Además de los 2 parámetros obligatorios que lleva ORIGIN, se le pueden añadir otros cuatro para definir una ventana gráfica de la misma manera a como se definian las ventanas de texto.

CLG

Esta instrucción, borra la pantalla gráfica. Por ahora su misión es muy semejante a CLS; en el capítulo siguiente veremos que tiene más posibilidades.

De momento, se puede comprobar que el efecto de CLS y de CLG no es el mismo con ayuda del siguiente programa:

```
10 MODE 1
20 PLOT 275,150
30 CLS
40 PLOTR 75,50
```

que sitúa el segundo punto en (350,200)

Sin embargo, si cambiamos la línea 30 por

```
30 CLG
```

el punto marcado en pantalla estará en las coordenadas (75,50)

DRAW

Ya hemos visto que se pueden dibujar rectas punto a punto con PLOT, pero este procedimiento es pesado y lento. En BASIC disponemos de una instrucción que simplifica esta cuestión. Con:

DRAW X,Y

se dibuja una recta que une el punto en donde se encuentra actualmente el cursor gráfico con el punto de coordenadas X,Y.

Así, si queremos unir los puntos de coordenadas:

(450,75) y (300,225)

con una recta, lo podemos hacer de la siguiente forma:

```
10 PLOT 450,75
20 DRAW 300,225
```

con la línea 10 hemos colocado el cursor gráfico en el primer punto y con la línea 20, hemos unido este punto con el segundo.

Más adelante veremos que no es necesario marcar un punto en la pantalla para desplazar hasta allí el cursor gráfico.

Observar que si damos las coordenadas de un punto que está situado fuera de la pantalla, el ordenador no da mensaje de error y se limita a dibujar la parte de la recta que está dentro de la pantalla. Comprobarlo con:

```
10 CLG
20 DRAW 750,500
```

El siguiente ejercicio nos permitirá familiarizarnos con esta instrucción.

EJERCICIO

Vamos a dibujar el contorno de la pantalla gráfica:

```
10 MODE 2
20 DRAW 639,0:DRAW 639,399
30 DRAW 0,399:DRAW 0,0
```

DRAWR

De forma análoga a lo que sucedía con PLOT y PLOTR , DRAWR se diferencia de DRAW en que las coordenadas que se colocan detrás de DRAWR son relativas. Así

DRAWR X,Y

dibuja una recta que une el punto donde se encuentra el cursor gráfico y el punto que resulta de desplazar X unidades horizontalmente e Y unidades verticalmente.

Es decir, si el punto donde se encuentra el cursor tiene de coordenadas

(a,b)

DRAWR X,Y dibujará la recta que une los puntos (a,b) y (a+X,b+Y).

Con DRAWR el ejercicio anterior se resuelve de la siguiente forma:

```
10 MODE 2
20 DRAWR 639,0:DRAWR 0,399
30 DRAWR -639,0:DRAWR 0,-399
```

MOVE

Ya hemos comentado anteriormente que podemos desplazar el cursor gráfico sin necesidad de dibujar. Esto se consigue con:

MOVE X,Y

que coloca el cursor gráfico, aunque no se vea, en el punto de coordenadas (X,Y). Comprobarlo con:

```
10 MODE 2
20 DRAWR 85,100
30 MOVE 365,286
40 DRAWR 85,100
```

MOVER

Como ya podemos suponer, con:

MOVER X,Y

logramos desplazar el cursor gráfico, a partir de la posición actual, X unidades horizontales e Y unidades verticales.

XPOS, YPOS

Aunque podemos tener una idea de las coordenadas que ocupa el cursor gráfico, a veces es interesante conocerlas con exactitud. Para ello disponemos de dos instrucciones:

XPOS
YPOS

La primera de ellas nos da la coordenada horizontal y la segunda la coordenada vertical.

Puede verse en el siguiente programa:

```
5 MODE 2
10 PRINT XPOS,YPOS
20 MOVE 300,200
30 PRINT XPOS,YPOS
40 DRAWR -80,-125
50 PRINT XPOS,YPOS
```

CIRCUNFERENCIAS

Vamos a ver varios procedimientos para dibujar circunferencias ya que no poseemos en nuestro AMSTRAD una instrucción que las dibuje directamente.

Procedimiento 1

Dibujarlas punto a punto con la instrucción PLOT. Por ejemplo, vamos a dibujar una circunferencia centrada en la pantalla y de radio 100:

```
10 MODE 2
20 ORIGIN 319,199
30 DEG
40 FOR X=1 TO 360
50 PLOT 100*COS(X),100*SIN(X)
60 NEXT
```

en la línea 20 se sitúa el origen en el centro de la pantalla, mientras que con la línea 30 logramos que el ordenador trabaje en grados. El bucle determinado por las líneas 40, 50 y 60 permite representar 360 puntos que configuran la circunferencia. Introduciendo STEP 2 en la línea 40 obtenemos aproximadamente la misma gráfica pero de forma más rápida.

Procedimiento 2

Podemos considerar la circunferencia como un polígono de infinitos lados. En realidad, no hacen falta infinitos para conseguir el mismo efecto óptico. Para trazar los lados lo haremos con la instrucción DRAW.

```
10 MODE 2
20 ORIGIN 319,199
30 MOVE 100,0
40 DEG
50 FOR X=0 TO 360 STEP 10
60 DRAW 100*COS(X),100*SIN(X)
70 NEXT
```

en este caso hemos considerado la circunferencia como un polígono de 36 lados.

Este segundo procedimiento es bastante más rápido que el primero con la ventaja adicional de que la circunferencia resulta ser una curva continua. A la hora de trabajar con color se comprenderá la importancia de este hecho.

Podremos medir el tiempo empleado en realizar la circunferencia en cada uno de los casos añadiendo las líneas:

```
5 A=TIME
80 PRINT(TIME-A)/300
```

Si ahora queremos dibujar una circunferencia de centro y radio cualquiera, podremos hacerlo con el siguiente programa:

```
10 MODE 2:DEG
20 INPUT "COORDENADAS DEL CENTRO: X,Y";X,Y
30 INPUT "RADIO";R
40 MODE 2
50 ORIGIN X,Y:MOVE R,0
60 FOR Z=0 TO 360 STEP 10
70 DRAW R*COS(Z),R*SIN(Z)
80 NEXT
```

STEP 10 se ha elegido porque da buenos resultados cualquiera que sea el radio, pero se podría haber elegido otro número, o, incluso R/10.

TAG - TAGOFF

A la hora de trabajar con gráficos, es interesante tener alguna instrucción que permita escribir texto en la pantalla. En algunos casos ésto se puede hacer con PRINT. Ahora bien, con este método estamos utilizando una pantalla con 25 filas y 20, 40 ó 80 columnas según el modo elegido. Debido a que las gráficas cuentan con otro tipo de resolución, no siempre es posible colocar el cursor de texto en la posición que nosotros queramos. Lo ideal sería hacer coincidir el cursor de texto en la posición exacta del cursor gráfico.

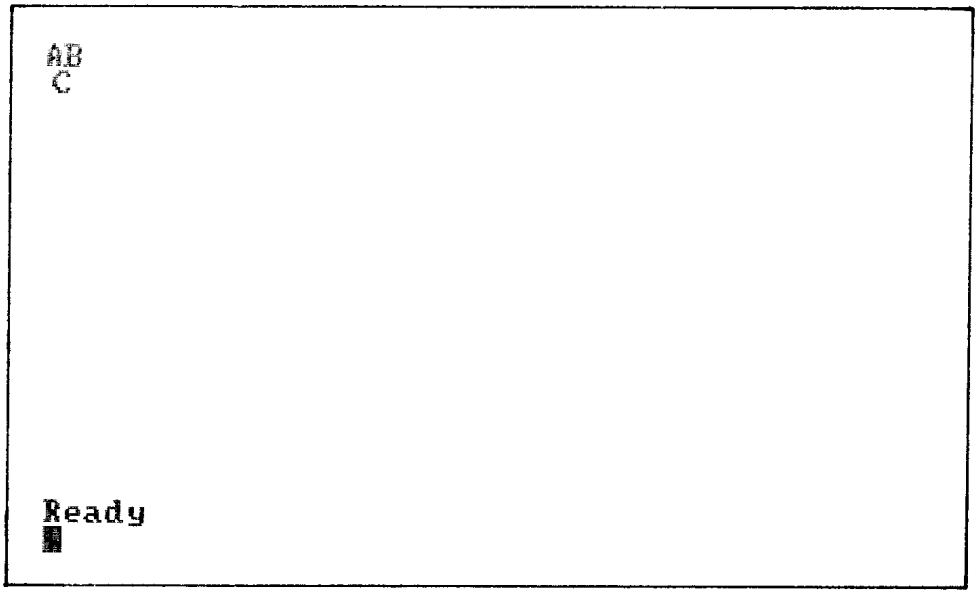
Esto se consigue con la instrucción:

```
TAG
```

para ver su efecto, introducir el siguiente programa:

```
10 MODE 1
20 PRINT"AB"
30 MOVE 8,383
40 TAG
50 PRINT"C";
60 LOCATE 1,24
```

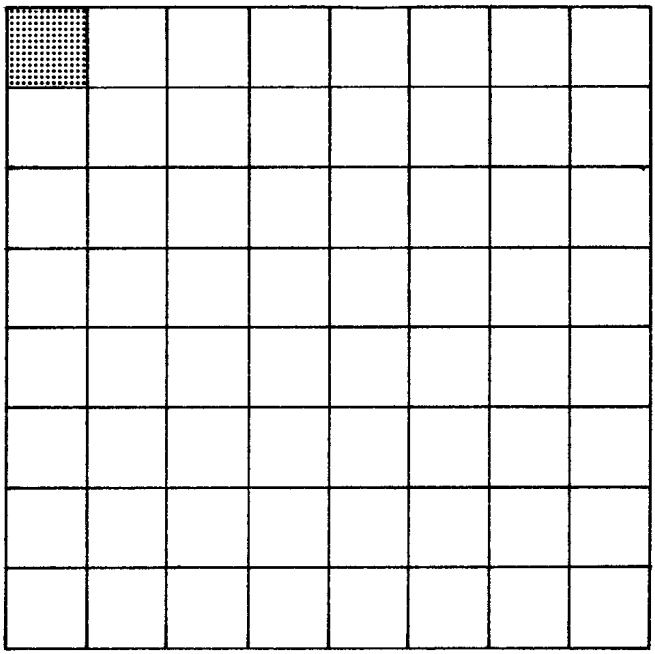
en la pantalla aparece:



En la línea 50 es necesario colocar el punto y coma al final para evitar que aparezcan ciertos símbolos molestos en la pantalla. La línea 60 se ha colocado con el fin de que el mensaje "READY" no borre la letra "C".

La posición de la letra "C" ha sido posible conseguirla gracias a TAG. Hay que darse cuenta que el espacio reservado para cada caracter de texto contiene 16x8, 16x16 ó 16x32 puntos (pixels) dependiendo del modo utilizado (2,1 ó 0).

Con la instrucción TAG, lo que hacemos es señalar el pixel que ocupa la posición superior izquierda del espacio reservado para el caracter.



Con TAG podremos conseguir que un determinado carácter se desplace por la pantalla con suavidad:

```
10 MODE 1
20 TAG
30 FOR X=399 TO 0 STEP -1
40 MOVE 319,X
50 PRINT CHR$(252);
60 NEXT
```

Todavía hay una instrucción, FRAME, que consigue un movimiento más suave aunque más lento. En algunos modelos de AMSTRAD esta instrucción no existe pero se puede simular con: CALL &BD19.

Si utilizamos PRINT sin TAG el programa anterior quedaría:

```
10 MODE 1
20 FOR X=1 TO 22
30 LOCATE 20,X:PRINT" "
40 LOCATE 20,X+1:PRINT CHR$(252)
50 FOR Y=1 TO 100:NEXT
60 NEXT
```

El efecto de TAG se anula con la instrucción:

TAGOFF

como se puede ver en este programa:

```
10 MODE 1
20 PRINT"ABCD";
30 MOVE 319,199
40 TAG:PRINT"E";
50 TAGOFF
60 PRINT"F"
```

GRAFICOS DEFINIDOS POR EL USUARIO

Existe la posibilidad de cambiar los caracteres que se obtienen en pantalla al pulsar una tecla. Esto puede ser interesante, pues podremos conseguir unos símbolos que no están en el teclado. Por ejemplo, podremos hacer que aparezca en la pantalla un pequeño círculo cuando presionemos una determinada tecla.

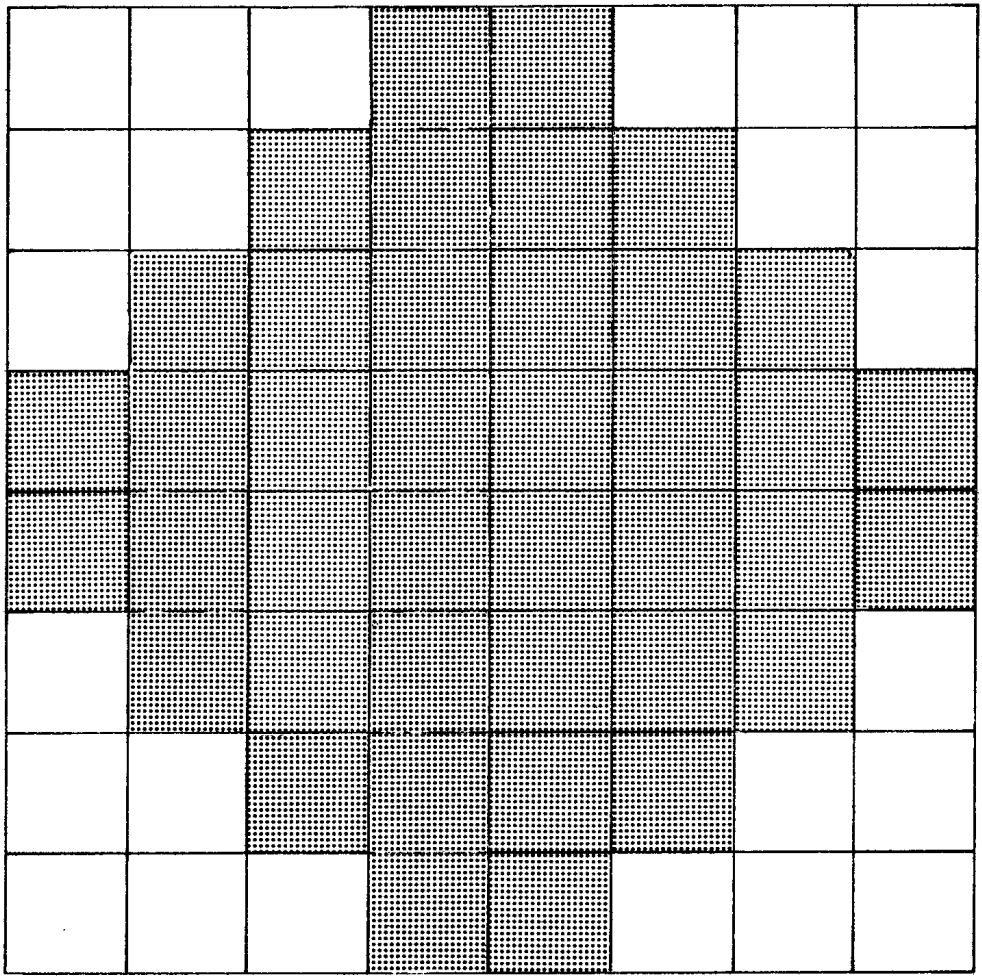
La redefinición se consigue con la instrucción:

SYMBOL

seguida de nueve números. El primero de ellos es el código ASCII del caracter a cambiar y que puede estar en el teclado o nó. Inicialmente sólo se pueden cambiar los caracteres cuyos códigos ASCII están comprendidos entre 240 Y 255 ambos inclusive. Los 8 números siguientes serán los encargados de diseñar el nuevo caracter.

Vamos a tratar de definir un pequeños círculo. Para ello se procede así:

Consideraremos que el espacio que va a ocupar este carácter gráfico es una cuadrícula 8x8. En ésta dibujaremos el círculo.



Los 8 números mencionados corresponden a cada una de las 8 filas tomadas de arriba abajo. Si en la primera fila se marcan los espacios en blanco con ceros y los oscuros con unos, obtendremos, en el sistema binario, el número:

00011000

que pasado a decimal resulta ser:

$$0.2^7 + 0.2^6 + 0.2^5 + 1.2^4 + 1.2^3 + 0.2^2 + 0.2^1 + 0.2^0 = 24$$

Y del mismo modo, procederemos con una de las demás filas.

2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0	
0	0	0	1	1	0	0	0	24
0	0	1	1	1	1	0	0	60
0	1	1	1	1	1	1	0	126
1	1	1	1	1	1	1	1	255
1	1	1	1	1	1	1	1	255
0	1	1	1	1	1	1	0	126
0	0	1	1	1	1	0	0	60
0	0	0	1	1	0	0	0	0

Con estos números la instrucción quedará:

```
SYMBOL 240,24,60,126,255,255,126,60,24
```

Para obtener el nuevo caracter en la pantalla, se emplea la instrucción:

```
PRINT CHR$(240)
```

Por este procedimiento sólo podemos definir 16 caracteres, los correspondientes a los códigos ASCII comprendidos entre 240 y 255. Sin embargo con:

```
SYMBOL AFTER b
```

se podrán redefinir desde los códigos ASCII b hasta el 255 ambos inclusive. El valor mínimo de b es 32, por lo que con:

```
SYMBOL AFTER 32
```

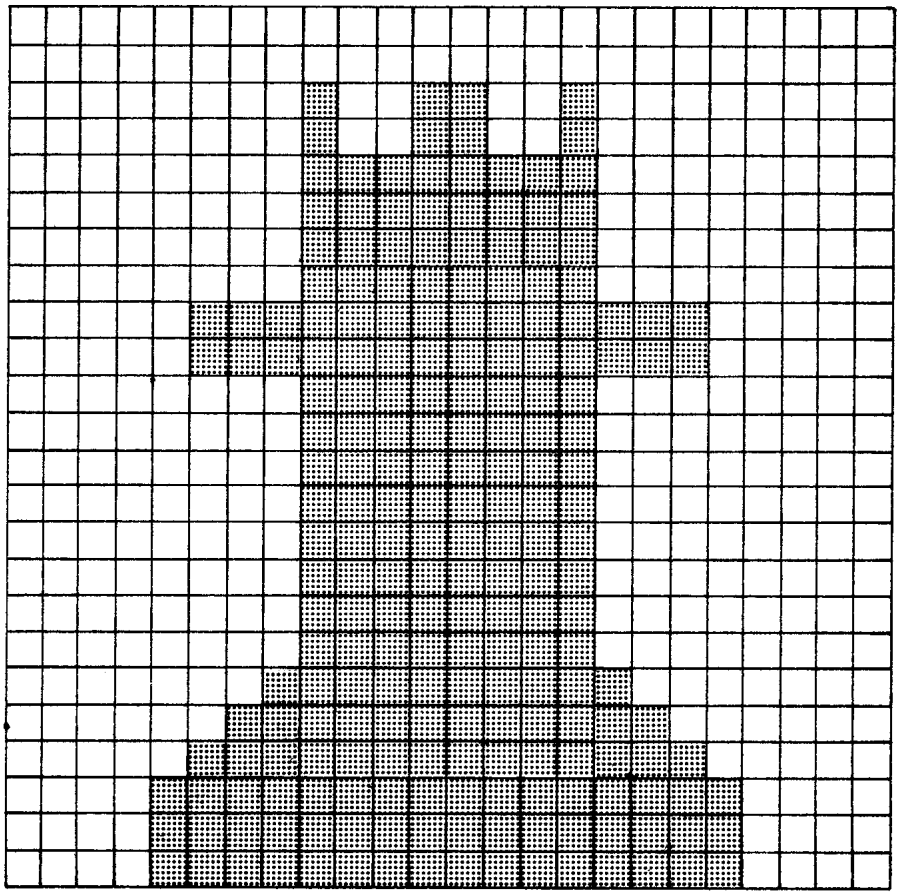
tendremos la posibilidad de definir el máximo número de caracteres.

