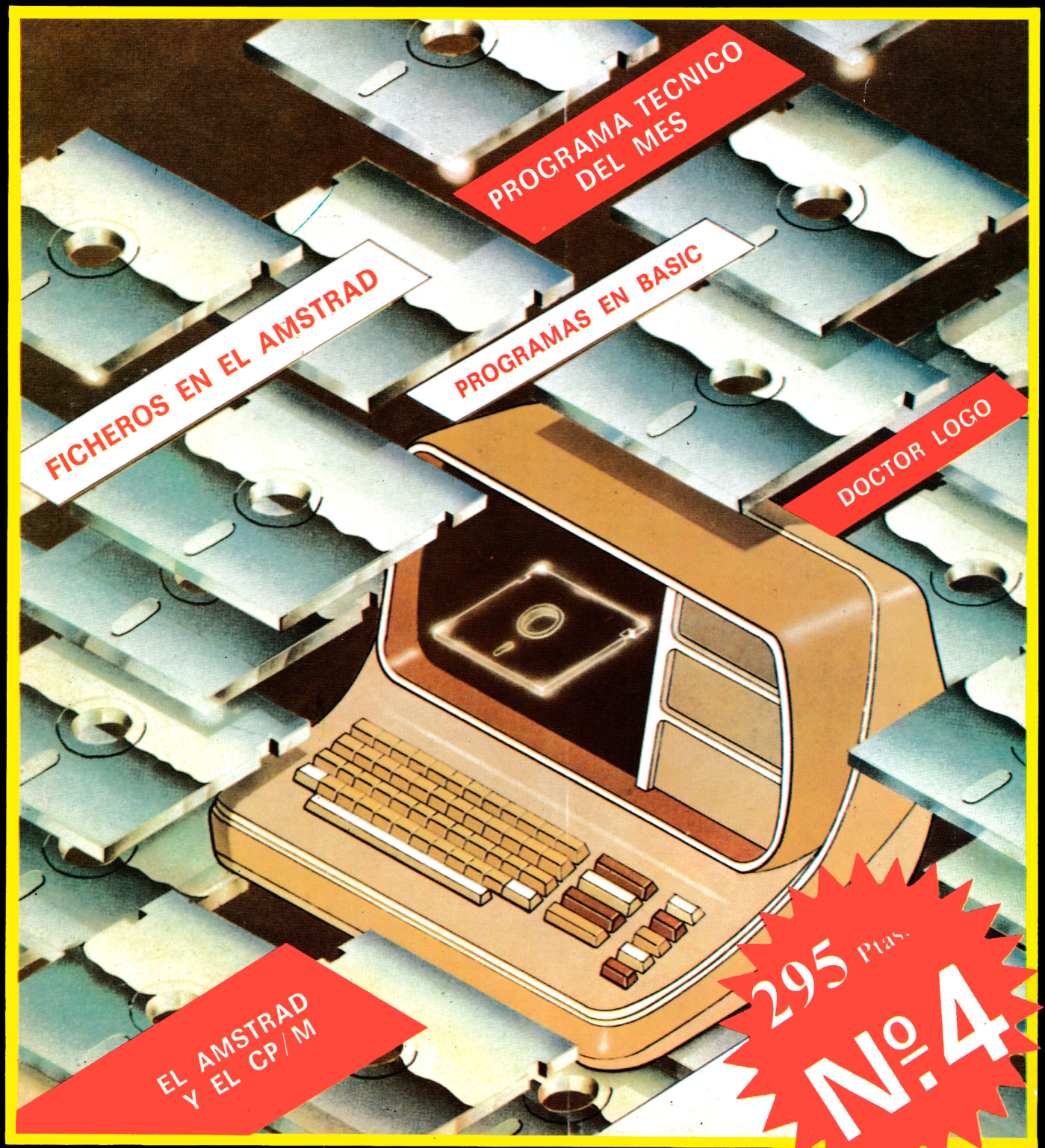
AMSTRAD EDUCATIVO

- PROGRAMANDO EN BASIC Y LOGO PROGRAMAS COMENTADOS
- EL AMSTRAD Y SU CP/M
- COMO CREAR FICHEROS EN DISCO



TODO SOBRE EL

AMSTRAD

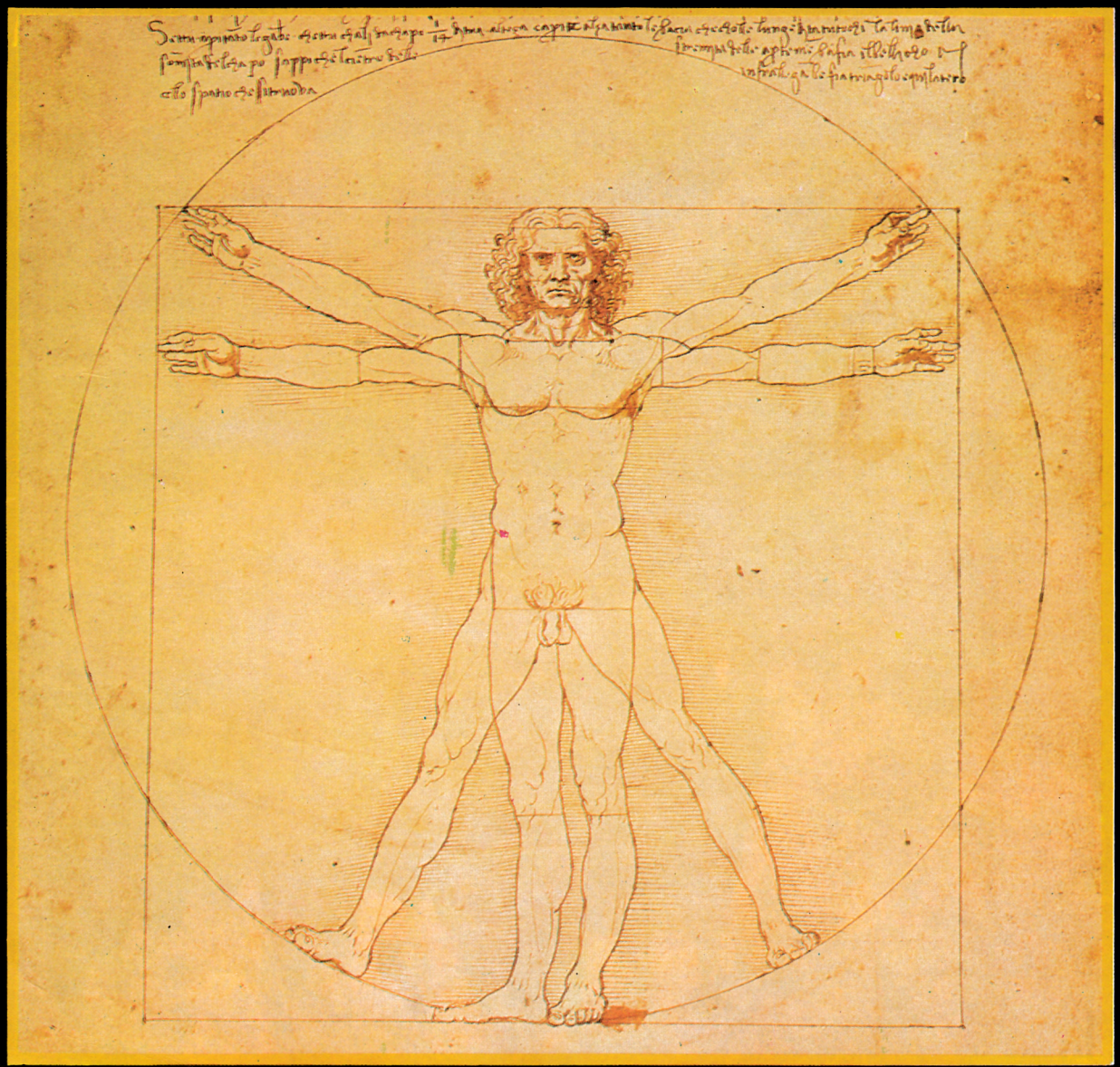
AÑO II - N.° 8

420 Ptas.

10 PROGRAMAS

SUMARIO:

- MODULO
- ARTILLERIA
- SLIPPERY
- DIBUJO
- PHANTOM FIRE
- SUICIDA
- NUCLEO
- NAUFRAGO
- MUSCULOS



EDITORIAL

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

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

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

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

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

Esperamos vuestras opiniones.

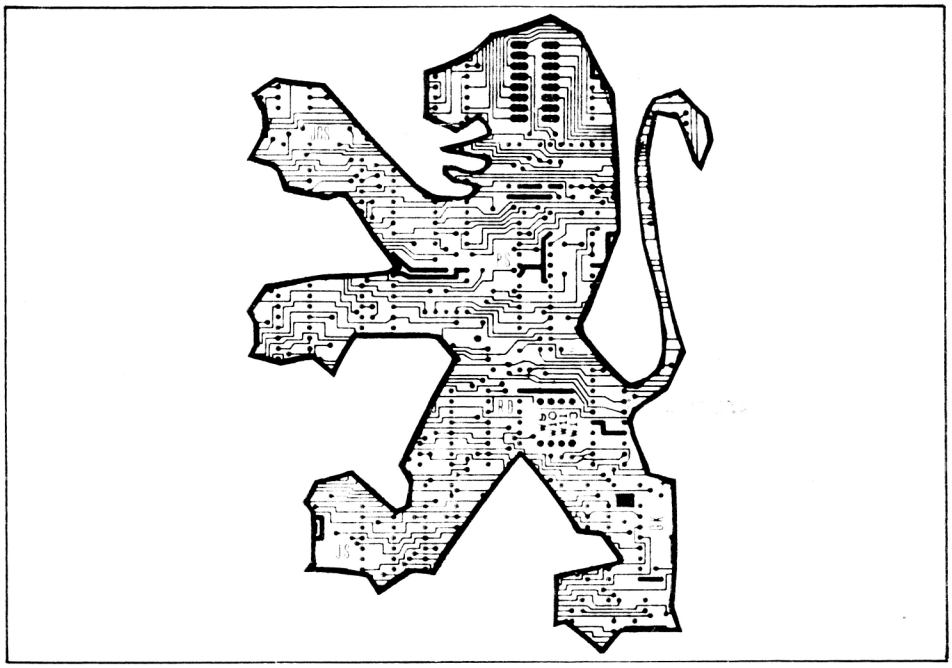
SUMARIO

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

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

EL AMSTRAD Y EL CP/M

Por A. BELLIDO



EJERCICIOS CON UNIDADES DE DISCO

1.° ¿Cuál es la unidad de disco por defecto si la impronta en pantalla es "A>".
Respuesta: A.

2.° ¿Qué indica a CP/M la orden B?:
Respuesta: Que cambie la unidad de disco por defecto a la unidad.

3.° ¿Cuál será la impronta tras la orden anterior?
Respuesta: B>.

4.° ¿Es correcta esta orden?: Ñ:
Respuesta: No, porque la Ñ no es un caracter estándar ASCII y, además, y en todo caso, la versión 20 sólo permite en línea 16 unidades de disco.

5.° ¿Cómo interpreta esta orden?: A B: PRUEBA. UNO.
Respuesta: Siendo A la unidad por defecto, cargar en memoria el programa PRUEBA. UNO desde la unidad B.

REFERENTE AL ALMACENAMIENTO EN DISCO DE UN FICHERO

Sabemos que, de forma genérica, un fichero es un grupo de datos relacionados entre sí, y que estos datos pueden ser almacenados en un disco. Centrándonos en este soporte, puede darse el caso de que un fichero no contenga ni un solo dato y, en el otro extremo, nada impide que los datos de un fichero ocupen toda la capacidad de un disco.

Estudiemos la forma en que se produce el almacenamiento de los datos de un fichero en un disco. Este es el tema que nos ocupará en el presente epígrafe.

En primer lugar, recordemos que la transferencia de información desde la RAM al disco, se hace sencuencialmente, partiendo de la primera posición de memoria y hasta la última, pero en lotes de *bytes* equivalentes a un sector, regulado por un *buffer.*

Esto es cierto desde un punto de vista físico por las razones expuestas en su momento.

No obstante, con el fin de aumentar la eficacia de las transmisiones y la operatividad del sistema en su conjunto, la grabación de los *bytes* de información correspondientes a un fichero se realiza de acuerdo con ciertas normas.

Un fichero ocupará como mínimo un predeterminado espacio del disco denominado *tamaño del bloque,* que corresponderá a un número preciso de sectores.

El tamaño del bloque suele ser de ocho a dieciséis sectores. En lo que sigue, admitiremos que el tamaño del bloque es de ocho sectores, y que el sector es de 128 *bytes,* un fichero con menos de 1.024 *bytes* de información ocupará un bloque completo (8 sector < > 1.024 bytes) en el disco.

Ya conocemos el límite inferior de ocupación de disco por parte de un fichero. Pero, ¿qué sucede si un fichero requiere más de un bloque?

Sencillamente, irá ocupando bloques sucesivos, hasta completar la grabación en unidades completas del bloque. En nuestro supuesto, grupos de 1.024 bytes.

Es evidente que, para ficheros que requieran menos de un bloque, la capacidad de almacenamiento de un disco está subexplotada, pero, a cambio, el acceso a la información de cada fichero mejora indudablemente.

Por otra parte, para ficheros que requieran más de un bloque el aprovechamiento es aceptablemente alto, ya que sólo es dudoso el porcentaje de ocupación del último bloque.

Hagamos un alto y recordemos que cada fichero queda individualizado por su nombre, gracias al cual se le distingue de cualquier otro. También se hizo referencia a que cada fichero queda registrado, en la pista catálogo mediante un grupo de 32 *bytes* que reflejan todos los

datos necesarios que para el DOS pueda controlarlos.

Pues bien, ésto es cierto mientras el fichero no necesite más de un número predeterminado de bloques, para su almacenamiento. A este núnero de bloques se le conoce por *extensión.*

Dependiendo del ordenador, la extensión es, generalmente, de ocho a dieciséis bloques.

Por cada extensión que requiera un fichero se producirá una nueva entrada de 32 *bytes* en la pista catálogo para hacer referencia a la misma.