En el caso de que el carácter cambiado esté en el teclado, bastará con presionar la tecla correspondiente para que aparezca en la pantalla. Así, con:

```
10 SYMBOL AFTER 97
20 SYMBOL 97,24,60,126,255,255,126,60,24
```

conseguiremos que, al presionar la tecla "a" minúscula, se imprima el símbolo definido anteriormente.

Hasta ahora hemos redefinido gráficos utilizando una cuadrícula 8x8. También podemos agrupar varias de ellas para un nuevo caracter de mayor tamaño. Veamos como ejemplo la definición, mediante 9 cuadrículas, de una reina de ajedrez.



```
10 SYMBOL 255,0,0,153,153,255,255,255,255
20 SYMBOL 254,7,7,0,0,0,0,0,0
30 SYMBOL 253,224,224,0,0,0,0,0,0
40 SYMBOL 252,0,0,1,3,7,15,15,15
50 SYMBOL 251,0,0,128,192,224,240,240,240
```

Para imprimir la dama:

```
PRINT CHR$(32);CHR$(255);CHR$(32)
PRINT CHR$(254);CHR$(143);CHR$(253)
PRINT CHR$(252);CHR$(143);CHR$(251)
```

PROGRAMAS PROPUESTOS

1) Dibujar la cuadrícula correspondiente a un tablero de ajedrez.
2) Completarla colocando letras y números.
3) Dibujar un sobre.
4) Dibujar un campo de tenis guardando las proporciones reglamentarias.
5) Si su ordenador no tiene la "ñ", definirla.
6) Dibujar la esfera de un reloj con las saetas señalando las 3 horas en punto.

16

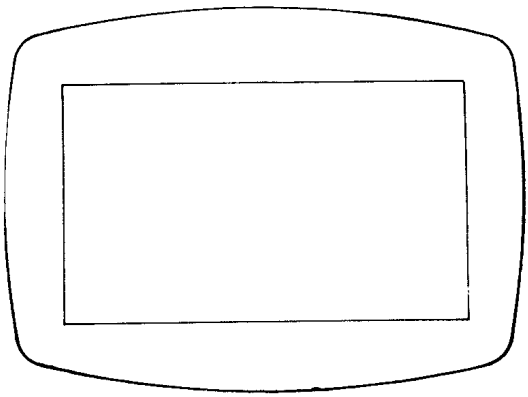
Color

En este capítulo pasamos revista a uno de los aspectos más llamativos de nuestro AMSTRAD. La gran cantidad de posibilidades que tiene este ordenador en cuanto a color hará posible mejorar la presentación de todos nuestros programas de una forma considerable.

Aunque con ciertas limitaciones, 27 son los colores que podremos utilizar. Cada color se identificará con un número comprendido entre 0 y 26, según la tabla:

0 NEGRO
1 AZUL
2 AZUL BRILLANTE
3 ROJO
4 MAGENTA
5 MALVA
6 ROJO BRILLANTE
7 PURPURA
8 MAGENTA BRILLANTE
9 VERDE
10 CYAN
11 AZUL CIELO
12 AMARILLO
13 BLANCO
14 AZUL PASTEL
15 NARANJA
16 ROSA
17 MAGENTA PASTEL
18 VERDE BRILLANTE
19 VERDE MARINO
20 CYAN BRILLANTE
21 VERDE LIMA
22 VERDE PASTEL
23 CYAN PASTEL
24 AMARILLO BRILLANTE
25 AMARILLO PASTEL
26 BLANCO BRILLANTE

Por otra parte, la pantalla del monitor tiene dos zonas claramente diferenciadas:



- PAPER (papel): zona rectangular central en donde podremos escribir y dibujar.
- BORDER (borde): el resto de la pantalla, y que sólo podremos colorear.

Al conectar el ordenador, estas dos zonas no son distinguibles ya que a las dos se les asigna un mismo color (AZUL, código 1).

Y ahora empezaremos por lo más sencillo:

BORDER

La forma general de esta instrucción es:

```
BORDER n
```

donde n indica el código del color elegido. Por lo tanto, n será un número comprendido entre 0 y 26.

Con esta instrucción le indicamos al ordenador el color que debe tener esa zona de la pantalla.

Así, BORDER 0, BORDER 14 y BORDER 26 colorearán el borde de la pantalla de negro, azul pastel o blanco brillante respectivamente.

Con el programa:

```
10 MODE 1
20 FOR X=0 TO 26
30 BORDER X
40 LOCATE 20,12
50 PRINT X
60 FOR Y=1 TO 1000:NEXT Y
70 NEXT
```

podremos observar, de uno a uno, todos los colores posibles con los que podemos colorear esa zona de la pantalla.

NOTAS

1) Incluso se le pueden dar a n valores fuera del intervalo 0–26 aunque sin conseguir nuevas coloraciones. De todos modos, si n es mayor que 31, aparecerá en pantalla el mensaje de error:

```
Improper argument
```

2) No tiene ninguna influencia en BORDER el modo en que se trabaja.

Todavía la instrucción BORDER, admite otra posibilidad:

```
BORDER m,n
```

donde m y n son los códigos correspondientes a dos colores. El resultado es colorear el borde de la pantalla intermitentemente con los dos colores m y n.

Así, con:

```
BORDER 15,23
```

conseguimos que el borde parpadee del color 15 (NARANJA) al color 23 (CYAN PASTEL)

También puede controlarse la velocidad con la que se intercambian estos colores con la instrucción:

SPEED INK

Que adopta la forma:

```
SPEED INK a,b
```

donde a y b marca el tiempo de permanencia de ambos colores medidos en 1/50 de segundo. Así,

```
SPEED INK 50,100
BORDER 15,23
```

hará que el color naranja permanezca durante un segundo y el cyan pastel durante dos.

Con:

```
BORDER 1,2
SPEED INK 10,100
```

colorearemos, intermitentemente, el borde con dos tonos de azul pero la permanencia del segundo es 10 veces superior a la de primero.

TINTEROS

Con BORDER se ha visto que se pueden utilizar hasta 27 colores diferentes. En las demás instrucciones relacionadas con el color, ésto no es posible. Es más, dependiendo del modo en que trabajemos, dispondremos de más o menos posibilidades.

Las diferentes tintas que se vayan a utilizar en cada modo, se guardarán en tinteros. El número de ellos está determinado por el modo de trabajo. Así:

MODO	NUMERO DE TINTEROS
0	16 (numerados del 0–15)
1	4 (numerados del 0–3)
2	2 (numerados del 0–1)

El que en cada modo se posea un número diferente de tinteros, está razonado por el hecho de que el ordenador emplea siempre 16 K de memoria para almacenar toda la información referida a la pantalla. Debido a ésto, cuantos más caracteres se puedan representar en ella, menos espacio libre queda para el color.

Inicialmente los diferentes tinteros están ocupados por un determinado color de tinta según las tablas siguientes:

MODE 0

NUMERO DE TINTERO	NUMERO DE TINTA
0	1 (AZUL)
1	24 (AMARILLO)
2	20 (CYAN BRILLANTE)
3	6 (ROJO BRILLANTE)
4	26 (BLANCO BRILLANTE)
5	0 (NEGRO)
6	2 (AZUL BRILLANTE)
7	8 (MAGENTA BRILLANTE)

8	10	(CYAN)
9	12	(AMARILLO)
10	14	(AZUL PASTEL)
11	16	(ROSA)
12	18	(VERDE BRILLANTE)
13	22	(VERDE PASTEL)
14	1,24	(AZUL Y AMARILLO BRILLANTE)
15	16,11	(ROSA Y AZUL CIELO)

los tinteros números 14 y 15, no están llenos con una sola tinta sino con dos, con lo que se consigue el efecto de "parpadeo" mencionado anteriormente.

MODE 1

NUMERO DE TINTERO	NUMERO DE TINTA	
0	1	(AZUL)
1	24	(AMARILLO BRILLANTE)
2	20	(CYAN BRILLANTE)
3	6	(ROJO BRILLANTE)

Observar que el color de la tinta contenida en estos cuatro tinteros es la misma que en los cuatro primeros tinteros del modo anterior.

MODE 2

NUMERO DE TINTERO	NUMERO DE TINTA	
0	1	(AZUL)
1	24	(AMARILLO BRILLANTE)

También aquí el color de la tinta de los dos tinteros es la misma que en los dos casos anteriores.

En realidad, el número de tinteros disponibles en cualquiera de los tres modos es siempre 16 pero en MODE 0 y en MODE 1 no contienen diferentes colores. Así, en MODE 1 los tinteros números: 4,5,6,7 están llenos de la misma tinta que los tinteros: 0,1,2,3, respectivamente.

En principio el tintero número 0 contiene el color del papel y el tintero 1 contiene el color de la tinta con la que escribiremos. Por esta razón, al conectar nuestro ordenador aparece siempre el fondo AZUL (código 1) y las letras en color AMARILLO (código 24).

La distribución inicial de los tinteros pueden alterarse con la instrucción:

INK

Con:

```
INK a,b
```

lograremos llenar el tintero número a con el color de tinta número b.

Supongamos que trabajamos en MODE 1, con

```
INK 0,5
```

se cambiará el color del tintero 0, llenándose de color MALVA (código número 5). Por ésto, veremos que el color del papel ha cambiado ya que, como se ha dicho anteriormente, el ordenador emplea la tinta del tintero 0 para colorear el papel.

Sin embargo, con

```
INK 3,7
```

llenamos el tintero número 3 con la tinta número 7 (PURPURA). Esto no tiene ningún efecto visible en la pantalla y lo único que hemos hecho es reservar esa tinta para su posterior utilización.

El programa:

```
10 MODE 1
20 FOR X=0 TO 26
30 FOR Y=0 TO 26
40 INK 0,Y:INK 1,X
50 FOR Z=1 TO 500:NEXT
60 LOCATE 12,12
70 PRINT"FONDO";Y;"TINTA";X
80 NEXT:NEXT
```

presenta las diferentes combinaciones que pueden hacerse con los códigos de la tinta y del fondo.

Otra forma de trabajar con INK es:

```
INK a,b,c
```

donde a es el número del tintero y b y c los códigos de dos colores de tinta. El efecto de esta instrucción es llenar el tintero número a con tinta que parpadea entre los colores b y c. Por ejemplo:

```
INK 1,0,26
```

hace que la tinta empleada para el texto parpadee entre los colores 0 (negro) y 26 (blanco).

Al igual que sucedía con BORDER m,n, aquí también podremos controlar la duración de cada color en el parpadeo con:

```
SPEED INK m,n
```

donde m es la duración del color b y n la del color c.

El siguiente programa:

```
10 INK 1,9,16
20 SPEED INK 40,20
30 BORDER 9,16
```

ayuda a comprender lo dicho anteriormente.

Una cuestión a tener en cuenta es que el efecto de todas estas instrucciones es PERMANENTE; esto es, duran hasta que instroduzcamos otra instrucción que las anule.

Según lo que se ha visto hasta ahora, al cambiar la tinta del tintero 1, se cambia el color de todo el texto escrito. Esto se debe a que, hasta ahora, solamente hemos utilizado ese tintero para escribir. Pero también existe la posibilidad de escribir con la tinta de otros tinteros.

Con la instrucción MODE, se restaura la composición inicial de los tinteros.

PEN (pluma)

La forma general de esta instrucción es

```
PEN n
```

donde n es el número de tintero con cuya tinta escribiremos.

Así, en MODE 1, con:

```
PEN 3
```

escribiremos en rojo.

Gracias a esta instrucción podemos escribir en la pantalla con diferentes colores de tinta según el número de tinteros disponibles. Así, en MODE 2, como solo disponemos de dos tinteros, únicamente existirán dos posibilida-

des: llenar nuestra pluma del tintero 0 o del tintero 1. En el primero de los casos, no se apreciará lo escrito ya que lo hacemos con tinta del mismo color que el fondo.

En los otros modos podremos simultanear diferentes colores de texto en la pantalla. En MODE 1, 3 colores y en MODE 0, 15.

Se pueden efectuar diferentes comprobaciones con:

```
10 INPUT "MODO";M
20 MODE M
30 INPUT "TINTERO";T
40 PEN T
50 GOTO 30
```

Si en una pantalla tenemos diferentes colores, al cambiar de color la tinta de un tintero, cambia todo lo escrito anteriormente con la tinta de dicho tintero. Esto es debido a que constantemente el ordenador consulta el color de la tinta de cada tintero y revisa todo lo escrito en la pantalla.

Hay otra manera de utilizar PEN en los modelos 664 y 6128:

```
PEN m,n
```

donde m, como antes, es el número de tintero y n puede tomar los valores 0 y 1 cuyo significado es:

```
0: OPACO
1: TRANSPARENTE
```

Así, con:

```
PEN 3,1
```

se puede escribir encima de otros caracteres ya impresos sin borrarlos. Comprobarlo con el programa:

```
10 MODE 0
20 PEN 1
30 LOCATE 1,6
40 PRINT"OOOOOO"
50 LOCATE 1,6
60 PEN 3,1
70 PRINT"//////"
```

PAPER

Su forma general es:

```
PAPER n
```

donde n es el número de un tintero. El efecto que tiene es el de colorear el fondo de los caracteres que se imprimen. Así, con:

```
PAPER 3
```

se observará que se colorea el fondo de todos los caracteres del color presente en ese momento en el tintero número 3. Si el tintero 3 mantiene su color primitivo, el fondo será rojo.

Si queremos que el efecto de PAPER afecte a todo el papel, será preciso teclear CLS. Así

```
10 MODE 1:PAPER 3:CLS
```

coloreará todo el papel de color rojo.

COLOR Y VENTANAS

Si en la pantalla tenemos definidas un conjunto de ventanas, con:

```
PAPER #a,b      y      PEN #a,b
```

podremos colorear el fondo y la tinta de los caracteres impresos en la ventana a.

Igual que sucedía anteriormente, si se quiere cambiar la totalidad del fondo de la ventana, será preciso introducir también la instrucción:

```
CLS #a
```

Como aplicación introducir el programa siguiente:

```
10 MODE 1
20 WINDOW#1,1,40,9,17
30 INK 0,6
40 PAPER#1,1:CLS#1
```

En la línea 20 se define una ventana rectangular centrada en la pantalla. Con la línea 30 coloreamos de rojo todo el papel, mientras que la línea 40 sirve para colorear de amarillo la ventana definida anteriormente.

Si queremos escribir dentro de una ventana con un color determinado, entonces emplearemos PEN.

```
10 MODE 1
20 WINDOW#1,1,40,1,12
30 PAPER#1,2:CLS#1
40 INK 1,0:PEN#1,3
50 PAPER#1,1:LOCATE#1,17,6
60 PRINT#1,"AMSTRAD"
70 GOTO 70
```

DIBUJO Y COLOR

Ahora, que estamos en condiciones de utilizar el color en nuestros dibujos, estudiaremos instrucciones ya vistas, pero a las que añadiremos un parámetro más referente al color.

CLG

Tiene de forma general:

```
CLG n
```

y su misión es la de borrar la pantalla gráfica coloreándola con la tinta del tintero n.

Así, con:

```
CLG 3
```

la pantalla se limpiará y quedará coloreada de rojo.

PLOT, PLOTR, DRAW, DRAWR, MOVE, MOVER

Todas las instrucciones han sido empleadas seguidas de dos parámetros. Ahora podemos emplear un tercero que tendrá como misión elegir el tintero de donde extraemos la tinta para dibujar.

Veamos un ejemplo:

dibuja una línea recta desde la posición (0,0) hasta la (150,200) de color rojo.

De la misma forma:

coloca el cursor gráfico en el punto (200,200) y a la vez elegimos el color de tinta del tintero 2.

ORDENES PARTICULARES

Ahora vamos a ver algunas instrucciones que no todos los modelos de AMSTRAD las poseen: GRAPHICS PAPER, GRAPHICS PEN, MASK y FILL.

GRAPHICS PAPER

En algunos modelos del ordenador AMSTRAD, como el 664 y el 6128, se distingue entre PAPER para texto y PAPER para gráficos. Así, podremos colorear la pantalla gráfica del color que deseemos con la instrucción GRAPHICS PAPER que adopta la forma:

```
GRAPHICS PAPER n
```

donde n es el número del tintero de donde tomamos la tinta.

Ahora podemos utilizar la instrucción CLG sin parámetro adicional. En ese caso, se borra la pantalla gráfica y se colorea el papel gráfico del color que le hayamos asignado con GRAPHICS PAPER.

El color del papel gráfico será visible en los siguientes casos:

1) Al borrar la pantalla con CLG.
2) Al utilizar la instrucción TAG, ya que entonces el fondo de los caracteres es el del papel gráfico.
3) Cuando dibujemos, con MASK, líneas discontinuas, como veremos más adelante.

El siguiente programa presenta los dos primeros casos reseñados anteriormente.

```
10 MODE 0
20 GRAPHICS PAPER 5
30 MOVE 200,200
40 TAG:PRINT"A";
50 IF INKEY$="" THEN 50
60 CLG
```

el primer caso se aprecia inmediatamente, mientras que para el segundo se debe presionar una tecla.

GRAPHICS PEN

Tiene de expresión general:

GRAPHICS PEN n

donde n es el número de un tintero. Utilizaremos esta instrucción siempre que deseemos dar color a las líneas gráficas o bien a los caracteres colocados en la pantalla mediante TAG. Comprobarlo con el programa:

```
10 MODE 0
20 GRAPHICS PEN 3
30 MOVE 0,200
40 DRAWR 639,0
50 MOVE 320,300
60 TAG
70 PRINT "AAAAAAA"
```

en el que la línea y las letras aparecerán en color rojo.

Otra manera de utilizar esta instrucción es:

GRAPHICS PEN a,b

donde el segundo parámetro puede tomar los valores:

0: Fondo opaco
1: Fondo transparente

Con el fondo transparente se logra imprimir encima de cualquier expresión sin borrarla previamente. Probarlo con el programa:

```
10 MODE 0
20 TAG:MOVE 200,200
30 GRAPHICS PEN 2
40 PRINT"TTTTTT";
50 MOVE 200,200
60 GRAPHICS PEN 11,1
70 PRINT"______";
```

con esta opción se podrá subrayar texto en la pantalla.

MASK

Como se comentó anteriormente, esta instrucción sirve para dibujar líneas discontinuas.

Su forma general es:

```
MASK n
```

donde n es un número comprendido entre 0 y 255. Para cada valor de n obtendremos una plantilla diferente que nos facilitará el trazado de dichas líneas. Precisamente con dicho valor lograremos que los intervalos de separación entre los diferentes tramos tengan la longitud deseada.

Para comprender el funcionamiento de esta instrucción, debemos pasar el número n a binario con lo que obtendremos un octeto formado por las cifras 0 y 1. Así, si n=89 se forma el octeto:

```
01011001
```

Con MASK 89, al dibujar una recta con DRAW, marcará cada punto con el color de papel o color de tinta según encuentre un 0 o un 1. Por esto, al dibujar los 8 primeros puntos de la recta, en nuestro ejemplo, aparecerá:

donde los cuadrados en negro indican puntos dibujados con el color de la tinta y los blancos con el color del papel.

Se puede comprobar el efecto de esta instrucción con el programa:

```
10 MODE 0
20 GRAPHICS PAPER 3
30 GRAPHICS PEN 2
40 LOCATE 1,1
50 INPUT N
55 CLS
60 MASK N
70 MOVE 0,200
80 DRAWR 639,0
90 GOTO 40
```

NOTA

También es posible introducir un cuarto parámetro, llamado "MODO DE TINTA", en las órdenes:

PLOT, PLOTR, DRAW, DRAWR, MOVE, MOVER

La finalidad de este cuarto parámetro consistirá en determinar las diferentes maneras de interaccionar el color de la tinta con el color del papel gráfico.

Puede tomar los valores:

0, 1, 2 y 3

que se corresponden con los efectos que llamaremos:

NORMAL, XOR, AND y OR

respectivamente.

Para ver su funcionamiento se puede introducir el programa que se dá a continuación. Cada vez que se presione una tecla obtendremos en la pantalla una demostración del efecto de este nuevo parámetro para los diferentes valores de T (color de la tinta), P (color del papel) y M (modo de tinta).

```
10 MODE 1
20 FOR M=0 TO 3
30 FOR P=0 TO 3
40 FOR T=0 TO 3
50 GRAPHICS PAPER P:CLG
60 LOCATE 1,1:PRINT"PAPEL"
70 LOCATE 18,1:PRINT"TINTA"
80 LOCATE 28,1:PRINT"MODO DE TINTA"
90 DRAWR 100,100,T,M
100 LOCATE 2,3:PRINT P
110 LOCATE 19,3:PRINT T
120 LOCATE 33,3:PRINT M
130 A$=INKEY$
140 IF A$="" THEN 130
150 NEXT:NEXT:NEXT
```

Se observará que según sea el valor del "modo de tinta" el color del dibujo varía según las siguientes tablas:

MODO DE TINTA = 0 (NORMAL)

TINTA \ PAPEL	0	1	2	3
0	0	0	0	0
1	1	1	1	1
2	2	2	2	2
3	3	3	3	3

MODO DE TINTA = 1 (XOR)

TINTA \ PAPEL	0	1	2	3
0	0	1	2	3
1	1	0	3	2
2	2	3	0	1
3	3	2	1	0

MODO DE TINTA = 2 (AND)

TINTA \ PAPEL	0	1	2	3
0	0	0	0	0
1	0	1	0	1
2	0	0	2	2
3	0	1	2	3

MODO DE TINTA = 3 (OR)

TINTA \ PAPEL	0	1	2	3
0	0	1	2	3
1	1	1	3	3
2	2	3	2	3
3	3	3	3	3

Por estar en MODE 1, se dispone de 4 tinteros: 0, 1, 2 y 3 cuyos colores iniciales son: azul, amarillo, cyan y rojo respectivamente.

En los casos en que el color del papel y el resultante coincidan, el dibujo no será visible.

Veamos con un ejemplo en cada caso, como se han obtenido estas tablas:

MODO DE TINTA 0 (NORMAL)

TINTA=2
PAPEL=3

RESULTANTE=2 puesto que en este caso el color de la tinta no se altera.

MODO DE TINTA 1 (XOR)

TINTA=2
PAPEL=3

RESULTANTE=1 Este número se obtiene expresando el valor de la tinta y el del papel en sistema binario y aplicando las tablas de verdad del capítulo 6 (XOR). Así:

TINTA=10
PAPEL=11
RESULTANTE=01

MODO DE TINTA 2 (AND)

TINTA=3
PAPEL=1
RESULTANTE=1 Procedimiento de la misma forma (AND).

TINTA=11
PAPEL=01
RESULTANTE=01

MODO DE TINTA 3 (OR)

TINTA=2
PAPEL=1
RESULTANTE=3 De forma análoga:

TINTA=10
PAPEL=01
RESULTANTE=11

FILL (Llenar)

Tiene de forma general:

FILL n

donde n es el número de tintero.

Con esta instrucción damos color a un recinto cerrado siempre y cuando el cursor gráfico esté situado dentro de él. Si dicho cursor estuviese situado en la frontera, no se produciría el efecto deseado.

En algunos casos, hay que tomar la precaución de no rellenar el recinto con el mismo color de la frontera, ya que en este caso no se podría volver a cambiar dicho color.

Para ver el funcionamiento de FILL, vamos a dibujar un cuadrado centrado en la pantalla y después lo colorearemos con el color del tintero que le indiquemos.

```
10 MODE 0
20 GRAPHICS PEN 2
30 ORIGIN 320,200
40 MOVE 50,-50
50 DRAWR 0,100:DRAWR -100,0
60 DRAWR 0,-100:DRAWR 100,0
70 INPUT T
80 MOVE 0,0
90 FILL T
100 GOTO 70
```

PROGRAMAS PROPUESTOS

1) Completar el dibujo del tablero de ajedrez, coloreándolo.
2) Dibujar una diana.
3) Hacer un programa que dibuje banderas con tres franjas horizontales de diversos colores (España, R.F. de Alemania,....).
4) Idem verticales (Italia, Bélgica.....)
5) Dibujar y colorear la bandera de la Cruz Roja.
6) Dibujar y colorear la placa de prohibido aparcar.
7) Hacer un programa que presente, ampliada, la plantilla que se emplea para dibujar con la instrucción MASK n. También se dibujará con esta plantilla el contorno de PAPER.

17
Sonido

Una de las cualidades realmente interesantes del ordenador AMSTRAD es su sonido. La facilidad con la que podemos programar diferentes tipos de sonido, hacen a este ordenador especialmente útil para componer cualquier tipo de música, obtener efectos especiales en juegos, etc.... .

Todas las cuestiones referentes al sonido se procesan en un circuito integrado (AY-3-8912). Debido a ésto, la programación del sonido goza de una cierta independencia.

Habitualmente, cuando hablamos de música, todas aquellas personas que no poseen conocimientos musicales se retraen. Sin embargo, hasta los que desconocen las más elementales normas musicales, pueden introducirse con su ordenador en el mundo musical. Solamente hace falta un poco de paciencia e interés.

Para comenzar hay que saber que todo sonido viene caracterizado por tres datos:

- Duración
  Es el tiempo durante el cual se emite el sonido.
- Amplitud
  Es el volumen de audición del sonido. Este volumen puede modificarse, o bien a lo largo del programa, o mediante el mando situado en la parte exterior del ordenador (su posición varía según el modelo)
- Tono
  Es el indicador de la gravedad o agudeza del sonido. Sabemos que el sonido se propaga mediante ondas y el tono depende del número de veces que la onda vibra en un segundo (frecuencia) de forma que, a mayor frecuencia le corresponde un sonido más agudo.

Nuestro ordenador maneja doce notas separadas por un semitono:

DO
DO#/ REb
RE
RE#/ MIb
MI
FA
FA#/ SOLb
SOL
SOL#/ FAb
LA
LA#/ SIb
SI

donde el símbolo # indica sostenido y b bemol.

Se estudiará ahora la instrucción fundamental.

SOUND

Esta orden es la que permite obtener todo tipo de sonidos con nuestro ordenador. Puede llevar asociado a ella hasta siete parámetros; aunque sólo los dos primeros son imprescindibles. Su forma general es:

```
SOUND C,T,D,V,EV,ET,R
```

Veamos detenidamente el significado de cada uno de estos parámetros

- C

Indica el canal que se va a utilizar. Nuestro ordenador posee tres canales lo que posibilita obtener diferentes efectos si las notas se emiten por uno u otro canal, o bien simultáneamente por varios canales a la vez. Además con C podemos realizar también dos operaciones que pueden ser muy interesantes: retener sonido y borrar la cola de sonido.

Todas estas operaciones se realizan según el número que asignemos a C, de acuerdo con la siguiente tabla

VALOR DE C	EFECTO
1	seleccionar canal A
2	seleccionar canal B
4	seleccionar canal C
8	sincronizar con canal A
16	sincronizar con canal B
32	sincronizar con canal C
64	retener sonido
128	borrar cola de sonido

Si queremos combinar diferentes efectos bastará con dar a C el valor que resulta de sumar los números correspondientes a los efectos combinados. Así, si queremos seleccionar el canal A, sincronizarlo con el C y borrar la cola de sonido, daremos a C el valor

```
1 + 32 + 128 = 161
```

Por esta razón, C puede tomar todos los valores comprendidos entre 1 y 255.

Las dos operaciones que pueden resultar más extrañas son las dos últimas. Pasamos a estudiarlas con más detalle.

1.- Retener sonido

Si introducimos en C el sumando correspondiente a una retención de sonido, la nota no sonará hasta que se libere con la instrucción RELEASE K, donde K será un número comprendido entre 1 y 7 para indicar el canal o canales donde anulamos la retención. Así

```
RELEASE 1  libera el canal A
RELEASE 2  libera el canal B
RELEASE 4  libera el canal C
RELEASE 3  libera los canales A y B
RELEASE 5  libera los canales A y C
RELEASE 6  libera los canales B y C
RELEASE 7  libera los canales A, B y C
```

Para comprender el efecto de la retención y liberación de sonido puede introducirse el siguiente programa:

```
10 SOUND 65,478,500,7
20 SOUND 2,253,500,7
```

En la primera línea se retiene el sonido del canal A y por tanto , sólo suena el sonido por el canal B definido en la línea 20.

Por tanto, si escribimos

```
15 RELEASE 1
```

liberamos el primer canal por lo que, al ejecutarse el programa, se escucharán las dos notas (la de la línea 10 por el canal A y la de la línea 20 por el canal B) simultáneamente aunque no sincronizadas.

Los tres últimos parámetros que se han introducido con la instrucción SOUND, se explicarán más adelante.

2.- Borrado de cola

El hecho de que todos los efectos musicales se procesen en un circuito integrado independiente, significa que el programa posiblemente se procesará antes de que acabe de emitirse el sonido. Esto puede comprobarse con el programa

```
10 SOUND 1,119,500,7
20 SOUND 2,106,500,7
30 SOUND 4,95,500,7
40 PRINT"YA SE HA REALIZADO EL PROGRAMA"
```

Comprobaremos que aparece el mensaje expresado en la línea 40 y todavía está sonando. Esto se debe a que el ordenador es capaz de almacenar en una cola hasta cinco sonidos que se emitirán uno detrás de otro si no hay orden de espera. Cuando hablamos de borrar cola de sonido nos estamos refiriendo a eliminar los sonidos que estaban en la lista de espera de ese canal.

Si introducimos el siguiente programa

```
10 SOUND 1,239,500,7
20 SOUND 2,190,500,7
30 SOUND 2,239,500,7
40 SOUND 2,113,500,7
50 SOUND 2,213,500,7
60 SOUND 2,113,500,7
```

observaremos que, durante los cinco primeros segundos, suenan dos notas a la vez por los canales A y B (líneas 10 y 20). A continuación, por el canal B, suenan las cuatro notas (cada una durante 5 segundos) correspondientes a las restantes líneas. Si sustituimos la línea 50 por

```
50 SOUND 130,213,500,7
```

se producirá el borrado de todos los sonidos almacenados en el canal B hasta llegar a la línea 50. Por eso, en este caso, solamente sonarán simultáneamente durante cinco segundos las notas correspondientes a las líneas 10 y 50 y posteriormente, la de la línea 60 que no ha sido borrada.

Si queremos saber si un canal posee cola, tenemos que recurrir a la instrucción SQ (K) donde K indica el canal que investigamos

K = 1 canal A
K = 2 canal B
K = 4 canal C

La condición SQ (K) > 127 indicará que en el canal marcado por K existe cola de sonido (sonidos que esperan emitirse). Podemos comprobarlo con el siguiente programa:

```
10 SOUND 1,956,500,7
20 SOUND 1,851,500,7
30 SOUND 1,758,500,7
40 FOR X=1 TO 5000:NEXT:PRINT SQ(1)
50 FOR X=1 TO 5000:NEXT:PRINT SQ(1)
60 FOR X=1 TO 5000:NEXT:PRINT SQ(1)
```

Con las líneas 10, 20 y 30 hemos formado una cola de sonido en el canal A. En la 40, 50 y 60 establecemos unos bucles de retardo con el fin de que, tras una duración aproximada al sonido emitido, se imprima SQ (1).

En pantalla podrán verse los números

```
131
132
4
```

indicándonos que existe cola de sonido en los dos primeros casos y no en el tercero.

- T

Con T se marcará la nota que queremos escuchar. Por ello, será necesario darle a T un valor comprendido entre 0 y 4095. Puede verse como varía la nota con

```
10 INPUT "VALOR DE T  (0<T<4095)";T
20 SOUND 1,T,500,7
30 WHILE SQ(1)>127 :WEND
40 GOTO 10
```

Con este programa podríamos encontrar los valores de T que corresponden a cada una de las notas conocidas. Hay una fórmula que soluciona este problema más fácilmente.

T = ROUND (125000/F)

donde F es un valor dado por

F = 440 * [2 ↑ (OCTAVA + (SEMITONO - 10)/12)]

donde OCTAVA viene dada por un número entero comprendido entre -3 y 4 ambos inclusive, y el SEMITONO es un número natural comprendido en-

tre 1 y 12 ambos inclusive. El número del SEMITONO marcará la nota que va a emitirse de acuerdo con la siguiente tabla

NOTA	SEMITONO
DO	1
DO#/ REb	2
RE	3
RE#/ MIb	4
MI	5
FA	6
FA#/ SOLb	7
SOL	8
SOL#/ LAb	9
LA	10
LA#/ SIb	11
SI	12

Por tanto, con nuestro ordenador somos capaces de obtener 8 octavas de 12 notas cada una. Así si queremos saber el valor de T correspondiente a la nota DO de la octava 0, procederemos del siguiente modo

F = 440 * [2 ↑ (0 + (1 -10)/12)]
F = 261.6255664
125000/F = 477.7820521
T = ROUND (125000/F) = 478

Podemos hacer un programa que imprima el valor de T correspondiente a cada octava y nota

```
5 MODE 1
10 FOR OCTAVA=-3 TO 4
20 PRINT"OCTAVA";OCTAVA:PRINT:PRINT
30 FOR X=1 TO 12
40 READ A$
50 F=440*(2^(OCTAVA+(X-10)/12))
60 T=ROUND (125000/F)
70 PRINT A$,T
80 NEXT
90 RESTORE
95 IF INKEY$="" THEN 95 ELSE CLS
100 NEXT
110 DATA DO,DO#,RE,RE#,MI,FA,FA#,SOL,SOL
#,LA,LA#,SI
```

Para pasar de una octava a otra pulsaremos una tecla cualquiera.

También podría pasarse de una octava a otra manteniendo el número de octava y aumentando el valor del semitono. De este modo, a un semitono de valor 13 le corresponde el DO de la octava siguiente.

Ya estamos en condiciones de que nuestro ordenador toque alguna cosa sencilla. Por ejemplo, podemos intentar que emita una escala. Lo hariamos así:

```
10 SOUND 1,119
20 SOUND 1,106
30 SOUND 1,95
40 SOUND 1,89
50 SOUND 1,80
60 SOUND 1,71
70 SOUND 1,63
80 SOUND 1,60
```

o bien

```
10 INPUT "NUMERO DE OCTAVA";OCTAVA
20 FOR Z=1 TO 8
30 READ X
40 F=440*(2^(OCTAVA+(X-10)/12))
50 T=ROUND (125000/F)
60 SOUND 1,T
70 NEXT
80 DATA 1,3,5,6,8,10,12,13
```

- D

Este parámetro mide el tiempo que durará el sonido en centésimas de segundo. Así, si D = 500, el sonido estará emitiéndose durante cinco segundos. Si en la instrucción SOUND no colocamos el valor de D, automáticamente el ordenador le asigna un valor igual a 20, por lo que la nota estará sonando durante 20 centésimas de segundo. Esto sucederá cuando a D le demos un valor positivo, pero D puede tomar también valores negativos. En realidad, D puede valer cualquier número comprendido entre -32768 y 32767. ¿Qué sucede cuando D = 0 o bien D<0? D = 0 significa que la duración del sonido está controlada por una envolvente de volumen (más adelante se explicará qué es una envolvente de volumen). D < 0 marca el número de veces que se repite el sonido de acuerdo con la envolvente de volumen.

Para ver el efecto de D en SOUND podemos realizar el siguiente programa:

```
10 INPUT "VALOR DE D";D
20 SOUND 1,16,D,7
30 WHILE SQ(1)>127:WEND
40 GOTO 10
```

- V

Este cuarto valor asociado a SOUND es el encargado de dar el volumen de la nota. Si D es un valor positivo, V estará comprendido entre 0 y 7 de tal modo que V = 7 indicará el volumen máximo. En caso de ser D = 0, entra en funcionamiento una envolvente de volumen y V no marca el volumen si no que hace referencia a la envolvente que va a utilizarse. Cuando estemos en esa situación, V puede tomar valores comprendidos entre 0 y 15.

Estos cuatro parámetro que hemos visto son suficientes para que nuestro ordenador suene ya aceptablemente. Veamos algún ejemplo de lo que puede hacerse.

Ejemplo 1

```
10 REM J.S.BACH
20 MODE 1
30 FOR I=1 TO 2
40 RESTORE 130
50 FOR X=1 TO 69
60 IF X=37 THEN RESTORE 130
70 IF X=53 THEN RESTORE 160
80 IF X=63 THEN RESTORE 150
90 READ A
100 SOUND 1,A,25-3*I,7
110 NEXT
120 NEXT
130 DATA 379,478,426,379,319,358,358,284
,319,319,239,253,239,319,379,478
140 DATA 426,379,358,319,284,319,358,379
,426,379,478,506,478
150 DATA 426,638,506,426,358,379,426
160 DATA 379,319,268,319,379,478,379,319
,284,358
```

Ejemplo 2

Ahora podemos asociar a algunas teclas de nuestro ordenador diferentes notas de tal modo que, al pulsarlas, se produzca el sonido. Lógicamente será necesario utilizar la instrucción KEY.

```
10 KEY 129,"SOUND 1,478,15"+CHR$(13)
20 KEY 130,"SOUND 1,426,15"+CHR$(13)
30 KEY 131,"SOUND 1,379,15"+CHR$(13)
40 KEY 132,"SOUND 1,358,15"+CHR$(13)
50 KEY 133,"SOUND 1,319,15"+CHR$(13)
60 KEY 134,"SOUND 1,284,15"+CHR$(13)
70 KEY 135,"SOUND 1,253,15"+CHR$(13)
88 SOUND 1,253,15
98 SOUND 1,253,15
```

Con este programa, al pulsar las teclas númericas f-1 hasta f-7, obtenemos respectivamente las notas DO, RE, MI, FA, SOL, LA y SI de la octava 0.