Por tanto, en base a todo lo expuesto, de suponer un fichero que ocupara la totalidad de un disco, diremos que el disco almacena un solo fichero, representado por su nombre, pero, en la pista catálogo aparecerán tantas entradas como extensiones (o grupos de bloques) permita la versión de CP/M implementada al ordenador:

Un ejemplo servirá para fijar las ideas.

Con los conceptos de sector, bloque y extensión en nuestro haber, analicemos un disco cuyas características resumidas, sean las que siguen:

— 40 pistas.
— 9 sectores por pista.
— 512 bytes por sector.
— 2 sectores por bloque.
— 16 bloques por extensión.
— 64 ficheros catalogables como máximo.

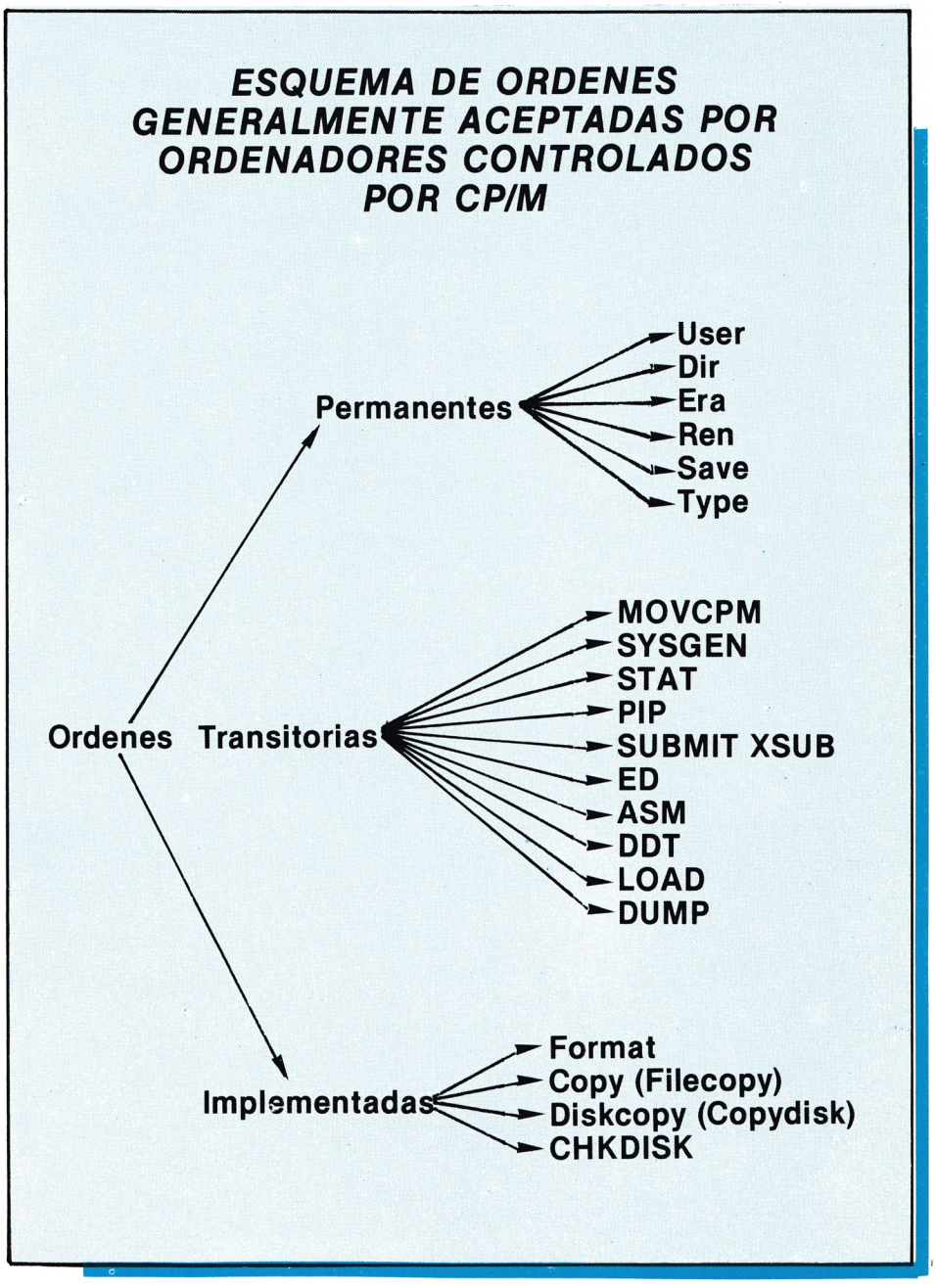
Es claro e inmediato, en función, de lo dicho, que nuestro disco permite manejar sectores de 512 *bytes,* el tamaño del bloque es de 1.024 *bytes* (2 sectores) y la extensión requiere 16.384 *bytes* (16 K<>1 bloque).

En cada cara se podrán almacenar 184.320 *bytes* (180 K), distribuidos en cuarenta pistas a razón de 9 sectores por pista o, lo que es igual, 180 K, grabados en 360 sectores.

El número máximo de ficheros permitidos es de 64, ya que la pista catálogo sólo admite esta cantidad, pero por cada extensión adicional que necesite un fichero disminuirá en una unidad este límite.

La ocupación mínima de disco requerida por un fichero, por pequeño que éste sea, es de 1.024 *bytes* (un bloque)

Si el disco contuviera las rutinas del DOS, éstas, quedarían almacenadas en las dos pistas exteriores, con lo cual la capacidad del disco para datos quedaría disminuida a 1.71 K. Este punto se verá con más detalle posteriormente.



REFERENTE A LAS ORDENES PERMANENTES

En el esquema anterior se han resumido los distintos tipos de órdenes que CP/M puede aceptar: Permanentes, transitorias e implementadas.

Las órdenes permanentes *(built in commands)* también son denominadas "residentes", y lo son por pertenecer a un subconjunto de programas de CP/M situado en el CCP y consiguientemente, tras la carga del DOS en la memoria interna, están ubicadas en la RAM de acuerdo con lo expuesto en el epígrafe "UBICACION DE CP/M". En otras palabras, las órdenes permanentes están a disposición inmediata del operador.

Con objeto de fijar esta idea, y an-

ticipando parte de la explicación que exigen las órdenes transitorias, diremos que la diferencia operativa que existe entre éstas y aquéllas se deriva del hecho de que CP/M para ejecutar una orden de las llamadas transitorias necesita cargar del disco en la RAM el programa correspondiente a la orden en cuestión.

Dicho esto, concluiremos con un resumen de las órdenes permanentes que CP/M interpreta y que están situadas en el CCP:

```
USER
DIR
ERA
REN
SAVE
TYPE
```

A su estudio individualizado se dedican los siguientes epígrafes:

USER

Función asignada: Asignar un **área de usuario** mediante un número comprendido entre 0 y 15, ambos inclusive, con lo cual todo el tratamiento y manejo de ficheros se referirá automáticamente a los situados en el área de usuario especificada, ignorando todos los asignados a diferentes áreas.

Esta orden está disponible en CP/M-86 y CP/M-80 versiones 2.0 y sucesivas.

Forma de obtenerla: Tecleando USER seguido de un número entre 0 y 15, ambos inclusive, representativo del área de usuario en el que se desea trabajar.