Este programa puede perfeccionarse y podemos llegar a tener con nuestro AMSTRAD un verdadero sintetizador.

- EV

Con la envolvente de volumen (EV) podemos modificar el volumen de una nota mientras está sonando.

Para manejar una envolvente de volumen será preciso definirla aparte mediante la instrucción ENV cuya forma general es:

ENV n,A_1,B_1,C_1,A_2,B_2,C_2,A_3,B_3,C_3,A_4,B_4,C_4,A_5,B_5,C_5

El valor de n indica el número de envolvente y es el que daremos a EV en la instrucción SOUND si queremos utilizarla. Evidentemente debe ejecutarse la orden ENV antes que la orden SOUND que la emplea. Puede haber hasta 15 envolventes de sonido y por tanto, EV puede tomar todos los números enteros comprendidos entre 0 y 15 ambos inclusive. EV = 0 indica que no se utiliza ninguna envolvente de volumen.

Podemos ver el efecto que causa ENV comparando el programa

```
10 ENV 1,5,1,50
20 SOUND 1,239,0,0,1
```

con

```
10 SOUND 1,239,500,7
```

Pasemos a estudiar con mayor detalle la instrucción ENV.

En primer lugar, hay que saber que, de los dieciseis parámetros que puede tener ENV, sólo son imprescindibles los cuatro primeros.

Ya hemos dicho que el primer parámetro indica el número de envolvente. Los quince restantes se agrupan en cinco familias de tres parámetros cada una

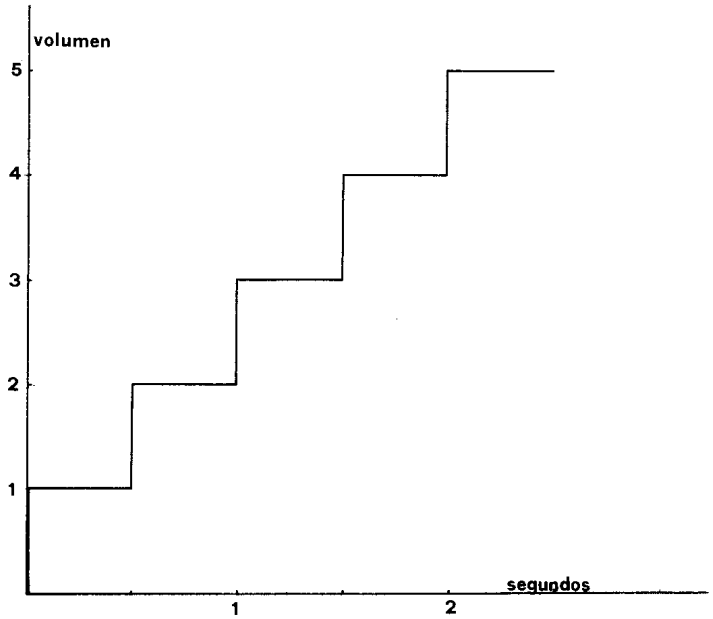
$A_i,B_i,C_i \quad i = 1,2,3,4,5$

Con estas cinco familias podemos repartir el tiempo que está sonando la nota en cinco secciones. Cada una de estas secciones viene determinada por un número de escalones (A_i), la altura de cada uno de los escalones (B_i) y la duración de cada escalón (C_i)

Así, si consideramos la envolvente del programa anterior

```
ENV 1,5,1,50
```

el control de volumen se hace mediante una sola sección cuya gráfica es



Los límites de A_i, B_i y C_i son:

$0 \leq A_i \leq 127$
$-128 \leq B_i \leq 127$
$0 \leq C_i \leq 255$

Los valores negativos de B_i dan lugar a escalones descendentes. La medida de C_i es en centésimas de segundo. Por tanto, la duración de la nota correspondiente a la envolvente anterior será de

5 x 50 = 250 centésimas de segundo = 2,5 segundos

NOTAS

1.– Recordemos que, si la duración D indicada en la instrucción SOUND es 0, ésta viene marcada por la envolvente de volumen.

2.– También puede darse al parámetro D otro distinto de 0. Para ese caso hay que tener en cuentra lo siguiente:

- Si D es menor que la duración marcada en la envolvente de volumen, el sonido sólo se emite durante el tiempo especificado por D.

```
10 ENV 1,10,1,50,25,2,40
20 SOUND 1,239,1000,1,1
```

Según la línea 10, el sonido debería durar

10 x 50 + 25 x 40 = 1500

lo cual equivale a 15 segundos. Sin embargo, sólo se emite durante 10 segundos que son los marcados por la orden SOUND.

- Si D es mayor que la duración marcada por la envolvente de volumen, después de realizada ésta, el sonido continúa emitiéndose el tiempo restante con el último volumen alcanzado en la envolvente. Podemos comprobarlo sustituyendo la línea 20 anterior por

```
20 SOUND 1,239,2000,1,1
```

3.– El volumen que alcanza la nota depende del número de escalones (A_i) y de la altura de éstos (B_i). Los valores que puede tener el volumen están comprendidos entre 0 y 15. Siempre que se sobrepase este valor, vuelve a repetirse de una forma cíclica. Es decir, un valor de volumen 16 equivale a 0, un 17 a 1 y así sucesivamente.

En la instrucción SOUND el parámetro correspondiente al volumen también puede tomar cualquier valor entre 0 y 15. Este valor marcará el volumen de comienzo. Podemos comprobarlo con

```
10 ENV 1,16,1,50
20 SOUND 1,239,800,0,1
```

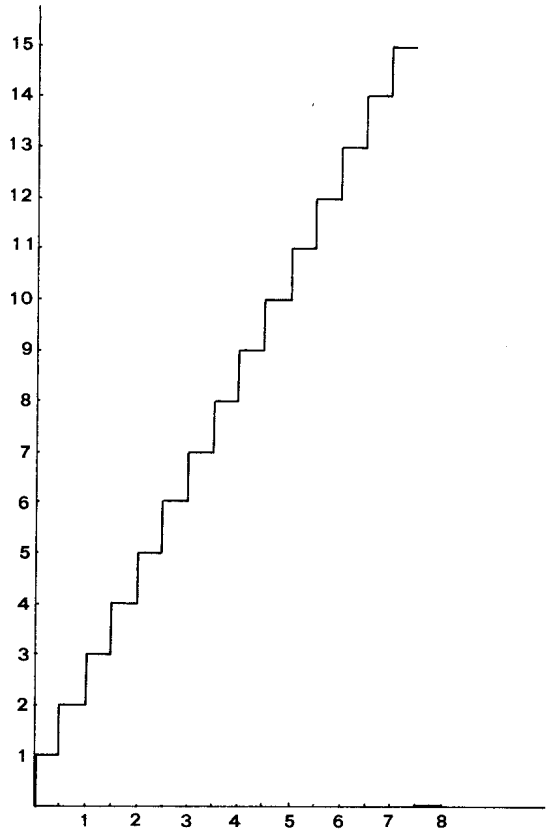
En este caso el volumen inicial es 0 y emitirá los 16 sonidos con el mismo tono y el volumen ascendente, finalizando en 16 (= 0).

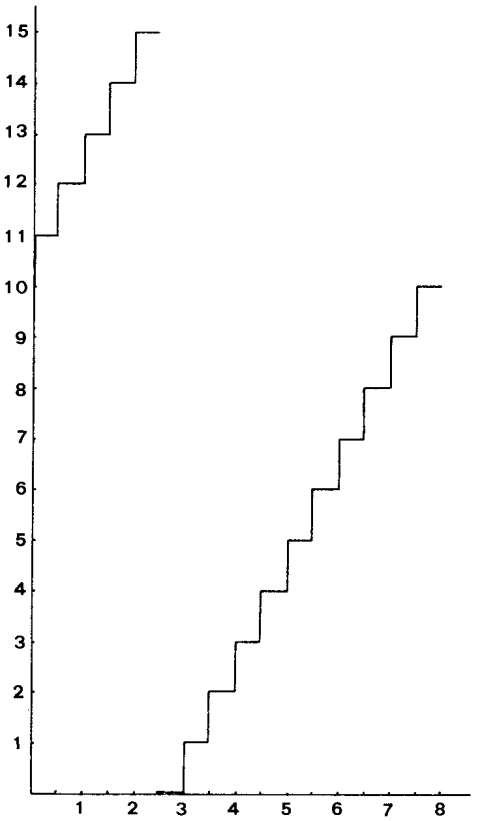
Si ahora sustituimos la línea 20 por

```
20 SOUND 1,239,800,10,1
```

el volumen de comienzo es 10 y a partir de éste aumenta de uno a uno. Cuan-

do llegue el 16 se emitirá con volumen 0 y luego continuará aumentando a partir de éste. Las dos gráficas correspondientes a cada situación serán:

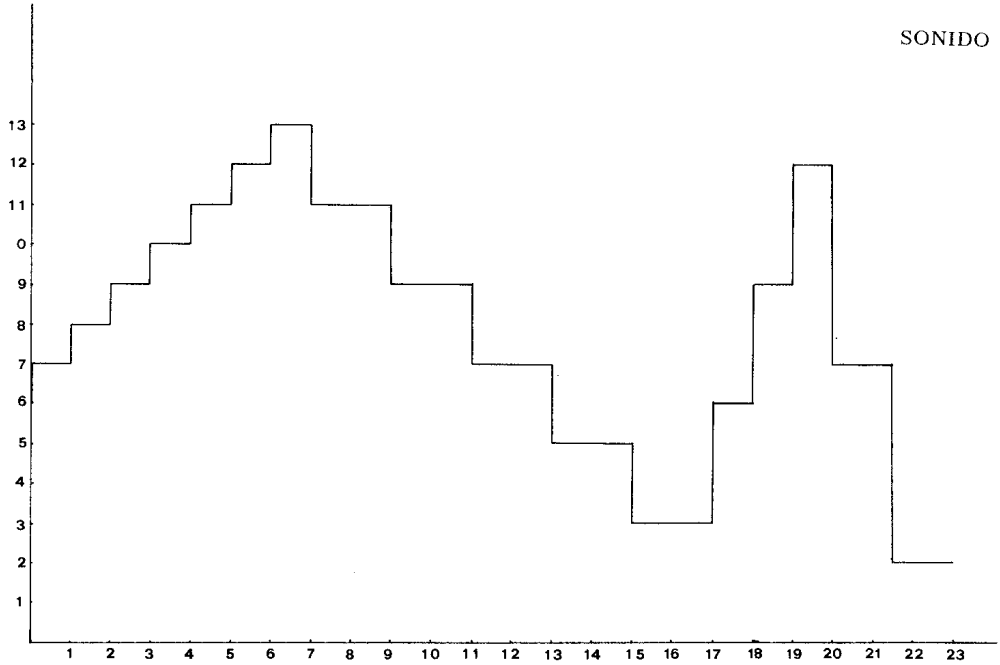




Si utilizamos más secciones comprobaremos mejor el efecto que produce la envolvente de volumen. Por ejemplo, con

```
10 ENV 1,7,1,100,5,-2,200,3,3,100,2,-5,150
20 SOUND 1,119,0,6,1
```

la gráfica de evolución del volumen a lo largo de su emisión será:



En este caso se han colocado en la envolvente cuatro secciones. La duración será:

10 x 10 + 20 x 5 + 5 x 5 + 15 x 10 = 375 centésimas = 3,75 segundos

Un programa que permitirá hacer pruebas con diferentes envolventes será el siguiente:

```
10 INPUT "CUANTAS SECCIONES ";S
20 FOR X=1 TO S
30 PRINT"SECCION";X
40 INPUT A(X)
50 INPUT B(X)
60 INPUT C(X)
70 NEXT
80 ENV 1,A(1),B(1),C(1),A(2),B(2),C(2),A
(3),B(3),C(3),A(4),B(4),C(4),A(5),B(5),C
(5)
90 SOUND 1,119,0,0,1
100 WHILE SQ(1)>127 :WEND
110 GOTO 10
```

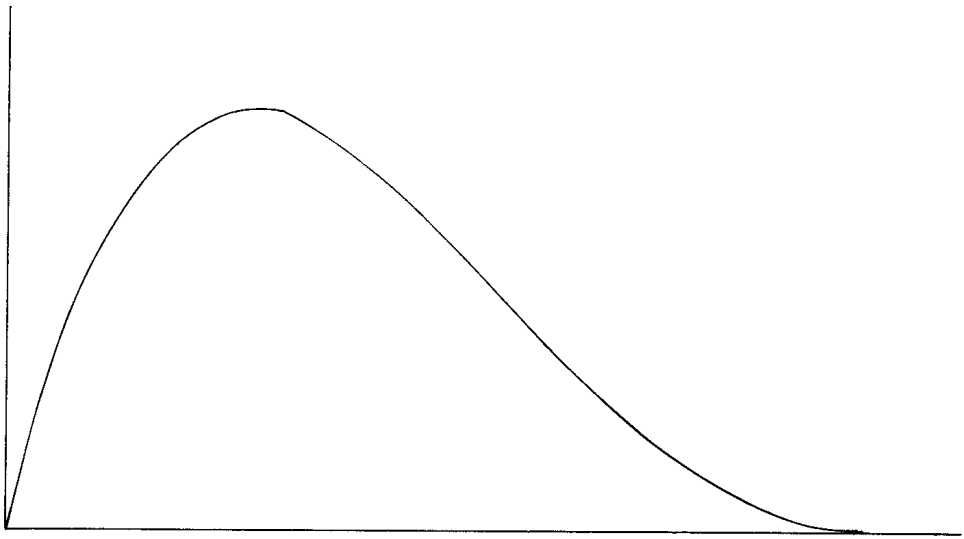
También podrían obtenerse los valores del tono, así como los diferentes valores de la envolvente, de una forma aleatoria. El siguiente programa pro-

duce una música que podriamos llamar caótica.

```
10 MODE 2
20 PRINT TAB(3);"TONO";TAB(22);"NUM. ESC
";TAB(44);"ALTURA";TAB(63);"DURACION"
30 RANDOMIZE TIME
40 TONO=INT(500*RND)
50 A=INT(20*RND)
60 B=INT(256*RND)-128
70 C=INT(256*RND)
80 PRINT TAB(3);TONO;TAB(25);A;TAB(45);B
;TAB(65);C
90 ENV 1,A,B,C
100 SOUND 1,TONO,0,0,1
110 WHILE SQ(1)>127:WEND
120 GOTO 30
```

NOTAS

1.- Con la envolvente de volumen podemos conseguir los sonidos que producen determinados instrumentos. Nunca serán iguales a los reales, pero podemos ir aproximándolos hasta un punto bastante convincente. Por ejemplo, una envolvente que semeja el sonido de un piano es de la forma



que puede conseguirse más o menos con tres secciones.

2.– Si queremos anular el efecto de la envolvente, basta con introducir la instrucción

ENV n

donde n es el número de la envolvente que queremos anular.

3.– Al definirse una envolvente de volumen se anula la definición anterior.

ET

La envolvente de tono (ET) funciona de manera análoga a la envolvente de volumen pero en este caso en vez de modificar el volumen mientras está sonando la nota, se modifica el tono.

También ET oscila entre 0 y 15 indicando el número de envolvente que consideraremos. La envolvente de tono se define con la instrucción ENT cuya forma general es:

ENT n,D_1,F_1,G_1,D_2,F_2,G_2,D_3,F_3,G_3,D_4,F_4,G_4,D_5,F_5,G_5

Ahora, el primer parámetro identifica la envolvente, pero además, en caso de poner un número negativo, se repetirá la envolvente mientras esté sonando la nota.

También existen hasta cinco secciones definidas cada una de ellas por D (número de escalones), F (altura de cada escalón) y G (duración de cada escalón). Los límites entre los que pueden moverse los parámetros D, F y G son los mismos que para A, B y C, con la salvedad de que el número de escalones (D) en el caso de la envolvente de tono, puede oscilar entre 0 y 239.

Para acabar de comprender la influencia de una envolvente de tono en la emisión de sonidos, pueden hacerse programas similares a los visto para la envolvente de volumen.

- R

Este último parámetro, cuyo valor puede estas entre 0 y 31 introduce ruido de una determinada frecuencia siempre que su valor sea distinto de 0. Precisamente R = 0 hace que la nota suene limpia. Este parámetro puede ser especialmente útil en muchos juegos dónde queremos conseguir un efecto sonoro efectivo.

Para ver su influencia podemos introducir en nuestro ordenador el siguiente programa:

```
10 INPUT "NUMERO DE RUIDO";R
20 SOUND 1,127,500,7,0,0,R
30 WHILE SQ(1)>127:WEND
40 GOTO 10
```

Con este último parámetro hemos terminado de explicar la instrucción SOUND. Ahora es el momento de afinar nuestro AMSTRAD, y con un poco de paciencia y habilidad, realizar diferentes pruebas para sacar el máximo partido a sus cualidades sonoras.

APENDICE A

Grabación y carga de programas

En este apéndice veremos los métodos y dispositivos que se emplean a la hora de almacenar programas para su posterior utilización.

La memoria del ordenador es de dos tipos: ROM y RAM. La primera viene grabada de fábrica, es de solo lectura y no se borra al cortar la alimentación. La segunda, memoria RAM, es de lectura y escritura, se usa para almacenar programas y datos, pero se borra al desconectar.

El ordenador perdería gran parte de su utilidad si cada vez que necesitásemos ejecutar un programa hubiera que teclearlo línea a línea. Esto es especialmente cierto en aquellos casos de programas con un número elevado de instrucciones. Por lo tanto, será necesario el empleo de dispositivos que nos permitan almacenar programas y datos de forma permanente: el cassette y la unidad de discos.

Una de las características de los ordenadores AMSTRAD es que siempre lleva, sea cual sea el modelo, uno de estos dispositivos incoroporado, el cassette en los modelos 464 y 472, la unidad de discos en los modelos 664 y 6128.

Todo esto no quiere decir que no se pueda usar en el modelo 6128 una cassette o en el modelo 472 una unidad de discos como sistema de almacenamiento alternativo.

Empezaremos describiendo el método a seguir para la utilización del cassette como sistema de almacenamiento.

UTILIZACION DEL CASSETTE

Teclear un programa de prueba y vamos a ver los diferentes pasos que hay que dar para guardarlo en cinta. La instrucción que emplearemos es:

SAVE (Salvar)

Cuya forma general es:

```
SAVE "TITULO"
```

pero que puede ir, como veremos más adelante, acompañada de otros parámetros. La misión de esta orden es guardar en cinta, con el nombre "TITULO" el programa que en ese momento está almacenado en la memoria del ordenador.

La expresión entrecomillada es el nombre que llevará el programa y que nos servirá para identificarlo posteriormente. Dicho nombre podrá tener 16 caracteres, incluidos los espacios en blanco, como máximo. Si pretendemos asignar un nombre con 17 caracteres o más, sólo admitirá los 16 primeros.

Para almacenar el programa de prueba, procederemos así:

Tecleamos:

```
SAVE "PRUEBA 1"  (e INTRO)
```

y aparecerá el mensaje:

```
Press REC and PLAY then any key
```

que nos indica que pongamos el magnetófono en grabación y luego presionemos una tecla cualquiera (menos ESC, MAY, FIJA MAY ó CONTROL).

En este momento aparecerá en la pantalla el nuevo mensaje:

```
Saving PRUEBA 1 block 1
```

y, según la longitud del programa, block 2, block 3, ... hasta acabar la grabación.

NOTAS

1) Podríamos haber grabado un programa sin asignarle nombre con:

```
SAVE""
```

y entonces el mensaje anterior sería:

```
Saving Unnamed file block 1
```

2) Empleando el caracter "!" antes del nombre, se consigue que no aparezca ninguno de los mensajes descritos anteriormente a excepción de "Ready"

3) Es posible prescindir de las últimas comillas. Con:

```
SAVE "PRUEBA 1
```

es suficiente.

Esta instrucción, como ya hemos advertido, puede ir acompañada por nuevos parámetros. Así:

```
SAVE "TITULO",A
```

almacena el programa en un fichero en formato ASCII.

Con:

```
SAVE "PRUEBA 1",P
```

se consigue guardar el programa "protegido" para que no pueda ser listado cuando se carge posteriormente y que, una vez ejecutado, se borre de la memoria automáticamente.

Una última manera de usar esta instrucción es:

```
SAVE "PRUEBA 1",B,m,n
```

donde la letra "B" viene de binario. Almacena, en cinta, el contenido de las n celdillas de la memoria del ordenador empezando a contar desde la celdilla m.

Como la imagen de la pantalla se guarda a partir de la celdilla 49152 y con una longitud de 16384 bytes, la instrucción:

```
SAVE "IMAGEN",B,49152,16384
```

graba en la cinta dicha imagen. Un byte está formado por 8 dígitos binarios y ocupa una celdilla de la memoria.

SPEED WRITE (Velocidad de escritura)

Tenemos la posibilidad de elegir entre dos velocidades de grabación en cinta. Con

```
SPEED WRITE n
```

se puede elegir la velocidad lenta (si n=0) o la velocidad rápida (si n=1). La primera es la que se supone elegida si no se indica lo contrario y es la mitad de la velocidad rápida.

También hay que añadir que con la velocidad rápida se consigue guardar en el mismo espacio más información, pero que la velocidad lenta es más segura.

LOAD

Su forma general es:

```
LOAD "TITULO"
```

y el efecto que tiene es el de buscar el programa de nombre "TITULO" y, que una vez encontrado, lo carga en memoria borrando el programa anterior, las variables y las funciones que se hubieran definido.

Con:

```
LOAD "PRUEBA 1"
```

nos aparecerá el mensaje:

```
Press PLAY then any key
```

que nos indica que pongamos en funcionamiento el cassette y que presionemos una tecla. Una vez que ha encontrado el programa PRUEBA 1 lo indica en la pantalla con:

```
LOADING PRUEBA 1 block 1
```

de forma similar a lo explicado con SAVE y finalizando con el mensaje "Ready".

Hasta llegar al programa buscado nos indicará los nombres de los programas que va encontrando con:

```
Found NOMBRE block n
```

NOTAS

1) Con:

```
LOAD "!PRUEBA"
```

se evita que aparezcan los mensajes que hemos mencionado.

2) La instrucción:

```
LOAD ""
```

carga el primer programa que encuentra.

3) Se pueden suprimir las segundas comillas. Así es suficiente con:

```
LOAD "PRUEBA 1
```

4) Una vez cargado el programa, para ejecutarlo se emplea RUN.

5) Si se ha grabado una pantalla con:

```
SAVE "IMAGEN",B,49152,16384
```

se cargará después con:

```
LOAD "IMAGEN" ó LOAD "!IMAGEN"
```

CAT (CATálogo)

Con esta instrucción se pueden conocer los títulos de los programas grabados en una cinta. Así, al teclar:

```
CAT
```

aparece el mensaje:

```
Press PLAY then any key
```

y se van imprimiendo los títulos de los programas así como el número de bloques de cada uno. También se indica el tipo de fichero empleando para ello los caracteres siguientes:

```
$ para un programa en BASIC
% idem protegido
* fichero en ASCII
& binario.
```

RUN

Su forma general es:

```
RUN "TITULO"
```

y su misión es cargar en la memoria y ejecutar el programa cuya denominación viene entre comillas.

NOTAS

1) Con:

```
RUN "!TITULO"
```

se consigue eliminar los mensajes en la pantalla.

2) RUN "" carga y ejecuta el primer programa que encuentra.

3) Los programas protegidos, que no se pueden cargar con LOAD, sí que se cargarán y ejecutarán con RUN.

4) Como en casos anteriores, se pueden suprimir las segundas comillas.

5) Presionando simultáneamente CONTROL e INTRO, se consigue RUN " para cargar y ejecutar el primer programa que se encuentre.

MERGE (Fundir)

Hasta ahora el cargar un programa, con LOAD o RUN, se borra el que hasta ese momento ocupaba la memoria del ordenador.

Con:

```
MERGE "TITULO"
```

se consigue fundir dos programas, uno presente en la memoria del ordenador y el otro cargado del cassette y de nombre TITULO.

Si en los dos programas existen líneas con igual numeración la línea correspondiente el programa cargado del cassette borrará la línea del programa presente en la memoria.

Por ejemplo, si tenemos:

```
10 A=15
20 B=20
30 PRINT A,B
```

y después cargamos, con MERGE, el programa:

```
10 A=30
20 C=50
30 PRINT A,B,C
```

obtendremos como resultado:

```
10 A=30
20 B=20
30 C=50
40 PRINT A,B,C
```

en el que las líneas 10 y 30 del primer programa han sido sustituidas por las líneas de igual numeración del segundo.

NOTAS

1) Con:

   ```
   MERGE "!TITULO"
   ```

   se suprimen los mensajes igual que en casos anteriores.

2) MERGE "" cargará el primer programa que se encuentre en la cinta.

3) Los programas protegidos no pueden ser cargados con esta instrucción.

4) Las variables y las funciones definidas se pierden con MERGE. También se olvida el efecto de DEFINT, DEFREAL y DEFSTR.

5) Los ficheros se cierran y además se hace un RESTORE.

CHAIN (encadenar)

Pueden presentarse de varias formas:

CHAIN "TITULO"

Carga en la memoria y ejecuta el programa "TITULO" pero sin borrar las variables que tuviesemos definidas anteriormente.

CHAIN "TITULO",n

Tiene el mismo efecto que en el caso anterior, pero ejecutándose el programa a partir de la línea n.

CHAIN MERGE "TITULO"

Tiene el mismo efecto que "MERGE" pero ejecutando el programa resultante sin olvidar las variables.

CHAIN MERGE "TITULO",n

Hace lo mismo que la anterior instrucción pero ejecutándose el programa a partir de la línea n.

CHAIN MERGE "TITULO",n,DELETE m-p

En este caso, además, se borra desde la línea m hasta la p del programa que inicialmente ocupaba la memoria del ordenador. Si se quiere que el programa.resultante se ejecute desde la primera línea, usaremos la instrucción:

```
CHAIN MERGE "TITULO",,DELETE m-p
```

NOTAS

1) Con:

```
CHAIN ""
```

se carga el primer programa que se encuentra en la cinta.

2) Con:

```
CHAIN "!TITULO"
```

se carga el programa "TITULO" pero sin que aparezcan los mensajes habituales en la pantalla.

3) Los programas protegidos no admiten MERGE.

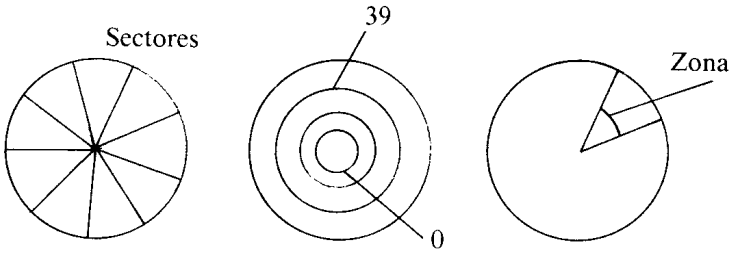
4) No se borran las variables, como ya hemos comentado, pero sí que se borran las funciones definidas y los ficheros abiertos. Se produce un RESTORE y se dejan sin efecto DEFINT-DEFREAL-DEFSTR.

UNIDAD DE DISCOS

Uno de los mayores atractivos, a nuestro juicio, de los modelos 664 y 6128 de AMSTRAD es el hecho de llevar incorporada una unidad de discos en lugar del cassette de los modelos anteriores.

El disco tiene indiscutibles ventajas respecto de la cinta en cuanto a rapidez, fiabilidad y capacidad.

El disco que se emplea es de 3 pulgadas, se puede usar por ambas caras y vá protegido por un envoltorio de plástico rígido que lo protege del polvo y de los golpes. Dicho disco se considera dividido en 9 sectores y en 40 pistas concéntricas que numeraremos del 0 al 39 de la parte exterior a la interior.



A la parte de una pista que se encuentra dentro de un sector, se le llama "zona". Cada pista tiene 9 zonas y una cara del disco tendrá:

40x9=360 zonas

en cada una de ellas se pueden almacenar hasta 512 bytes (grupos de 8 dígitos binarios), por lo que la capacidad de una cara es de:

512x360=184320 bytes

como cada 1024 bytes forman lo que se llama 1K de memoria, tendremos:

184320/1024=180 K

que es la capacidad total de una cara del disco.

Con el ordenador y su unidad de discos se suministra, además del lenguaje LOGO, el sistema operativo CP/M en su versión 2.2 para el modelo 664 y en las veriones 2.2 y PLUS en el modelo 6128.

Una operación que es obligatorio realizar antes de poder grabar en un disco es el FORMATEADO que tiene la misión de preparar el disco para ser usado. Esta utilidad se realizará cargando el sistema operativo CP/M siendo diferentes los pasos a dar si se trata del modelo 664 o del 6128.

Modelo 664

1) Conectar el ordenador.
2) Introducir el disco CP/M.
3) Teclar

```
|CPM
```

teniendo en cuentra que el caracter "ñ" se encuentra con la tecla inmediatamente a la derecha de la tecla "P".
Después de algunos instantes, aparecerá en la pantalla el mensaje

```
CP/M - Amstrad Consumer Electronics plc
A>
```

4) En este momento debemos escribir:

```
FORMAT + "ENTER"
```

con lo que habremos elegido la opción correspondiente al formateado de discos. A partir de este momento sólo hay que seguir las instrucciones que van apareciendo en la pantalla.

MODELO 6128

En este caso, y de una manera muy similar a la anterior, se procede así:

1) Conectar el ordenador.
2) Introducir el disco CP/M PLUS.
3) Teclear:

```
CPM
```

y aparecerá el mensaje:

```
CP/M Plus Amstrad Consumer Electronics plc
A>
```

4) Y ahora, tecleemos:

```
DISCKIT 3 + "INTRO"
```

y obtendremos un menú de opciones. Elegimos para nuestro fines la opción FORMAT que se obtiene pulsando la tecla "f4" del teclado numérico. Nos aparecerá otro menú para elegir el tipo de formato que deseemos; por el momento empezaremos con el formato de DATOS que se obtiene pulsando la tecla "f-6" del teclado numérico. A partir de ese momento, no hay más que seguir las instrucciones que el mismo programa facilita.

NOTAS

1) Al formatear un disco se borra toda la información grabada con anterioridad en él.

2) No se debe conectar o desconectar el ordenador con un disco en el interior ya que se corre el riesgo de que se borre la información grabada.

3) La operación de formateado debe realizarse en ambas caras.

4) El disco, al igual que la cinta, contiene un mecanismo mediante el cual se protege contra un borrado indeseado.

Ahora veremos las instrucciones de grabación y carga de programas en disco.

SAVE (Salvar)

Su forma general es:

```
SAVE "TITULO"
```

pero podrá ir acompañada de algunos parámetros que veremos más adelante.

El efecto de esta instrucción es el de almacenar en el disco el programa presente en la memoria del ordenador. El "TITULO" se refiere al nombre que asignaremos al programa y por el que después lo indentificaremos. Podrá tener una longitud máxima de 8 caracteres que pueden ser alfabéticos, numéricos y algunos, no todos, de los especiales. Por ejemplo, no se admiten espacios en blanco ni comas.

Si pretendemos asignar un nombre con más de 8 caracteres, o que contenga algún caracter no permitido, se emitirá el mensaje:

```
Bad command
```

Vamos a ver otras formas de usar esta instrucción:

SAVE "PRUEBA",P

Guarda el programa protegido de manera que no se podrá listar y se borrará una vez ejecutado.

SAVE "PRUEBA",A

Graba en ASCII, un fichero con nombre PRUEBA.

SAVE "PRUEBA",B,m,n,p

Graba en disco el contenido de las n celdillas de memoria a partir de la celdilla m. el parámetro p es el punto de entrada y se puede omitir.

Por ejemplo, con:

```
SAVE "IMAGEN",B,49152,16384
```

se graba en disco la imagen de la pantalla ya que ésta imagen se guarda a partir de la celdilla 49152 de la memoria y con una longitud de 16384 bytes (un byte está formado por 8 dígitos binarios y ocupa una celdilla de la memoria).

Al trabajar con cinta nada impide que sean grabados, uno detrás del otro, dos o más programas con el mismo nombre. En el disco esto es diferente, ya que sólo admite que se graben hasta dos programas con el mismo nombre pero automáticamente los distingue ya que el último se graba con la terminación:

```
BAS
```

y el anterior queda con la terminación

```
BAK
```

por lo que de, esta forma, se evita que por error se borre un programa al grabar otro con igual nombre en la misma cara del disco.

Por ejemplo, si se graba un programa con el nombre "PRUEBA" y después otro con el mismo nombre, el primero habrá quedado con la denominación:

```
PRUEBA.BAK
```

y el segundo con:

```
PRUEBA.BAS
```

CAT (CATálogo)

Con esta instrucción se saca por la pantalla un catálogo de los programas grabados en el disco, así como la longitud aproximada de cada uno de ellos.

En cada cara del disco, hay una zona reservada llamada directorio, para almacenar los nombres de los programas grabados en dicha cara.

Como máximo, se podrán guardar en este directorio un total de 64 títulos de programas.

MERGE (Fundir)

Su forma general es:

```
MERGE "TITULO"
```

y su función es la de cargar el programa "TITULO" mezclándolo con el programa que hasta entonces ocupaba la memoria del ordenador.

En el caso de que coincida la numeración de líneas en ambos programas, la línea procedente del disco borra la línea de igual numeración del programa que ocupaba la memoria inicialmente.

Los programas que se han grabado protegidos no se pueden cargar con esta instrucción.

LOAD (Cargar)

Su forma general es:

```
LOAD "TITULO"
```

y el efecto que tiene es el de cargar del disco a la memoria del programa "TITULO" borrando cualquier otro.

Si en la cara del disco con la que estamos trabajando no hay ningún programa con esa denominación, el ordenador nos dará el mensaje de error:

```
TITULO not found (TITULO no encontrado)
```

Por este método no se pueden cargar programas protegidos.

CHAIN

Puede adoptar una de estas formas:

CHAIN

Al cargar un programa con esta instrucción, no serán olvidadas las variables definidas con anterioridad. Puede adoptar una de las siguientes formas:

CHAIN "TITULO"

Carga un programa del disco a la memoria del ordenador borrando el que había hasta entonces en ella y ejecutándose desde el principio.

CHAIN "TITULO",n

Lo mismo pero ejecutándose el programa a partir de la línea n.

CHAIN MERGE "TITULO"

Carga el programa "TITULO" mezclándolo con el que había en la memoria, MERGE, y ejecutándose a partir de la primera línea.

CHAIN MERGE "TITULO",n

Tiene el mismo efecto que en el caso anterior pero el programa resultante se ejecuta a partir de la línea n.

CHAIN MERGE "TITULO",,DELETE a-b

Carga el programa "TITULO" mezclándolo con el presente en la memoria borrando de éste último desde la línea a hasta la b. El programa resultante se ejecuta a partir de la primera línea.

CHAIN MERGE "TITULO",n,DELETE a-b

Lo mismo que en el caso anterior pero ejecutándose a partir de la línea n.

Con CHAIN se pueden cargar y ejecutar los programas protegidos, pero con CHAIN MERGE ésto no es posible.

RUN

Su forma general es:

```
RUN "TITULO"
```

carga y ejecuta el programa grabado en disco con el nombre "TITULO" a la vez que borra el programa anteriormente grabado en la memoria.

Los programas protegidos pueden ser ejecutados por esta orden.

USO DE SISTEMAS ALTERNATIVOS DE ALMACENAMIENTO

Como ya hemos mencionado anteriormente, el hecho de que todos los modelos de AMSTRAD lleven incorporado su propio dispositivo de grabación de programas, no significa que no se pueda emplear otro distinto.

Así, la instrucción:

```
|DISC
```

logra que se trabaje con la unidad de disco en los modelos que llevan el cassette incorporado.

Con:

```
|TAPE
```

se trabajará con cinta en los modelos que llevan incorporada la unidad de disco.

Es posible grabar con un dispositivo y cargar en el otro. Así:

```
|TAPE.IN
|DISC.OUT
```

hace que se lea por el cassette y se grabe en disco.

Y con:

```
|DISC.IN
|TAPE.OUT
```

se consigue leer en disco y grabar en cinta.

APENDICE B

Print Using

PRINT USING

Supongamos que queremos presentar en pantalla una tabla ARTICULOS-PRECIOS de una tienda de electrodomésticos. Con el siguiente programa:

```
10 MODE 1
20 PRINT"ARTICULO";TAB(16);"PRECIO"
30 PRINT
40 FOR X=1 TO 10
50 READ A$,B
60 PRINT A$;TAB(15);B
70 NEXT
80 LOCATE 1,24
90 DATA TELEVISOR,110000,ASPIRADOR,12500
,BATIDORA,3500,CAFETERA,5600,PLANCHA,325
0,VIDEO,120000,ORDENADOR,98000,TOCADISCO
S,38000,FILMADOR,225000,DISCO,900
```

obtenemos en pantalla:

```
ARTICULO        PRECIO

TELEVISOR       110000
ASPIRADOR       12500
BATIDORA        3500
CAFETERA        5600
PLANCHA         3250
VIDEO           120000
ORDENADOR       98000
TOCADISCOS      38000
FILMADOR        225000
DISCO           900
```

Sin embargo, esta presentación no es la habitual en este tipo de tablas, ya que, normalmente, se ajusta el precio por la derecha. Es decir, lo normal es presentar la tabla, en la forma,

```
ARTICULO          PRECIO

TELEVISOR         110000
ASPIRADOR          12500
BATIDORA            3500
CAFETERA            5600
PLANCHA             3250
VIDEO             120000
ORDENADOR          98000
TOCADISCOS         38000
FILMADOR          225000
DISCO                900
```

Esto puede conseguirse, si consideramos los números como cadenas. De este modo, se obtiene la tabla con la presentación anterior, sustituyendo la línea 60, por:

```
55 B$=MID$(STR$(B),2)
60 PRINT A$;TAB(22-LEN(B$));B$
```

Este último procedimiento puede llegar a resultar complicado, por lo que nuestro ordenador nos ofrece una forma mucho más cómoda de conseguir lo mismo. La instrucción USING, que debe ir siempre con PRINT, será la encargada de ello;

```
PRINT USING "# # # # # #";K
```

reservará seis espacios para la impresión de K. Si K ocupa menos de los seis espacios reservados, la impresión se completa con espacios en blanco, a la izquierda de K.