Ejemplo: Para trabajar, en el área de usuario número 2 daríamos la siguiente orden:

A >USER 2

Al pusar<cr>para hacerla efectiva, la pantalla simplemente nos devolverá la impronta de la unidad de disco por defecto (A >en este caso) indicando con ésto que la orden fue cumplida y el efecto subsecuente asumido.

Comentarios: Las consecuencias prácticas de esta orden se derivan del hecho de capacitar al operador a generar 16 distintos directorios dentro de la pista catálogo y de tal forma que aquellos ficheros que hayan sido creados y nominados en un disco estando activa un área de usuario determinado, simplemente no existirán para CP/M al cambiar de área de usuario.

Un área de usuario se mantiene en activo, hasta que una nueva orden USER lo cambia o se inicializa CP/M.

Al inicializar CP/M se sitúa en el área de usuario número 0.

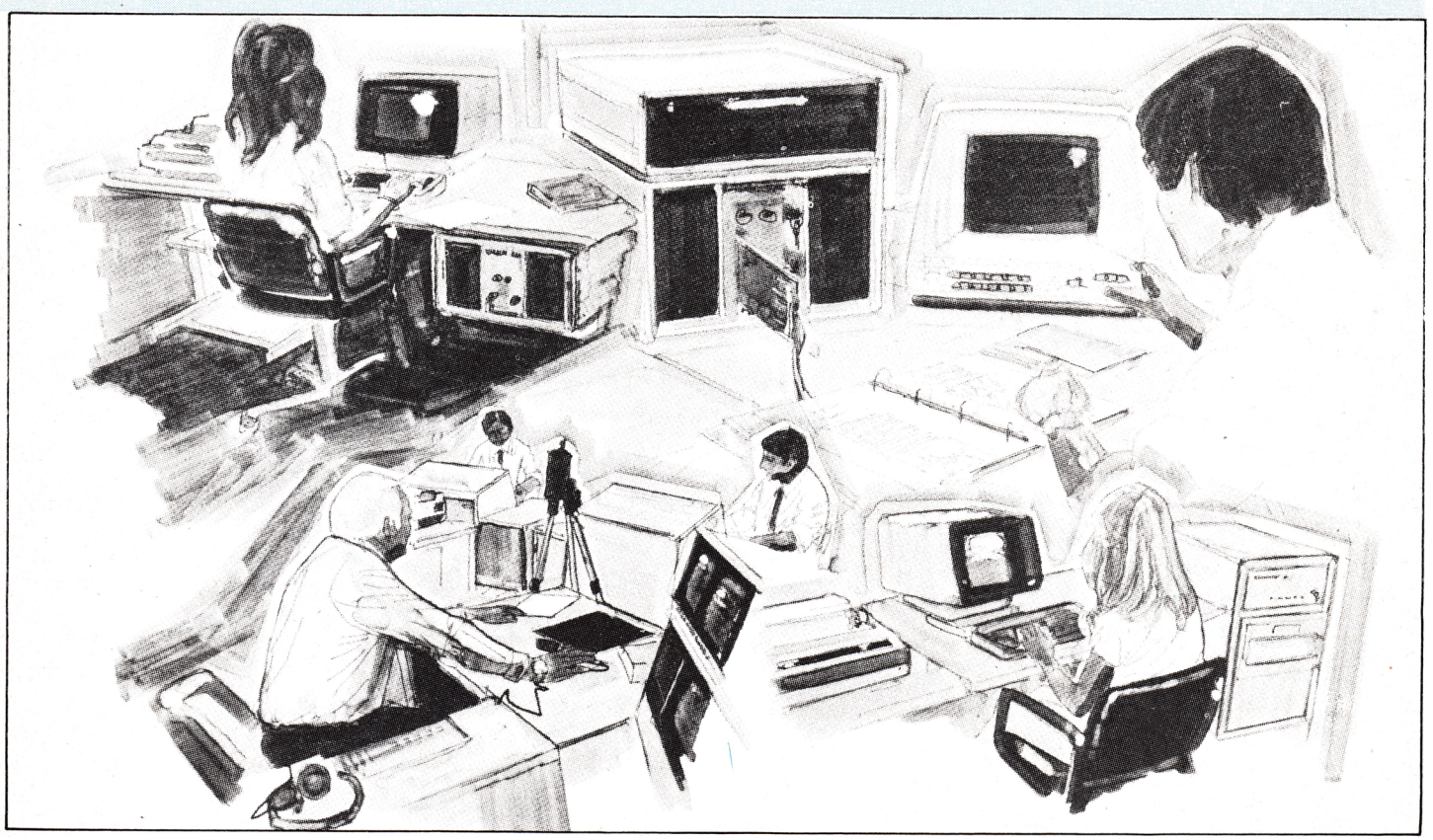
Al directorio de la versión 1.4 de CP/M-80 es compatible con el directorio del área de usuario 0 de la versión 2.0.

Los ficheros creados estando activo un determinado número de área de usuario, "existirán" sólo cuando esté activo este área de usuario. En otras palabras, al catalogar un nuevo fichero en la pista catálogo se le incluye y asocia el número de área de usuario en la que se trabaja en el momento de almacenarlo en el disco.

Por lo dicho, es fácil asumir que una orden del tipo DIR sólo mostrará el listado de los ficheros creados con el número de área de usuario que a la sazón esté activado.

Igualmente, un ERA ** borrará, exclusivamente, los ficheros generados con el número de área de usuario actual.

Por lo demás, toda referencia a ficheros se asume que es a los del área de usuario en la que se trabaja en ese momento.



DIR

Función asignada: Mostrar una lista de todos o parte de los ficheros correspondientes al área de usuario actual contenidos en un disco situado en una unidad de disco específica. Dicho de otra forma, DIR retorna un directorio o catálogo de los ficheros referenciados del área de usuario activada.

Forma de obtenerlo: Con CP/M a nivel de operador teclear DIR seguido de la referencia a los ficheros —o fichero— de cuyos nombres se quiere obtener el listado, concluyendo la orden con<cr>

Si DIR no va seguido de referencia alguna a fichero, entonces el listado mostrará todos los nombres de ficheros contenidos en el área actual de usuario del disco en cuestión.

Ejemplos:

1.° Supongamos que se desea averiguar todos los ficheros contenidos en un determinado disco. Para ello, lo instalaremos en la unidad de disco por defecto —A, por ejemplo— y, en estas condiciones, damos la siguiente orden:

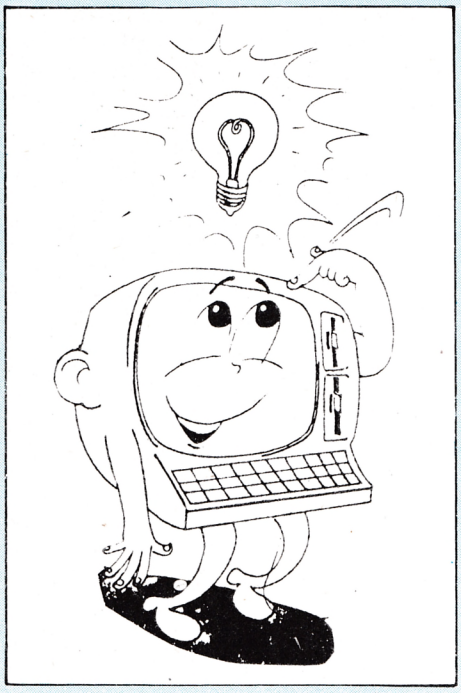
A > DIR

Al pusar<cr>para hacer efectiva la orden, veremos parpadear por unos instantes el piloto de la unidad de disco, y de inmediato aparecerá la lista completa de los nombres de los ficheros almacenados en el disco. Por ejemplo:

```
A : PRUEBA REN : EJEMPLO SEI :
EJEMPLO SIE PROG SIE
A : LIBROS DOS : PROGLI DOS : LI-
BROS PRG : PRUEBA SAV
A : BASIC 1 : PROG DOS : LIBROS
BAK : LIBROS PRD
A : LIBROS SEC : BASIC 2 : BASIC 3
```

Para obrener el directorio de un área de usuario distinta a la presente, debemos comenzar por dar la orden USER n según se explicó en el epígrafe anterior, seguido de un DIR como el de este ejemplo.