En general, si tras USING hay n símbolos # se reservan n espacios para la impresión de la expresión señalada posteriormente, ajustandose ésta siempre por la derecha. Por eso, el último carácter de la expresión a imprimir coincidirá con el lugar reservado por USING situado más a la derecha.

Así, si introducimos en el ordenador:

```
10 PRINT 28
20 PRINT USING "####";28
30 PRINT USING "##";28
```

aparece

```
 28
  28
28
```

El primer número ocupa 3 espacios, puesto que reserva el lugar situado más a la izquierda para el signo. Como USING en el primer caso ha reservado 4 espacios y el número ocupa 2, coloca 2 espacios en blanco antes de escribir 28. Finalmente, si sólo se reservan 2 espacios se imprime el número 28 en ellos, sin reservar ninguno para su posible signo.

Volviendo a nuestro ejemplo de la tabla, ARTICULOS-PRECIOS, podemos imprimirla tal como queríamos, con el programa:

```
10 MODE 1
20 PRINT"ARTICULO";TAB(16);"PRECIO"
30 PRINT
40 FOR X=1 TO 10
50 READ A$,B
60 PRINT A$;TAB(16);USING"######";B
70 NEXT
80 LOCATE 1,24
90 DATA TELEVISOR,110000,ASPIRADOR,12500
,BATIDORA,3500,CAFETERA,5600,PLANCHA,325
0,VIDEO,120000,ORDENADOR,98000,TOCADISCO
S,38000,FILMADOR,225000,DISCO ,900
```

En la línea 60, al aparecer antes de USING, TAB(16), los seis espacios que se reserva se consideran a partir de la columna número 16.

NOTA

- Si el número de caracteres a imprimir, supera el número de espacios reservados por USING, se imprimen todos precedidos del símbolo % para indicarnos el error cometido. Así, si en la línea 60 en lugar de reservar 6 espacios, reservamos 5, obtenemos:

```
ARTICULO        PRECIO

TELEVISOR       %110000
ASPIRADOR       12500
```

```
BATIDORA          3500
CAFETERA          5600
PLANCHA           3250
VIDEO           %120000
ORDENADOR         98000
TOCADISCOS        38000
FILMADOR        %225000
DISCO               900
```

Esta función USING admite muchas otras posibilidades de impresión. Estudiémoslas detenidamente:

NUMEROS DECIMALES

Cuando trabajamos con tablas de números decimales, se nos plantean dos cuestiones:

- El punto decimal debe colocarse en todos los números en la misma columna de la tabla.
- Será conveniente presentar todos los números con el mismo número de decimales. Esto llevará consigo la necesidad de redondearlos y de añadir ceros en aquellos que no los tengan.

Ambas cuestiones se solucionan con USING del siguiente modo:

Tras USING se colocan tantos símbolos # como espacios se quieran reservar para la parte entera. A continuación se escribe el punto decimal y tantos símbolos # como cifras decimales deseemos.

Con el siguiente ejemplo:

```
10 MODE 1
20 FOR X=1 TO 3
30 READ A
40 PRINT USING "###.###";A
50 NEXT
60 DATA 123.4567,567,45.1
```

conseguimos imprimir 3 números decimales de tal forma, que el punto decimal se coloque en la cuarta columna, y todos se consideren con tres decimales:

```
123.457
567.000
 45.100
```

Una aplicación interesante, puede ser el obtener una tabla, que nos de el capital acumulado a lo largo de años sucesivos, si está colocado a interés compuesto y a un rédito dado.

Un programa es:

```
10 MODE 1
20 INPUT "TANTO POR CIENTO";R
30 PRINT:PRINT
40 R=R/100
50 PRINT"AÑOS        CAPITAL"
60 PRINT
70 FOR X=1 TO 15
80 Y=80000*(1+R)^X
90 PRINT X;TAB(10);USING"#######.##";Y
100 NEXT
110 LOCATE 1,24
```

En él, partimos de un capital inicial de 80.000 pesetas. El rédito se introduce desde el teclado y en pantalla aparecerán los capitales con dos decimales que se obtendrían año a año hasta un total de 15.

```
TANTO POR CIEN? 7
AÑOS          CAPITAL
 1             85600.00
 2             91592.00
 3             98003.44
 4            104863.68
 5            112204.14
 6            120058.43
 7            128462.52
 8            137454.89
 9            147076.74
10            157372.11
11            168388.16
12            180175.33
13            192787.60
14            206282.73
15            220722.52
```

Cuando utilizamos números bastante elevados, es costumbre separar sus cifras enteras de derecha a izquierda en grupos de 3. En la notación inglesa, esta separación se efectúa mediante comas; al contrario que en la muestra que se hace en puntos. Esto puede conseguirse de forma análoga al caso anterior, sin más que añadir una coma antes del punto:

PRINT USING"# # # # ,. # #"

donde el número de símbolos # antes y después de ,. indica el número de cifras enteras y decimales.

Hay que tener en cuenta, que al colocar las comas separadoras, éstas también ocupan espacios reservados por USING. Por eso, el número de símbolos # antes de la coma debe ser como mínimo igual al número de cifras enteras más el número de comas menos uno.

Por ejemplo con:

```
10 PRINT USING "##########,.##";123456789
20 PRINT USING "######,.##";123456.789
```

obtenemos:

```
123,456,789.00
123,456.79
```

NOTA

- Tanto en todo lo escrito anteriormente, como en lo siguiente, podemos definir la expresión entrecomillada que sigue a USING, como una variable alfanumérica, lo que clarificará nuestro programa. Así, se obtendría el mismo efecto que antes con:

```
10 A$="##########,.##"
20 B$="######,.##"
30 PRINT USING A$;123456789
40 PRINT USING B$;123456.789
```

SIGNOS

Si introducimos:

```
10 PRINT USING"#####";1234
20 PRINT USING"#####";-1234
30 PRINT USING"####";1234
40 PRINT USING"####";-1234
```

obtenemos:

```
 1234
-1234
1234
%-1234
```

se observará que, cuando el número es positivo, no se imprime su signo, mientras que si el número es negativo, el signo - se imprime y ocupa lugar. Por esta razón, el último número aparece precedido del símbolo %.

Si deseamos que aparezcan ambos signos, hay que colocar antes del primer # el símbolo +. Por ejemplo, con:

```
10 PRINT USING"+#####";1234
20 PRINT USING"+#####";-1234
30 PRINT USING"+####";1234
40 PRINT USING"+####";-1234
```

conseguimos:

```
 +1234
 -1234
+1234
-1234
```

observar que, al poner el signo + se reserva una posición más.

Si para alguna cuestión muy particular, deseamos que los signos aparezcan a la derecha del número, colocaremos el signo + tras el último símbolo #. Así con,

```
10 PRINT USING"#####+";1234
20 PRINT USING"#####+";-1234
30 PRINT USING"####+";1234
40 PRINT USING"####+";-1234
```

se obtiene:

```
 1234+
 1234-
1234+
1234-
```

si sólo interesara consignar a la derecha el signo - (como sucede en contabilidad con los números rojos), se colocará el signo - tras el último símbolo #.

Por ejemplo con:

```
10 PRINT USING"#####-";1234
20 PRINT USING"#####-";-1234
30 PRINT USING"####-";1234
40 PRINT USING"####-";-1234
```

se imprimirá:

```
 1234
 1234-
1234
1234-
```

NOTACION EXPONENCIAL

Si queremos que los números aparezcan con notación exponencial, escribiremos los signos ↑↑↑↑ después del último # y antes de los signos + ó -, siempre que los hubiera.

Con el siguiente programa:

```
10 PRINT USING"######.##^^^^";1234.567
20 PRINT USING"#####.##^^^^";1234.567
30 PRINT USING"####.##^^^^";1234.567
40 PRINT USING"###.##^^^^";1234.567
50 PRINT USING"##.##^^^^";1234.567
60 PRINT USING"#.##^^^^";1234.567
```

se obtiene:

```
12345.67E-01
1234.57E+00
123.46E+01
12.35E+02
1.23E+03
.12E+04
```

Se puede observar que el exponente depende del número de cifras enteras, que viene determinado por el número de símbolos # antes del punto, menos unos.

CARACTERES ESPECIALES

Con el fin de mejorar la presentación de una lista de números, podemos rellenar los espacios en blanco de la izquierda, con asteriscos. Esto, se consigue colocando, en la expresión asociada a USING, dos asteriscos antes del primer símbolo #.

Puede comprobarse con:

```
10 MODE 1
20 FOR X=1 TO 5
30 READ I
40 PRINT TAB (10);USING "*******.##"; I
50 NEXT
60 DATA 387.153,12,1.8,10352.23,0.03
```

obteniéndose:

```
****387.15
*****12.00
******1.80
**10352.23
******0.03
```

En muchos trabajos de contabilidad se acostumbra a colocar antes de la cantidad el símbolo de la moneda con que se trabaja. Nuestro ordenador posee dos símbolos:

$ (dólares)
£ (libras)

Además el modelo castellano, admite un tercer símbolo:

Pt (pesetas)

Para colocar cualquiera de estos símbolos delante de la cantidad a imprimir, hay que duplicarlo antes del primer símbolo #. Por ejemplo con:

```
5 MODE 1
10 PRINT USING "$$####.##";125.3
20 PRINT USING "££###.##";125.3
30 PRINT USING "PtPt#####.##";125.3
```

se obtiene:

```
 $125.30
£125.30
  Pt125.30
```

CADENAS

También PRINT USING puede utilizarse con cadenas. En este caso, utilizaremos en lugar del símbolo #, espacios en blanco delimitados por el símbolo \.

Con:

```
PRINT USING " \ n espacios \ ";A$
```

logramos que se impriman los n+2 primeros caracteres de A$.

Si A$ tiene menos caracteres que n+2, se completa la impresión con espacios en blanco a su derecha.

También pueden colocarse en un mismo PRINT USING varias cadenas afectando la orden a cada una de ellas.

El siguiente programa ilustra lo anterior:

```
10 A$="MI":B$="AMSTRAD"
20 PRINT USING "\\";A$;B$
30 PRINT USING "\ \";A$;B$
40 PRINT USING "\   \";A$;B$
50 PRINT USING "\    \";A$;B$
60 PRINT USING "\     \";A$;B$
70 PRINT USING "\      \";A$;B$
80 PRINT USING "\       \";A$;B$
```

Como podemos comprender fácilmente, el mínimo número de caracteres que pueden imprimirse es dos, (que corresponden a los símbolos \\). Si se desea que solamente aparezca el primer carácter, debemos hacerlo con la orden:

```
PRINT USING "!";a$
```

Así con:

```
PRINT USING "!";"MI";"AMSTRAD"
```

se obtiene:

```
MA
```

Veamos una última aplicación de USING. Según se explicó en el capítulo

5, si ordenamos imprimir una cadena a partir de una columna y su longitud es mayor que los espacios disponibles en esta fila, se imprime en su totalidad, en la siguiente. Por ejemplo, si ordenamos:

```
10 MODE 1
20 LOCATE 38,4
30 PRINT"AMSTRAD"
```

observaremos que la palabra AMSTRAD se imprime al principio de la quinta fila. Para conseguir que comience la impresión de AMSTRAD en la fila 4, columna 38, tras PRINT hay que colocar USING "&";

Por ejemplo:

```
10 MODE 1
20 LOCATE 38,4
30 PRINT USING"&";"AMSTRAD"
```

Soluciones de los ejercicios propuestos

Capítulo 5

1

```
10 MODE 0
20 LOCATE 6,12
30 PRINT"AMSTRAD"
40 LOCATE 1,24
```

2

```
10 INPUT "MODO";M
20 MODE M
30 A=7*2^M
40 LOCATE A,9
50 PRINT"******"
60 PRINT TAB(A);"*     *"
70 PRINT TAB(A);"*     *"
80 PRINT TAB(A);"*     *"
90 PRINT TAB(A);"*     *"
100 PRINT TAB(A);"******"
110 LOCATE 1,24
```

3

```
10 INPUT "MODO";M
20 MODE M
30 A=7*2^M
40 LOCATE A,9
50 PRINT"******"
60 PRINT TAB(A);"**  **"
70 PRINT TAB(A);"* ** *"
80 PRINT TAB(A);"* ** *"
90 PRINT TAB(A);"**  **"
100 PRINT TAB(A);"******"
110 LOCATE 1,24
```

4

```
10 MODE 1
20 LOCATE 19,7:PRINT"*"
30 LOCATE 18,8:PRINT"***"
40 LOCATE 17,9:PRINT"*****"
50 LOCATE 16,10:PRINT"*******"
60 LOCATE 15,11:PRINT"*********"
70 LOCATE 14,12:PRINT"***********"
80 LOCATE 13,13:PRINT"*************"
```

```
90 LOCATE 12,14:PRINT"***************"
100 LOCATE 18,15:PRINT"* *"
110 LOCATE 18,16:PRINT"* *"
120 LOCATE 18,17: PRINT"* *"
130 LOCATE 18,18: PRINT"* *"
140 LOCATE 18,19: PRINT"***"
150 LOCATE 1,24
```

5

```
10 MODE 1
20 WINDOW 1,10,1,25
30 WINDOW #1,11,20,1,25
40 WINDOW #2,21,40,1,8
50 WINDOW #3,21,40,9,16
60 WINDOW #4,21,40,17,25
70 LOCATE 5,12:PRINT 0
80 LOCATE #1,5,12:PRINT#1,1
90 LOCATE #2,10,4:PRINT#2,2
100 LOCATE #3,10,4:PRINT#3,3
110 LOCATE #4,10,4:PRINT#4,4
120 LOCATE 1,23
```

Capítulo 6

1

```
10 MODE 1
20 INPUT "COEFICIENTE DE X^2";A
30 INPUT "COEFICIENTE DE X";B
40 INPUT "TERMINO INDEPENDIENTE";C
50 D=B*B-4*A*C
60 PRINT
```

2

```
10 MODE 0
20 INPUT "NUMERO";N
30 PRINT
40 C=C+1
50 PRINT N;"x";C;"=";N*C
60 IF C<9 THEN 30
70 GOTO 70
```

```
70 IF D<0 THEN PRINT "NO HAY RAICES REAL
ES":END
80 IF D=0 THEN PRINT "RAIZ DOBLE =";-B/2
/A:END
90 PRINT "LAS DOS RAICES SON"
100 PRINT (-B+D^0.5)/(2*A),(-B-D^0.5)/(2
*A)
```

3

```
10 MODE 1
20 INPUT"QUE FIGURA";N
30 ON N GOTO 40,130,250
40 C=C+1
50 LOCATE C,1:PRINT "*"
60 LOCATE C,20:PRINT "*"
70 IF C<20 THEN LOCATE 1,C:PRINT "*"
80 IF C<20 THEN LOCATE 30,C:PRINT "*"
90 IF C<15 THEN LOCATE 4,C+3:PRINT "*":L
OCATE 27,C+3:PRINT "*"
100 IF C<25 THEN LOCATE C+3,4:PRINT "*":
LOCATE C+3,17:PRINT "*"
110 IF C<30 THEN 40
120 END
130 REM HORIZONTALES
140 C=C+1
150 D=0
160 D=D+1:LOCATE C,4*D-3:PRINT "*":IF D<
6 THEN 160
170 IF C<25 THEN 140
180 REM VERTICALES
190 C=0
200 C=C+1:D=0
210 D=D+4:LOCATE D-3,C:PRINT "*":IF D<25
 THEN 210
220 IF C<20 THEN 200
230 LOCATE 1,24
240 END
250 C=C+1
260 IF C<20 THEN LOCATE 1,C:PRINT "*":LO
CATE 21-C,C:PRINT "*":LOCATE 31-C,C:PRIN
T "*"
270 IF C<20 THEN LOCATE 30,C:PRINT "*":L
OCATE C,C:PRINT "*":LOCATE 31-C,21-C:PRI
NT "*"
280 LOCATE C,1:PRINT "*"
290 LOCATE C,20:PRINT "*"
300 IF C<30 THEN 250
```

4

```
10 MODE 1
20 INPUT "PRIMER NUMERO";A
30 MAYOR=A:MENOR=A
40 C=1
50 CLS
60 INPUT "OTRO NUMERO  ";B
70 C=C+1
80 IF MAYOR <B THEN MAYOR = B
90 IF MENOR >B THEN MENOR =B
100 IF C=10 THEN CLS:LOCATE 10,12:PRINT
"MAYOR:";MAYOR:LOCATE 10,15:PRINT "MENOR
:";MENOR:LOCATE 1,24: END ELSE GOTO 50
```

5

```
10 MODE 1
20 C=C+1
30 IF C=13 THEN END
40 LOCATE 1,2*C-1
50 INPUT "NUMERO DEL MES";N:IF N<1 OR N>
12 THEN 40
60 LOCATE 25,2*C-1
70 ON N GOSUB 90,100,110,120,130,140,150
,160,170,180,190,200
80 GOTO 20
90 PRINT "ENERO":RETURN
100 PRINT"FEBRERO":RETURN
110 PRINT"MARZO":RETURN
120 PRINT"ABRIL":RETURN
130 PRINT"MAYO":RETURN
140 PRINT"JUNIO":RETURN
150 PRINT"JULIO":RETURN
160 PRINT"AGOSTO":RETURN
170 PRINT"SEPTIEMBRE":RETURN
180 PRINT"OCTUBRE":RETURN
190 PRINT"NOVIEMBRE":RETURN
200 PRINT"DICIEMBRE":RETURN
```

6

```
10 MODE 1
20 INPUT "CUAL ES LA BASE";A
30 PRINT
40 INPUT "CUAL ES EL EXPONENTE";N
50 S=0
60 P=1
70 PRINT
80 I=1
```

```
90 GOSUB 140
100 P=S: I=I+1: IF I<=N THEN 90
110 PRINT A;"^";N;"=";P
120 PRINT: PRINT
130 GOTO 20
140 S=0
150 J=1
160 S=S+P: J=J+1: IF J<A+1 THEN 160
170 RETURN
```

Capítulo 7

1

```
10 INPUT "MODO";M
20 MODE M
30 FOR X=1 TO 20*2^M
40 PRINT "*";
50 NEXT
60 FOR X=2 TO 24
70 LOCATE 1,X
72 PRINT"*";SPC(20*2^M-2);"*"
80 NEXT
85 LOCATE 1,25
90 FOR X=1 TO 20*2^M
100 PRINT"*";
110 NEXT
120 GOTO 120
```

2

```
10 MODE 1
20 FOR X=1 TO 8
30 LOCATE 20-X,6+X
40 FOR Y=1 TO 2*X-1
50 PRINT"*";
60 NEXT
70 NEXT
80 FOR X=1 TO 4
90 LOCATE 18,14+X
100 PRINT"* *"
110 NEXT
120 LOCATE 18,19:PRINT"***"
130 LOCATE 1,24
```

3

```
10 MODE 1
20 INPUT "elegir el metodo: 1- FOR-NEXT
                            2- WHILE-WEN
D ",J
30 IF j<>1 AND j<>2 THEN GOTO 20
40 CLS
50 INPUT "cual es el numero";n
60 PRINT
70 PRINT n;"! = ";
80 ON j GOSUB 110,170
90 PRINT f
100 PRINT: GOTO 50
110 REM ****con FOR-NEXT****
120 f=1
130 FOR i=1 TO n
140 f=f*i
150 NEXT
160 RETURN
170 REM ****con WHILE-WEND****
180 f=n
190 WHILE n<>1
200 n=n-1
210 f=f*n
220 WEND
230 RETURN
```

4

```
10 MODE 1
20 INPUT "margen inferior";a
30 INPUT "margen superior";b
40 IF b<a OR a<0 THEN 20
50 CLS
60 PRINT"NUMERO   RAIZ CUADRADA","RAIZ C
UBICA"
70 FOR x=a TO b
80 PRINT x;TAB(10);x^0.5;TAB(27);x^(1/3)
90 NEXT
```

5

```
10 REM el valor inicial del segundo bucl
e es a, para evitar las repeticiones de
fichas
20 MODE 2
30 FOR a=0 TO 6
40 FOR b=a TO 6
50 LOCATE 11*a+1,2*b+1
60 PRINT a;b
70 NEXT b,a
80 LOCATE 1,24
```

6

```
10 MODE 2
20 FOR x=1 TO 6
30 FOR y=1 TO 6
40 FOR z=1 TO 6
50 IF x<>y AND x<>z AND y<>z THEN PRINT
x;y;z,
60 NEXT z,y,x
70 LOCATE 1,24
```

7

```
10 MODE 1
20 INPUT "tabla del";n
30 IF n<1 OR n>9 THEN 20
40 CLS
50 PRINT "TABLA DEL";n:PRINT
60 FOR x=1 TO 9
70 LOCATE 1,2*x+1
80 PRINT n;"x";x;"="
90 LOCATE 1,23:INPUT "el producto vale";
p
100 LOCATE 18,23:PRINT "       "
110 LOCATE 10,2*x+1:PRINT p;"    ";
120 IF p=n*x THEN PRINT "CORRECTO":GOTO
140
130 PRINT "error, es ";n*x
140 NEXT
150 LOCATE 1,23:PRINT"
      "
```

8

```
5 DEFINT X-Z
10 MODE 1
20 INPUT "VALOR LIMITE";N
30 FOR X=1 TO N-3
40 FOR Y=X+1 TO N-2
50 FOR Z=Y+1 TO N-1
60 IF Z*Z=X*X+Y*Y THEN PRINT X;Y;Z,
70 NEXT Z,Y,X
```

Capítulo 8

1

```
10 MODE 1
20 PRINT "ANGULO  SENO       COSENO     TA
NGENTE"
```

```
30 FOR X=1 TO 36:PRINT"=";:NEXT:PRINT
40 DEG
50 FOR X=0 TO 90 STEP 5
60 IF X<>90 THEN PRINT X;TAB (8);ROUND (
SIN (X),4);TAB (18);ROUND (COS (X),4);TA
B (28);ROUND (TAN (X),4)
70 IF X=90 THEN PRINT X;TAB (8);ROUND (S
IN (X),4);TAB (18);ROUND (COS (X),4);TAB
 (29);"--------"
80 NEXT
90 LOCATE 1,24
```

2

```
10 MODE 1
20 INPUT "NUMERO";N:PRINT:PRINT
30 PRINT"SIGNO:";SGN (N)
40 PRINT"VALOR ABSOLUTO:";ABS(N)
50 IF N>=0 THEN PRINT "RAIZ CUADRADA:";S
QR(N)
60 IF N<88.0297 THEN PRINT "EXPONENCIAL:
";EXP(N)
70 IF N>0 THEN PRINT "LOGARITMO NEPERIAN
O:";LOG(N):PRINT"LOGARITMO DECIMAL:";LOG
10(N)
80 PRINT"PARTE ENTERA:";INT(N)
90 PRINT"SIN PARTE DECIMAL:";FIX(N)
100 PRINT"REDONDEO:";ROUND (N)
110 IF N>-32769 AND N<32768 THEN PRINT "
ENTERO MAS CERCANO";CINT (N)
120 PRINT"SENO:";SIN(N)
130 PRINT"COSENO:";COS (N)
140 IF COS (N)<>0 THEN PRINT "TANGENTE:"
;TAN (N)
150 PRINT"ARCO TANGENTE:";ATN(N)
160 LOCATE 1,24
```

3

```
10 FOR x=12 TO 49998  STEP 2
20 FOR y=3 TO 11 STEP 2
30 IF x-INT(x/y)*y<>2 THEN 60
40 NEXT
50 PRINT x
60 NEXT
```

4

```
10 INPUT "DAME UN NUMERO";N
15 IF N<>INT(N) OR N<2 THEN 10
20 CLS
```

```
30 PRINT"NUMEROS PRIMOS MENORES QUE";N
35 PRINT
40 PRINT 1,2,
50 FOR X=3 TO N-1
60 FOR Y=2 TO SQR(X)
70 IF X/Y=INT(X/Y) THEN 100
80 NEXT
90 PRINT X,
100 NEXT
110 PRINT:PRINT
```

5

```
10 INPUT"ESCRIBE UN NUMERO";N
20 CLS
30 PRINT N;"=";
40 FOR X=2 TO SQR(N)
50 IF INT (N/X)=N/X THEN 80
60 NEXT
70 PRINT N:PRINT:PRINT:END
80 PRINT X;"*";
90 IF N=X THEN PRINT:PRINT: END
100 N=N/X:GOTO 40
```

6

```
10 INPUT "DAME UN NUMERO";N
20 IF N<>INT(N) OR N<2 THEN 10
30 S=1
40 CLS
50 FOR X=2 TO N/2
60 IF INT(N/X)=N/X THEN S=S+X
70 NEXT
80 IF S=N THEN PRINT N;"ES PERFECTO":END
90 PRINT N;"NO ES PERFECTO"
```

7

```
10 INPUT "DAME LOS DOS NUMEROS";A,B
11 IF A<0 OR B<0 THEN 10
12 IF A<>INT(A) OR B<>INT(B) THEN 10
20 SA=1
30 FOR X=2 TO A/2
40 IF INT(A/X)=A/X THEN SA=SA+X
60 NEXT
70 SB=1
80 FOR X=2 TO B/2
90 IF INT(B/X)=B/X THEN SB=SB+X
100 NEXT
110 IF A=SB AND B=SA THEN PRINT"SON AMIG
OS":END
120 PRINT"NO SON AMIGOS"
```

Capítulo 9

1

```
10 MODE 1
20 PRINT
30 INPUT "escribe un verbo";a$
40 b$=RIGHT$(UPPER$(a$),2)
50 IF b$="AR" THEN PRINT "PRIMERA CONJUG
ACION":GOTO 20  ELSE IF b$="ER" THEN PRI
NT "SEGUNDA CONJUGACION":GOTO 20  ELSE I
F b$="IR" THEN PRINT "TERCERA CONJUGACIO
N":GOTO 20 ELSE PRINT "NO ES EL INFINITI
VO DE UN VERBO":GOTO 20
```

2

```
10 MODE 1
20 INPUT "escribe un verbo";a$
30 b$=RIGHT$(UPPER$(a$),2)
40 IF b$<>"AR" AND b$<>"ER" THEN PRINT:P
RINT "NO ES EL INFINITIVO DE UN VERBO DE
 LA   PRIMERA O SEGUNDA CONJUGACION":FOR
 X=1 TO 3000:NEXT:RUN
50 CLS
60 LOCATE 8,1:PRINT "VERBO ";CHR$(24);UP
PER$(A$);CHR$(24)
70 RAIZ$=LEFT$(UPPER$(A$),LEN(A$)-1)
80 LOCATE 8,5:PRINT "YO ";LEFT$(RAIZ$,LE
N(RAIZ$)-1);"O
90 LOCATE 8,8:PRINT "TU ";RAIZ$;"S"
100 LOCATE 8,11:PRINT "EL ";RAIZ$
110 LOCATE 8,14:PRINT "NOSOTROS ";RAIZ$;
"MOS"
120 LOCATE 8,17:PRINT "VOSOTROS ";RAIZ$;
"IS"
130 LOCATE 8,20:PRINT "ELLOS ";RAIZ$;"N"
140 LOCATE 1,25:GOTO 20
```

3

```
10 MODE 2
20 LOCATE 1,8
30 LINE INPUT "ESCRIBE LA FRASE ";A$
40 LOCATE 1,16
50 LINE INPUT "CARACTER BUSCADO? ";B$
60 CLS
70 C=0
80 FOR I=1 TO LEN(A$)
```

```
90 LOCATE 1+I MOD 80,2+2*(I\80):PRINT MI
D$(A$,I,1)
100 LOCATE 1+I MOD 80,8+2*(I\80):IF MID$
(A$,I,1)=B$ THEN PRINT B$:C=C+1 ELSE PRI
NT"-"
110 NEXT
120 LOCATE 1,15:PRINT CHR$(24); B$;" APA
RECE ";C;"VECES EN LA FRASE";CHR$(24)
130 LOCATE 1,19
140 PRINT "PULSAR OPCION:"
150 PRINT TAB(12);"1   ESTUDIAR LA MISMA
 FRASE"
160 PRINT TAB(12);"2   ESTUDIAR OTRA FRA
SE "
170 INPUT OPCION
180 ON OPCION GOTO 50,10
```

4

```
10 MODE 1
20 LOCATE 1,8
30 LINE INPUT "ESCRIBE LA FRASE ";A$
40 LOCATE 1,16
50 LINE INPUT "PALABRA BUSCADA? ";P$
60 CLS
70 PRINT "FRASE:"
80 PRINT:PRINT A$
90 C$=" "+A$+" "
100 B$=" "+P$+" "
110 C=0
120 IF INSTR(C$,B$)=0 THEN GOTO 160
130 C=C+1
140 X=LEN(C$)-(INSTR(C$,B$)+LEN(B$)-2):I
F X<=0 THEN 160
150 C$=RIGHT$(C$,X):GOTO 120
160 LOCATE 1,15:PRINT CHR$(24); B$;" APA
RECE ";C;"VECES EN LA FRASE";CHR$(24)
170 LOCATE 1,19
180 PRINT "PULSAR OPCION:"
190 PRINT TAB(12);"1   ESTUDIAR LA MISMA
 FRASE"
200 PRINT TAB(12);"2   ESTUDIAR OTRA FRA
SE "
210 INPUT OPCION
220 ON OPCION GOTO 50,10
```

5

```
10 MODE 1
20 PRINT:PRINT:INPUT"NUMERO";A$:IF LEN(A
$)<>4 THEN 20
```

```
30 IF VAL(LEFT$(A$,1))=0 THEN 70
40 IF VAL(LEFT$(A$,1))=1 THEN 60
50 ON VAL (LEFT$(A$,1))-1 GOSUB 420,430,
440,450,460,470,480,490
60 PRINT" MIL ";
70 IF VAL(MID$(A$,2,1))=0 THEN 150
80 IF VAL(RIGHT$(A$,3))=100 THEN PRINT "
CIEN":GOTO 20
90 IF VAL(MID$(A$,2,1))=1 THEN PRINT"CIE
NTO ";:GOTO 150
100 IF VAL(MID$(A$,2,1))=5 THEN PRINT"QU
INIENTOS ";:GOTO 150
110 IF VAL(MID$(A$,2,1))=7 THEN PRINT "S
ETE";:GOTO 140
120 IF VAL(MID$(A$,2,1))=9 THEN PRINT"NO
VE";:GOTO 140
130 ON VAL(MID$(A$,2,1))-1 GOSUB 420,430
,440,450,460,470,480,490
140 PRINT"CIENTOS ";
150 IF VAL(MID$(A$,3,1))=0 THEN 250
160 IF VAL(RIGHT$(A$,2))<20 THEN 320
170 IF VAL(MID$(A$,3,1))=2 THEN PRINT "V
EINTE ";
180 IF VAL(MID$(A$,3,1))=3 THEN PRINT "T
REINTA ";
190 IF VAL(MID$(A$,3,1))=4 THEN PRINT "C
UARENTA ";
200 IF VAL(MID$(A$,3,1))=5 THEN PRINT "C
INCUENTA ";
210 IF VAL(MID$(A$,3,1))=6 THEN PRINT "S
ESENTA ";
220 IF VAL(MID$(A$,3,1))=7 THEN PRINT "S
ETENTA ";
230 IF VAL(MID$(A$,3,1))=8 THEN PRINT "O
CHENTA ";
240 IF VAL(MID$(A$,3,1))=9 THEN PRINT "N
OVENTA ";
250 IF VAL(RIGHT$(A$,1))=0 THEN 300
260 IF VAL(RIGHT$(A$,1))=1 AND VAL(MID$(
A$,3,1))<>0 THEN PRINT "Y UNO":GOTO 310
270 IF VAL(RIGHT$(A$,1))=1 AND VAL(MID$(
A$,3,1))=0 THEN PRINT "UNO":GOTO 310
280 IF VAL(MID$(A$,3,1))<>0 THEN PRINT "
Y ";
290 ON VAL(RIGHT$(A$,1))-1 GOSUB 420,430
,440,450,460,470,480,490
300 PRINT
310 GOTO 20
320 IF VAL(RIGHT$(A$,2))=10 THEN PRINT "
DIEZ":GOTO 20
```

```
330 IF VAL(RIGHT$(A$,2))=11 THEN PRINT "
ONCE":GOTO 20
340 IF VAL(RIGHT$(A$,2))=12 THEN PRINT "
DOCE":GOTO 20
350 IF VAL(RIGHT$(A$,2))=13 THEN PRINT "
TRECE":GOTO 20
360 IF VAL(RIGHT$(A$,2))=14 THEN PRINT "
CATORCE":GOTO 20
370 IF VAL(RIGHT$(A$,2))=15 THEN PRINT "
QUINCE":GOTO 20
380 IF VAL(RIGHT$(A$,2))=16 THEN PRINT "
DIECISEIS":GOTO 20
390 IF VAL(RIGHT$(A$,2))=17 THEN PRINT "
DIECISIETE":GOTO 20
400 IF VAL(RIGHT$(A$,2))=18 THEN PRINT "
DIECIOCHO":GOTO 20
410 IF VAL(RIGHT$(A$,2))=19 THEN PRINT "
DIECINUEVE":GOTO 20
420 PRINT"DOS";:RETURN
430 PRINT"TRES";:RETURN
440 PRINT"CUATRO";:RETURN
450 PRINT"CINCO";:RETURN
460 PRINT"SEIS";:RETURN
470 PRINT"SIETE";:RETURN
480 PRINT"OCHO";:RETURN
490 PRINT"NUEVE";:RETURN
```

Capítulo 10

1

```
10 MODE 1
20 RANDOMIZE TIME
30 PRINT "LOTERIA PRIMITIVA"
40 RANDOMIZE TIME
50 A=INT (49*RND)+1
60 B=INT (49*RND)+1
70 C=INT (49*RND)+1
80 D=INT (49*RND)+1
90 E=INT (49*RND)+1
100 F=INT (49*RND)+1
110 IF A=B OR A=C OR A=D OR A=E OR A=F O
R B=C OR B=D OR B=E OR B=F OR C=D OR C=E
 OR C=F OR D=E OR D=F OR E=F THEN 40
120 FOR X=1 TO 7
```

```
130 FOR Y=1 TO 7
140 LOCATE 5*X-4,3*Y+1:LET N=7*(X-1)+Y
150 IF N=A OR N=B OR N=C OR N=D OR N=E O
R N=F THEN PRINT CHR$(24);N;CHR$(24):GOT
O 170
160 PRINT N
170 NEXT Y
180 NEXT X
190 LOCATE 1,24
```

2

```
10 MODE 1
20 RANDOMIZE TIME
30 A=INT(100*RND)
40 B=INT(100*RND)
50 PRINT "CUAL ES EL VALOR DE";A;"+";B,
60 INPUT S
70 IF S=A+B THEN PRINT "CORRECTO":GOTO 9
0 ELSE PRINT "incorrecto":C=C+1:IF C=1 T
HEN 60
80 PRINT "la respuesta es";A+B
90 PRINT:PRINT
100 PRINT "CUAL ES EL VALOR DE";A;"*";B,
110 INPUT P
120 IF P=A*B THEN PRINT "CORRECTO":GOTO
140  ELSE PRINT "incorrecto":L=L+1:IF L=
1 THEN 110
130 PRINT "la respuesta es";A*B
140 LOCATE 1,24
```

3

```
10 MODE 1
20 P=1:Q=100
30 LOCATE 1,24:PRINT SPACE$(18)
40 IF P>Q THEN LOCATE 14,15:PRINT CHR$(2
4);"TRAMPOSO";CHR$(24):END
50 X=INT ((Q-P+1)*RND)+P
60 LOCATE 1,2:PRINT X;" ES :"
70 LOCATE 10,4:PRINT "1. MAYOR QUE SU NU
MERO"
80 LOCATE 10,6:PRINT "2. MENOR QUE SU NU
MERO"
90 LOCATE 10,8:PRINT "3. SU NUMERO"
100 LOCATE 1,24:INPUT "OPCION ELEGIDA";A
110 IF A<>1 AND A<>2 AND A<>3 THEN 100
120 N=N+1
130 IF A=3 THEN LOCATE 1,24:PRINT "HE EN
CONTRADO EN";N;"INTENTOS SU NUMERO":END
140 IF A=1 THEN Q=X-1:GOTO 30 ELSE P=X+1
:GOTO 30
```

4

```
10 MODE 1
20 RANDOMIZE TIME
30 A=INT (RND*9)+1:B=INT(10*RND)
40 C=INT(10*RND):D=INT(10*RND)
50 IF A=B OR A=C OR A=D OR B=C OR B=D OR
 C=D THEN 30
60 FOR N=1 TO 9
70 INPUT "numero";Z:IF Z<1000 OR Z>9999
THEN 70
80 Z$=STR$(Z)
90 U=VAL(MID$(Z$,2,1))
100 V=VAL(MID$(Z$,3,1))
110 X=VAL(MID$(Z$,4,1))
120 Y=VAL(MID$(Z$,5,1))
130 IF A=U THEN M=M+1
140 IF B=V THEN M=M+1
150 IF C=X THEN M=M+1
160 IF D=Y THEN M=M+1
170 IF A=V OR A=X OR A=Y THEN H=H+1
180 IF B=U OR B=X OR B=Y THEN H=H+1
190 IF C=U OR C=V OR C=Y THEN H=H+1
200 IF D=U OR D=V OR D=X THEN H=H+1
210 PRINT "MUERTOS:";M,
220 PRINT "HERIDOS:";H:PRINT
230 IF M=4 THEN PRINT:PRINT "HAS ACERTAD
O EN";N;"INTENTOS":PRINT:END
240 M=0:H=0:PRINT
250 NEXT
260 PRINT:PRINT "EL NUMERO ERA  ";A;B;C;
D:PRINT
```