En otro orden de cosas, y con la intención de completar la explicación dada en su momento respecto a P, veamos cómo se



puede obtener el listado anterior a través de la impresora, si el ordenador en uso lo permite. Con CP/M a nivel de operador, damos un P (CONTROL/P), con lo cual activamos la salida por impresora, y a continuación se ordena un DIR según lo anteriormente explicado. A partir de este momento irán imprimiéndose en pantalla y papel los nombres de los ficheros del disco.

Para desactivar la salida por impresora bastará pulsar CONTROL y P nuevamente.

2.° Si por cualquier razón deseamos obtener el directorio de nuestro disco en una unidad de disco —la B, por ejemplo— distinta a la que a la sazón sea la omitida, lo colocaríamos en la unidad elegida —la B— y, a continuación, daremos la siguiente orden:

```
A > B: DIR
```

Con lo cual conseguiremos nuestro actual objetivo.

3.° Si sobre el mismo disco y en la unidad A nos interesa averiguar exclusivamente cuántos ficheros son del tipo DOS, la orden será:

```
A > DIR * .DOS
```

Con lo cual, al ser ejecutada, tendríamos:

```
A: LIBROS .DOS: PROGLJ
.DOS: PROG .DOS
```

4.° Este nuevo ejemplo consiste en obtener todos los ficheros cuyo nombre es LIBROS, sea cual sea su tipo. Para ello, escribiremos:

```
A>DIR LIBROS.*
Siendo el listado subsecuente:
A : LIBROS .DOS: LIBROS
.PROG.: LIBROS .BAK: LIBROS
.PRD A: LIBROS .SEC
```

5.° Para concluir esta serie de ejercicios, utilizaremosa el comodín?, para localizar todos los ficheros cuyos nombres estén compuestos por cinco caracteres cualesquiera y sus tipos sólo tengan un caracter, sin precisar. La orden sería:

```
A > DIR??????
Y la respuesta:
A: BASIC.1: BASIC. 2: BASIC. 3
```

Observe que un DIR *.* es equivalente a un DIR, ya que en ambos casos nos estamos refiriendo a todos los ficheros.

Recuerde que todos los ejemplos anteriores son válidos para cualquier área de usuarios, con la única condición de dar la oportuna orden USER *n,* donde *n* es el número del área de usuario que se desea activar (ver USER).

Comentarios: Cuando CP/M nos muestra su impronta indicativa de que está a nivel operador (por ejemplo, A), ignora qué disco hemos instalado en una determinada unidad de disco; por esta razón, la primera acción que llevará a efecto tras recibir una orden que implique la manipulación del disco, será "tomar contacto" con él, con lo cual averiguará si efectivamente hay un disco colocado en la unidad indicada y, en tal caso, lo catalogará. A este proceso se le conoce en inglés por *log in.*

Esta operación la puede llevar a efecto el DOS gracias a la información contenida en la pista catálogo o directorio de la cual ya se dio razón en capítulo dedicado a "DISCOS", donde se advertía que al almacenar en un disco un nuevo fichero se grababa, en la pista catálogo una secuencia de treinta y dos *bytes.* Cada uno de estos **bytes** cumple la siguiente función:

El primero actúa como un semá-

foro que avisa al DOS que el fichero en cuestión puede ser leído. Una instrucción posterior de borrado de este fichero del disco haría cambiar el valor de este *byte,* con lo cual (CP/M consideraría que no existe.

Los ocho siguientes representarán los caracteres ASCII del nombre del fichero de izquierda a derecha. Si el nombre no requiere de los ocho bytes, los sobrantes a la derecha representarán códigos del espacio (32 decimal, 10 hexadecimal).

Los tres siguientes se refieren al tipo del fichero, en las mismas condiciones de los ocho anteriores.

El decimotercero indica el número de *extensión* actual dentro del fichero en cuestión.

A continuación, CP/M se reserva dos *bytes.*

El decimosexto contiene la medida de la *extensión* expresada en sectores.

Los restantes *bytes* indican la situación física dentro del disco de los diferentes bloques que ha requerido el fichero para ser almacenado.

ERA

Función asignada: Borrar todos o parte de los ficheros contenidos en un disco instalado en la unidad de disco especificada. Los ficheros a borrar dependerán de la referencia que se haga a ellos y del área de usuario que esté activada.

Forma de obtenerlo: Con CP/M a nivel de operador teclear ERA seguido de la referencia al fichero —o ficheros— a borrar, concluyendo la orden con<cr> Si el área de usuario correspondiente a los ficheros interesados es distinta a la actual se debe dar, previamente la orden USER *n* oportuna (ver USER).

Ejemplo:

1.º Para borrar el fichero LIBROS .DOS de un disco instalado en la unidad de disco por defecto —la A, por ejemplo—, daremos las siguiente orden:

```
A> ERA LIBROS .DOS
```

Al pulsar<cr>para hacer efectiva la orden, y tras ser manipulado el disco, el fichero en cuestión habrá desaparecido del catálogo para CP/M. La operación se da por concluida al aparecer la impronta (A>) nuevamente. Con una orden DIR, según lo expuesto en el anterior epígrafe, se puede comprobar el nuevo directorio.

2 . ºPara borrar todos los ficheros del disco, recurrimos al comodín "*" de esta forma:

```
A > ERA *.*
```

Con lo cual conseguiríamos el fin propuesto.

3 . ºPara borrar todos los ficheros cuyo tipo sea DOS con una sola orden, escribiríamos:

```
A > ERA * DOS
```

Al pulsar <cr> se iniciaría la operación de borrado que concluirá con la aparición de la impronta (A>).

4 . ºPara borrar todos los ficheros de nombre LIBROS, actuaríamos de forma similar:

```
A > ERA LIBROS.*
```

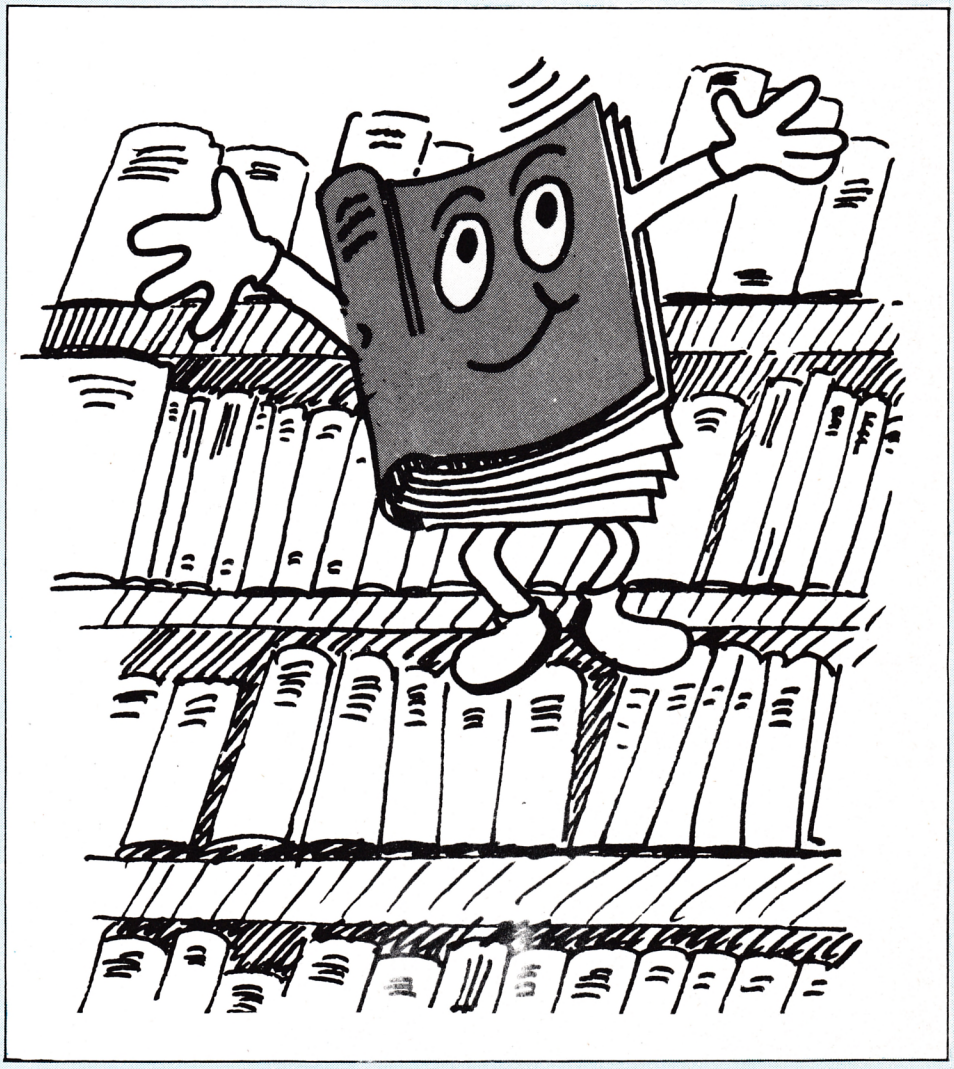
5 . ºPara borrar todos los ficheros cuyo nombre empiece por PRO y el tipo por S, podríamos usar el comodín "?" así:

```
A> ERA PRO ?????.S??
```

6 . ºPor último para borrar uno o varios ficheros de un disco situado en una unidad de disco distinta a la que actualmente está considerada por el DOS como unidad por defecto, utilizaremos el mismo criterio expuesto hasta el momento, pero anteponiendo la referencia oportuna a la unidad elegida. Por ejemplo:

```
A> ERA B: *.DOS
```

Con esta orden se borrarían todos los ficheros de tipo .DOS



del disco instalado en la unidad de disco B.

Comentarios: Con esta orden se borra sólo la información oportuna de la pista catálogo dedicada a los ficheros referenciados, con lo cual CP/M interpretará que el espacio ocupado por estos ficheros está disponible. Por el contrario, FORMAT, del cual ya se ha comentado algo, borra el disco entero y completamente.

REN

Función asignada: Cambia el nombre de un fichero contenido en un disco instalado en la unidad de disco especificada, debiéndose tener en cuenta las siguientes normas:

1.° Está prohibido el uso de comodines, siendo necesario, por tanto, referirse al fichero por su nombre sin ambigüedad (ufn).
2.° El nombre "nuevo" no debe coincidir con ninguno de los existentes en el disco. Si fuera así, el DOS enviaría un mensaje FILE EXISTS (existe fichero) y la orden quedaría cancelada, apareciendo la impronta nuevamente.
3.° La orden REN sólo actúa sobre la unidad de disco especificada (la omitida o la explicitada). En ningún caso sobre dos unidades de disco distintas.
4.° El nombre "viejo" debe existir en el área de usuario actual del disco, de otra norma un mensaje del tipo NO FILE o NOT FOUND aparecería con lo que en definitiva nos viene a decir que no existe el fichero.

Forma de obtenerla: Con CP/M a nivel de operador, teclear REN, un espacio, seguido de la referencia, sin ambigüedad, del nuevo nombre a asignar al fichero, a continuación el símbolo " = " y el nombre actual del fichero.

Ejemplos:

1.° Supongamos que en el directorio de un disco instalado en la unidad A (admitida por defecto) existe un fichero denominado EJEMPLO SEI y pretendemos cambiarlo de nombre, siendo este EJEMPLO SIE. Si, con estas premisas, diéramos esta orden:

```
A> REN EJEMPLO SIE = EJEMPLO SIE
```

Obtendríamos el mensaje FILE EXISTS, ya que hemos invertido el orden de los nombres: primero debe ir el nombre "nuevo" y después el "viejo".

2.° La orden correcta en el caso anterior es:

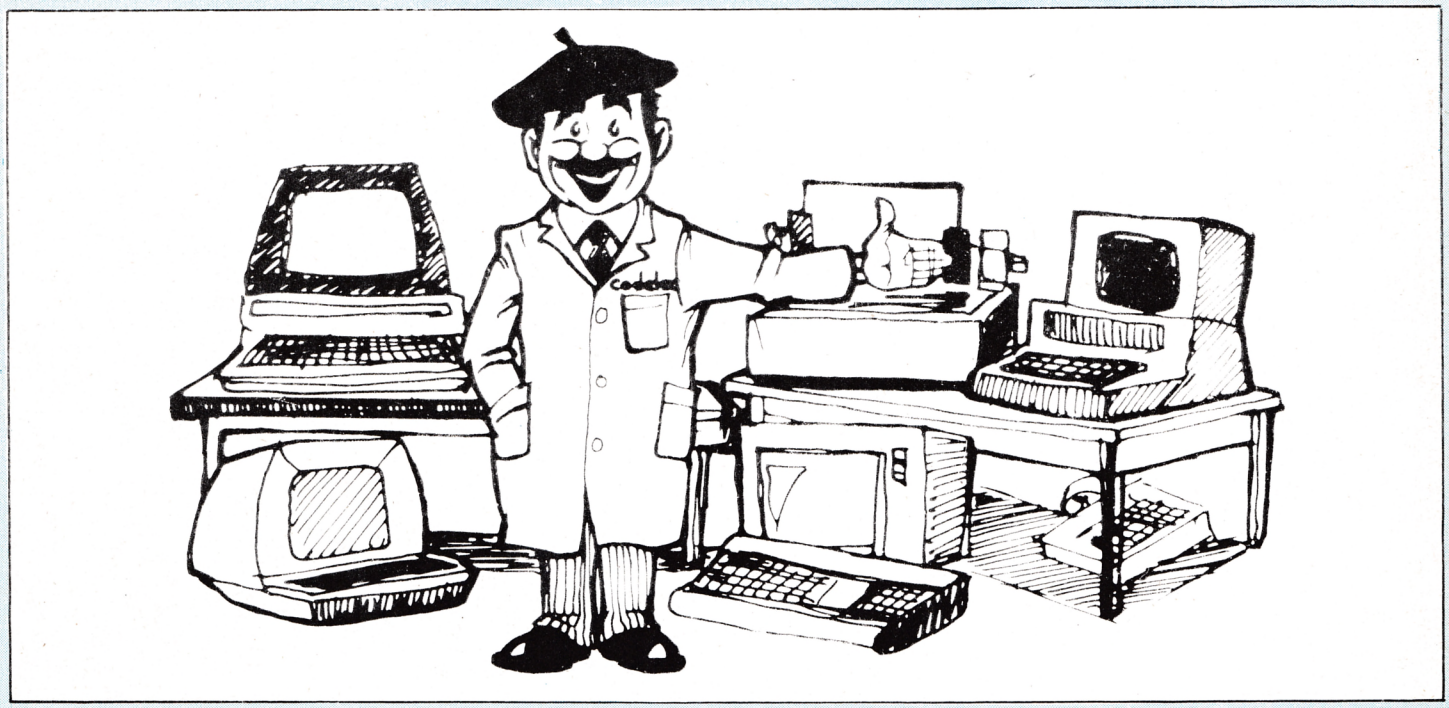
```
A> REM EJEMPLO SIE = EJEMPLO SEI
```