5

```
10 RANDOMIZE TIME
20 MODE 1
30 N=INT (RND*1000000)
40 PRINT "NUMERO INICIAL ";N:PRINT
50 P=3*N+3*N-1+3*N-2
60 PRINT "TRAS EL PRODUCTO ";P:PRINT
70 A$=STR$(P):A$=RIGHT$(A$,LEN (A$)-1)
80 FOR K=1 TO LEN (A$)
90 S=S+VAL(MID$(A$,K,1))
100 NEXT
110 IF S<10 THEN PRINT:PRINT "SUMA FINAL
";S:PRINT:END
120 PRINT"SUMA PARCIAL";S:PRINT
130 P=S:S=0:GOTO 70
```

6

```
10 MODE 1
20 INPUT "ESCRIBE UNA PALABRA CON TODAS
LAS LETRAS DISTINTAS ";A$
30 CLS
40 PRINT "LA PALABRA A ENCONTRAR TIENE";
LEN (A$);"LETRAS"
50 PRINT:PRINT
60 N=1
70 INPUT "PALABRA";B$:IF LEN (B$)<>LEN(A
$) THEN 70
80 IF A$=B$ THEN PRINT:PRINT "HAS ACERTA
DO EN";N;"INTENTOS":PRINT:END
90 P=0
100 FOR I=1 TO LEN(B$)
110 FOR J=1 TO LEN (A$)
120 IF MID$(B$,I,1)=MID$(A$,J,1) THEN P=
P+1
130 NEXT J,I
140 IF P MOD 2=0 THEN PRINT"par" ELSE PR
INT "impar"
150 N=N+1:PRINT:GOTO 70
```

Capítulo 11

1

```
10 MODE 1
20 RANDOMIZE TIME
30 DIM a(90)
40 LOCATE 15,2
50 PRINT CHR$(24);"B I N G O";CHR$(24)
60 FOR X=1 TO 90
70 N=INT (90*RND)+1
80 IF A(N)<>0 THEN 70
90 A(N)=N
100 M=Nñ10
110 P=N MOD 10
120 IF P=0 THEN P=10:M=M-1
130 LOCATE 4*P-3,2*M+6
140 PRINT CHR$(24);A(N);CHR$(24)
150 PRINT CHR$(7)
160 FOR T=1 TO 2000:NEXT
170 LOCATE 4*P-3,2*M+6
180 PRINT A(N)
190 NEXT
200 LOCATE 1,24
```

2

```
10 REM POR CUESTION DE IMPRESION NO INTR
ODUCIR ORDENES SUPERIORES A 19
20 MODE 2
30 INPUT "ORDEN DEL CUADRADO";N
40 IF N MOD 2=0 THEN RUN
50 DIM A(N,N)
60 I=(N-1)/2:J=(N+1)/2:A(I,J)=1
70 FOR K=2 TO N*N
80 J=J+1
90 IF J<=N THEN 110
100 J=1
110 I=I-1
120 IF I<>0 THEN 140
130 I=N
140 IF A(I,J)=0 THEN A(I,J)=K:GOTO 180
150 J=J-1
160 IF J<>0 THEN 110
170 J=N:GOTO 110
180 NEXT
190 FOR I=1 TO N
200 FOR J=1 TO N
210 IF N<12 THEN LOCATE 5*J,2*I ELSE LOC
ATE 4*J,I+1
220 PRINT A(I,J)
230 NEXT
240 NEXT
250 LOCATE 1,24
```

3

```
10 MODE 1
20 RANDOMIZE TIME
30 INPUT "CUANTOS NUMEROS DESEA";NUM
40 CLS
50 DIM a(49)
60 LOCATE 10,1
70 PRINT CHR$(24);"LOTERIA PRIMITIVA";CH
R$(24)
80 FOR X=1 TO NUM
90 N=INT (49*RND)+1
100 IF A(N)<>0 THEN 90
110 A(N)=N
120 NEXT
130 FOR X=1 TO 40 STEP 6
140 FOR Z=2 TO 20 STEP 3
150 C=C+1
160 LOCATE X,Z+1
170 IF A(C)<>0 THEN PRINT CHR$(24);C;CHR
$(24) ELSE PRINT C
```

```
180 NEXT
190 NEXT
200 LOCATE 1,24
```

4

```
10 DEFINT A-Z
20 ZONE 8
30 MODE 1
40 LOCATE 8,10:PRINT"ESPERAR UNOS 20 SEG
UNDOS"
50 RANDOMIZE TIME
60 DIM A(200)
70 K=1
80 I=INT(200*RND)+1
90 A(I)=A(I)+1
100 N=N+1
110 FOR J=K TO 200
120 IF A(J)=0 THEN K=J:GOTO 80
130 NEXT
140 MODE 2
150 PRINT"HA NECESITADO COMPRAR";N;"CROM
OS"
160 PRINT
170 FOR L=1 TO 200
180 PRINT A(L),
190 NEXT
200 LOCATE 1,24
```

Capítulo 12

1

```
10 REM INTRODUCIR LOS TELEFONOS EN LA LI
NEA 40
20 REM MODIFICAR CONVENIENTEMENTE EL VAL
OR DE LA LINEA 70
30 MODE 1
40 DATA MIGUEL,111111,LAURA,456789,ANDRE
S,909090,JULIAN,232323,MARIA,343434
50 PRINT:PRINT
60 INPUT "NOMBRE DEL USUARIO";A$:A$=UPPE
R$(A$)
70 NUMERODETELEFONOS=5
80 FOR I=1 TO NUMERODETELEFONOS
90 READ X$,Y
```

```
100 IF X$=A$ THEN PRINT "SU TELEFONO ES
";Y:RESTORE:GOTO 50
110 NEXT
120 PRINT "NO ESTA INCLUIDO EN EL FICHER
O"
130 RESTORE
140 GOTO 50
```

2

```
10 REM INTRODUCIR LAS PALABRAS DESEADAS,
EN MAYUSCULAS, EN LA LINEA 40.PRIMERO EN
 CASTELLANO Y LUEGO EN INGLES
20 REM MODIFICAR ADECUADAMENTE EL VALOR
DE LA LINEA 50
30 MODE 1
40 DATA CASA,HOUSE,CHICO,BOY,CHICA,GIRL,
CABALLO,HORSE,DISCO,DISK,PAZ,PEACE
50 NUMERODEPALABRAS=6
60 INPUT "PALABRA A BUSCAR ";X$:X$=UPPER
$(X$)
70 PRINT
80 FOR I=1 TO NUMERODEPALABRAS
90 READ A$,B$
100 IF A$=X$ THEN PRINT"EN INGLES ES ";B
$:GOTO 140
110 IF B$=X$ THEN PRINT "EN CASTELLANO E
S ";A$:GOTO 140
120 NEXT
130 PRINT"NO SE ENCUENTRA EN EL DICCIONA
RIO"
140 RESTORE
150 PRINT:PRINT:PRINT
160 GOTO 60
```

3

```
10 ZONE 8:MODE 1
20 DIM A$(36)
30 FOR X=1 TO 36:READ A$(X):NEXT
40 PRINT:PRINT:PRINT
50 INPUT "ESCRIBA LA FRASE";F$
60 PRINT
70 F$=UPPER$(F$)
80 FOR I=1 TO LEN(F$)
90 B$=MID$(F$,I,1)
100 IF ASC(B$)<48 THEN 160
110 IF ASC(B$)>90 THEN 160
120 IF ASC(B$)>57 AND ASC(B$)<65  THEN 1
60
```

```
130 C=ASC(B$)-54
140 IF C<=0 THEN C=C+7
150 PRINT B$;" ";A$(C),
160 NEXT
170 GOTO 40
180 DATA -----,.----,..---,...--,....-,.
....,-....,--...,---..,----.,.-,-...,-.-
.,-..,.,..-.,--.,....,..,.---,-.-,.-..,-
-,-.,---,.--.,--.-,.-.,...,-,..-,...-,.-
-,-...,-.--,--..
```

Capítulo 15

1

```
90 FOR Y=20 TO 400 STEP 45
100 PLOT 120,Y:DRAWR 400,0
110 NEXT
120 FOR X=120 TO 520 STEP 50
130 PLOT X,20:DRAWR 0,360
140 NEXT
```

2

```
10 MODE 1
20 TAG
30 FOR X=0 TO 7
40 MOVE 145+50*X,16
50 PRINT CHR$(65+X);
60 MOVE 85,42+45*X
70 PRINT X+1;
80 NEXT
```

3

```
10 MODE 2:MOVE 25,0
20 FOR X=1 TO 8
30 READ A,B:DRAWR A,B
40 NEXT
50 DATA 0,260,390,0,-195,130,-195,-130,3
90,-260,0,260,-390,-260,390,0
```

4

```
10 MODE 0
20 MOVE 0,52
30 DRAW 639,52:DRAW 639,348
```

```
40 DRAW 0,348:DRAW 0,52
50 FOR X=1 TO 5
60 READ A,B,C,D
70 MOVE A,B:DRAW C,D
80 NEXT
90 GOTO 90
100 DATA 319,52,319,348,0,89,639,89,0,31
1,639,311,148,89,148,311,492,89,492,311
```

5

```
10 SYMBOL 255,0,60,0,60,36,36,36,0
20 PRINT CHR$(255)
```

6

```
10 MODE 1
20 ORIGIN 319,199
30 DEG:TAG
40 FOR X=1 TO 12
50 MOVE 180*COS(60+30*X)-30,180*SIN(60+3
0*X)
60 PRINT 13-X;
70 NEXT
80 MOVE 200,0
90 FOR X=0 TO 360 STEP 10
100 DRAW 199*COS(X),199*SIN(X)
110 NEXT
120 MOVE 0,0:DRAW 0,160
130 MOVE 0,0:DRAW 100,0
```

Capítulo 16

1

```
10 REM VERSION 664-6128
20 MODE 1
30 TAG
40 FOR X=0 TO 7
50 MOVE 145+50*X,16
60 PRINT CHR$(65+X);
70 MOVE 85,42+45*X
80 PRINT X+1;
90 NEXT
100 FOR Y=20 TO 380 STEP 45
```

```
110 MOVE 120,Y:DRAWR 400,0
120 NEXT
130 FOR X=120 TO 520 STEP 50
140 MOVE X,20:DRAWR 0,360
150 NEXT
160 FOR x=145 TO 445 STEP 100
170 FOR y=42 TO 312 STEP 90
180 MOVE x,y:FILL 1
190 MOVER 50,45:FILL 1
200 NEXT:NEXT
```

1

```
10 REM VERSION 446 Y 472
20 MODE 1
30 TAG
40 FOR X=0 TO 7
50 MOVE 145+50*X,16
60 PRINT CHR$(65+X);
70 MOVE 85,42+45*X
80 PRINT X+1;
90 NEXT
100 MOVE 122,22:DRAWR  400,0
110 DRAWR 0,360:DRAWR -400,0:DRAWR 0,-360
120 FOR x=122 TO 422 STEP 100
130 FOR y=22 TO 292 STEP 90
140 FOR z=0 TO 42
150 MOVE x,y+z:DRAWR 48,0
160 MOVE x+50,y+45+z:DRAWR 48,0
170 NEXT:NEXT:NEXT
```

2

```
10 REM VERSION 664-6128
20 MODE 0
30 ORIGIN 319,199
40 DEG
50 MOVE 199,0
60 FOR x=0 TO 360 STEP 15:DRAW 199*COS(x),199*SIN(x):NEXT
70 MOVE 150,0
80 FOR x=0 TO 360 STEP 15:DRAW 150*COS(x),150*SIN(x):NEXT
90 MOVE 100,0
100 FOR x=0 TO 360 STEP 15:DRAW 100*COS(x),100*SIN(x):NEXT
110 MOVE 50,0
120 FOR x=0 TO 360 STEP 15:DRAW 50*COS(x),50*SIN(x):NEXT
130 MOVE 0,60:FILL 3
140 MOVE 0,0:FILL 1
150 MOVE 0,140:FILL 10
160 MOVE 0,180:FILL 5
```

3

```
10 REM VERSION 664-6128
20 MODE 0
30 MOVE 0,133:DRAWR 639,0
40 MOVE 0,266:DRAWR 639,0
50 MOVE 0,100:FILL 3
60 MOVE 0,150:FILL 1
70 MOVE 0,300:FILL 3
80 GOTO 80
```

3

```
10 REM VERSION 464-472
20 MODE 0
30 FOR Y=0 TO 132:MOVE 0,Y:DRAWR 639,0,3:NEXT
40 FOR Y=134 TO 265:MOVE 0,Y:DRAWR 639,0,1:NEXT
50 FOR Y=267 TO 400:MOVE 0,Y:DRAWR 639,0,3:NEXT
60 GOTO 60
```

4

```
10 REM VERSION 664-6128
20 MODE 0
30 MOVE 200,0:DRAWR 0,399
40 MOVE 400,0:DRAWR 0,399
50 MOVE 0,0:FILL 5
60 MOVE 204,200:FILL 1
70 MOVE 404,200:FILL 3
80 GOTO 80
```

4

```
10 REM VERSION 464-472
20 MODE 0
30 MOVE 200,0:DRAWR 0,400
40 MOVE 400,0:DRAWR 0,400
50 FOR X=0 TO 200 STEP 4:MOVE X,0:DRAWR 0,399,5:NEXT
60 FOR X=201 TO 400 STEP 4:MOVE X,0:DRAWR 0,399,1:NEXT
70 FOR X=401 TO 639 STEP 4:MOVE X,0:DRAWR 0,399,3:NEXT
80 GOTO 80
```

5

```
10 REM VERSION 664-6128
20 MODE 2
30 INK 0,26:INK 1,6
40 MOVE 270,50
50 FOR X=1 TO 12
60 READ A,B:DRAWR A,B
70 NEXT
80 MOVE 320,200
90 MOVE 320,200:FILL 1
```

```
100 GOTO 100
110 DATA 100,0,0,100,100,0,0,100,-100,0
120 DATA 0,100,-100,0,0,-100,-100,0,0,-100
130 DATA 100,0,0,-100
```

5

```
10 REM version 464-472
20 MODE 0
30 INK 0,26:INK 1,6
40 FOR x=150 TO 450
50 MOVE x,150:DRAWR 0,100
60 NEXT
70 FOR y=50 TO 350
80 MOVE 250,y:DRAWR 100,0
90 NEXT
100 GOTO 100
```

6

```
10 REM VERSION 664-6128
20 MODE 1:DEG
30 ORIGIN 319,199:MOVE 195,0
40 FOR x=0 TO 360 STEP 10
50 DRAW 195*COS(x),195*SIN(x)
60 NEXT
70 MOVE 91,-42
80 FOR X=-25 TO 115 STEP 10
90 DRAW 100*COS(x),100*SIN(x)
100 NEXT
110 DRAW 91,-42
120 MOVE -92,42
130 FOR X=155 TO 295 STEP 10
140 DRAW 100*COS(x),100*SIN(x)
150 NEXT
160 DRAW -91,42
170 MOVE 0,0:FILL 3
180 GOTO 180
```

7

```
10 REM MASK
20 INK 2,25
30 GRAPHICS PAPER 3
40 MODE 1:MOVE 185,200
50 DRAWR 256,0:DRAWR 0,32:DRAWR -256,0
55 LOCATE 1,20
56 PRINT"   PRESIONAR UNA TECLA PARA CONTINUAR"
60 FOR X=1 TO 8
70 DRAWR 0,-32:MOVER 32,32
```

```
80 NEXT
90 FOR y=0 TO 255
100 TAGOFF:LOCATE 17,8:PRINT"MASK";y
110 a$=BIN$(y,8)
120 FOR x=1 TO 8
130 IF MID$(a$,x,1)="1" THEN c=2 ELSE c=3
140 MOVE 169+32*x,216:FILL c
150 TAG:MOVER -8,-25:PRINT MID$(a$,x,1);
160 MOVER 8,25:NEXT:MASK y
170 MOVE 0,0:DRAWR 639,0:DRAWR 0,399
180 DRAWR -639,0:DRAWR 0,-399
190 IF INKEY$=""THEN 190 ELSE NEXT
```

Otro libro sobre AMSTRAD publicado por

PARANINFO S.A.

LOS MEJORES PROGRAMAS PARA EL AMSTRAD

R. Erskine, H. Walwyn, P. Stanley y M. Bews.
288 páginas. 21,5 x 15,5 cm.

Válido para todos los modelos CPC, constituye toda una biblioteca de programas por un precio más bajo que el de un casete. Juegos explosivos, gráficos dinámicos, invalorables utilidades. Una colección que lleva al lenguaje BASIC hasta el mejor aprovechamiento de sus cualidades.

Recopila 60 programas para Amstrad, escritos por cuatro programadores británicos, autores de muchos de los juegos más vendidos del mercado. Los cuatro han aplicado sus talentos para superar los clichés de la programación y explotar la potencialidad del ordenador.

Un libro imprescindible para los amantes de los juegos de ordenador y para usuarios de aspiraciones más serias.

Los programas que contiene son:

- CHOMPER
- CAMPEONATO DE BOXEO
- DAMBUSTER
- INVASORES
- FALLGUY
- BARCOS
- CARRERAS DE CABALLOS
- CARRERA ESTELAR
- LA MINA
- EVOLUCION 1
- EVOLUCION 2
- EVOLUCION 3
- RESCATE
- CRUZAR EL RIO
- PAREJAS
- PAGAS
- DIBUJOS
- ALUNIZAJE
- COCODRILOS-COMEPAREDES
- TRAFICO ESPACIAL
- RAICES
- SUMA DE PALABRAS
- PESCA SUBMARINA
- ANAGRAMATICA
- BIORRITMOS
- ALIENIGENA
- PERRO PASTOR
- ABEJA ZUMBADORA
- RATAS GIGANTES
- NOCHEBUENA
- BRICKLAYER
- CAZA DE GANSOS
- NEWMARKET
- CAMPO DE ENERGIA
- TOROS Y VACAS
- INSTANTANEO
- AGENDA
- DIBUJOS EN 3-D
- ORGANO MUSICAL
- BINGO
- SUBMARINO
- PROFESOR DE MORSE
- TENIS
- HELICOPTERO BOMBARDERO
- BOMBARDERO AEREO
- FRONTON
- CODIGO SECRETO
- EXOCET
- GENERADOR DE ENVOLVENTES
- LUCHA A MUERTE
- CODIGO MAQUINA
- CONTEMOS
- DUELO AEREO
- INVASION DE HONGOS
- RULETA RUSA
- LA BOLSA
- INVASORES

EL LIBRO DEL AMSTRAD

Con este libro tendrá la oportunidad de conocer exhaustivamente el lenguaje BASIC del ordenador AMSTRAD en los modelos CPC 464, CPC 472, CPC 664 y CPC 6128.

Es importante resaltar que este volumen no es una adaptación de los modelos anteriores al nuevo CPC 6128, sino que se ha escrito para todos ellos, indicando y desarrollando las particularidades de cada uno.

Cualquier lector, aunque no tenga conocimientos del lenguaje BASIC, podrá seguir fácilmente el desarrollo de todos los capítulos. Los diferentes conceptos se van introduciendo poco a poco hasta llegar a las cuestiones más complejas.

Gran cantidad de ejemplos y ejercicios resueltos acompañan a los desarrollos teóricos y, para afianzar todavía más los conocimientos adquiridos, al final de los capítulos se proponen una serie de programas para realizar, cuyas soluciones se encuentran en las páginas finales del libro.

Magallanes, 25 - 28015 Madrid

ISBN: 84-283-1486-1