Con lo cual, una vez ejecutada la orden, habrá desaparecido el fichero nombrado EJEMPLO SEI y, en su lugar, tendremos EJEMPLO SIE
El contenido del directorio antes y después de la orden se puede comprobar con un DIR anterior y otro posterior a la misma.

3.° En este nuevo ejemplo asumiremos que el cambio de nombre se ha de producir en un disco instalado en una unidad de disco distinta a la considerada por defecto. La orden, en este caso, puede darse de una de estas tres formas:

```
A> REN B: NOMBRE NUE = NOMBRE VIE
o
A> REN NOMBRE NUE = B: NOMBRE VIE
o
A> REN B: NOMBRE NUE = B: NOMBRE VIE
```

Comentarios: Esta orden, como las dos anteriores, sólo actúa sobre la pista catálogo, y los ficheros situados en el área de usuario que a la sazón esté activada.



COMO CREAR FICHEROS EN DISCO

Por A. BELLIDO

En otro orden de cosas es interesante conocer un par de peculiaridades.

Por una parte, la velocidad de acceso a una determinada información contenida en disco depende más del tiempo invertido por el cabezal en su movimiento radial que de la velocidad de giro del disco.

Por otra, los sectores ocupados por una información no son consecutivos, pudiéndose dar el caso de que el primero esté situado en la pista 4 y sector 4 y el siguiente en el octavo sector de la misma pista.

Esto es así debido a que la unidad mínima de transferencia, como ya se dijo, entre computador y *drive* es equivalente a la capacidad en bytes de un sector, lo cual implica que antes de producirse la emisión de éstos se requiera un tiempo para llenar el *buffer* o depósito encargado de este trabajo, lo que, a la postre, lleva consigo que el disco se haya desplazado y el siguiente sector disponible no sea el consecutivo.

El *buffer* es una zona de memoria de dimensión predeterminada y ajustada a la función que ha de cumplir.

En nuestro caso, la capacidad del *buffer* corresponde a la de un sector, de forma y manera que la información sea transmitida por grupos de bytes convenidos entre la memoria receptora o emisora y la memoria o receptora la RAM y el disco.

El buffer, en todo caso, actúa como un depósito de bytes que al llenarse procede a actuar del modo que se haya previsto.

La misión fundamental del *buffer* es aumentar la velocidad de transferencia de la información.

Consideremos, en este sentido, que el tiempo consumido en localizar espacio disponible en el disco para el almacenamiento de una información es mucho más elevado que el necesario para transferir esa información de la CPU al disco. Consiguientemente, la operatividad general del sistema gana en eficacia si la transmisión se efectúa en bloques de *bytes* en lugar de *byte* en lugar de *byte* a *byte*.

RESUMEN DEL PROCESO DE FORMATEADO

A) Da forma, divide en pistas y sectores, la superficie del disco.

B) Inicializa todos los *bytes* de control de cada sector.

C) Inicializa la pista catálogo con todos los datos necesarios al DOS para manipular el disco.

D) Borra toda la información previa que el disco pudiera contener.

E) El comando FORMAT formatea el disco reservando, según los casos, las pistas 0 y 1 para el DOS.

FICHEROS. ORGANIZACION

En su momento se dijo que un fichero es un grupo de datos relacionados entre sí, y también se apuntó que el contenido de un fichero depende de la voluntad del usuario.

Ahora nos aproximaremos a la forma en que el programador puede organizar los datos contenidos en un fichero. Y al léxico utilizado.

Para ello, supongamos que se ha decidido controlar por medio de fichas los libros de una biblioteca en función de tres datos: Título, autor y editor.

Las fichas, de cartulina, han sido diseñadas con tres reglones preimpresos, en los cuales se escribirán los datos citados:

Título: _____
Autor: _____
Editor: _____

Para crear el fichero, tendremos que tomar la caja que ha de contener las fichas y colocar en su frontal el título del fichero, para distinguirlo de otros que pueda haber y destinados a otro fin.

Al nuestro lo llamaremos LIBROS.

Con estos precedentes, tomaremos una ficha en blanco y el primer libro cuyos datos se vayan a plasmar en aquella.

Este libro resulta ser "El mus", de Mingote, editado por Prensa Española.

En este punto hagamos una pequeña recapitulación.

Desde que hemos comenzado el ejercicio han aparecido:

—Tres datos "El mus", Mingote y Prensa Española.
— Una ficha: que contendrá los datos anteriores.
— Tres apartados en la ficha: En cada uno de ellos se escribirá el dato correspondiente.
— Un fichero: De nombres LIBROS, en el que se guardarán la ficha presente y el resto de las necesarias para controlar la biblioteca en cuestión.

Expresemos esto mismo en términos informáticos y por idéntico orden:

— Tres datos: "El mus", Mingote y Prensa Española.
— Un registro: Que contendrá los datos anteriores.
— Tres campos: En cada uno de ellos se grabará el dato correspondiente.
— Un fichero: De nombre LIBROS en el que se almacenarán el registro presente y el resto de los necesarios para controlar la biblioteca en cuestión.

Se pueden observar dos cambios. El concepto transmitido por la palabra "ficha" se traspasa al vocablo "registro", y el abarcado por "apartados en la ficha", a "campos".

Veamos estas nuevas ideas mediante un ejemplo cualquiera en una representación gráfica esquemática.

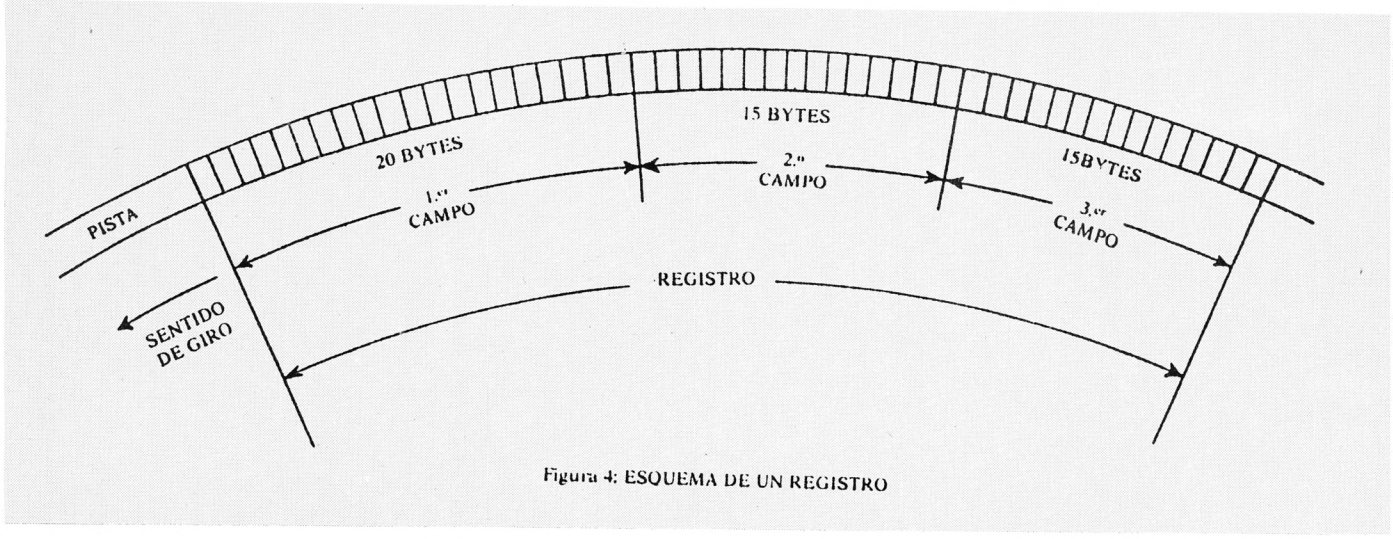


Figura 4: ESQUEMA DE UN REGISTRO

MATRICES

REVISION

El término inglés **array** expresa la idea de colocar en un orden preestablecido un grupo de elementos con un fin determinado. Más concretamente, desde un punto de vista matemático implica la organización de números u otros símbolos en filas y columnas, y el programador lo encontrará con frecuencia en ciertos mensajes del ordendor, en artículos y otros publicaciones.

El vocablo español equivalente más apropiado en **matriz** que es una de las acepciones, vale por: "Conjunto de número o símbolo algebraicos colocados en líneas horizontales y verticales y dispuestos en forma de rectángulo".

En términos informáticos una matriz es un conjunto de variables, numéricas o alfanuméricas, con el mismo nombre y subíndice acorde con la posición que ocupa en ella.

Cuando hablamos, p.e., de la matriz ELE o de ELE$ estamos haciendo referencia a una matriz cuyo nombre genérico es ELE y sus elementos en el primer caso, son numéricos y en el segundo alfanuméricos.

Esto mismo representado por medio de la escritura da lugar a ELE ( ) o ELE$ ( ), según el caso. Es decir, el nombre de la matriz seguido de un paréntesis vacío.

Cada una de las variables que integran la matriz es un elemento de la misma, y como tales variables pueden ser usadas con toda libertad.

Para usar matrices en un programa hay que empezar por dimensionarlas mediante la orden DIM. Por ejemplo:

DIM ELE (12). Con esta instrucción se crean trece variables numéricas de nombre ELE y subíndices correlativos del 0 al 12 de la siguiente forma:

ELE (0), ELE (1), ELE (2), ELE (3), ELE (4), ELE (5), ELE (6), ELE (7), ELE (8), ELE (9), ELE (10), ELE (11), ELE (12).

La instrucción DIM ELE (12) resulta en lo mismo pero con variables alfanuméricas, así: ELE$(0), ELE$(1), ELE$(2), ELE$(3) ELE$ (4), ELE$ (5), ELE$ (6), ELE$ (7), ELE$ (8), ELE$ (9), ELE$ (10), ELE$ (11), ELE$ (12).

Las siguientes notas resumen los puntos más importantes a tener en cuenta al trabajar con matrices, siempre teniendo en cuenta, claro está, las indicaciones del manual del BASIC que se esté utilizando:

NOTAS:

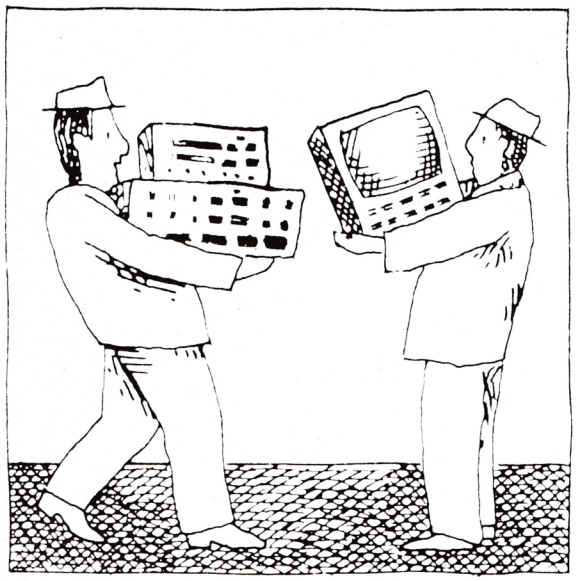
1.° El nombre genérico que designa a una matriz y a los elementos que comprende puede estar compuesto por cualquier combinación de letras y números siempre que el primer caracter sea una letra, y, para las matrices alfanuméricas, el último sea el símbolo "$".

2.° El subíndice más bajo que puede tener el elemento de una matriz es el 0.

3.° Mediante OPTION BASE n, siendo n 0 ó 1, se puede cambiar el subíndice más bajo a 0 ó 1 respectivamente. Esta instrucción, si se usa, ha de ser antes de dimensionar las matrices.
OPTION BASE no suele estar disponible en algunos BASIC.

4.° Hasta el subíndice 10, si se manejan elementos de una matriz que no ha sido dimensionada previamente se sobreentiende una matriz con el nombre dado a ese elemento y de subíndice mayor 10.

5.° Si se hace referencia a una variable cuyo subíndice sea mayor que el más alto indicado al dimensionar la matriz —ó 10 por defecto, según la nota anterior— se obtiene el error **"Subscript out of range"**, o cualquier otro que indique el error cometido con el subíndice.



6.° El mensaje **"Duplicate definition", "Array already dimensioned in"** u otros del mismo porte aparece siempre que se quiere dimensionar una misma matriz por segunda vez.

7.° Mediante la instrucción ERASE nombre matriz1, nombre matriz2,... Se borran las matrices indicadas.

8.° Nada impide utilizar una variable regular (sin subíndice) con el mismo nombre que las variables de una matriz.

9.° Los elementos de una matriz pueden organizarse según una sencuencia lineal de los subíndices —unidimensionales— o distribuyéndolos en varias dimensiones —multidimensionalmente—. Algunos BASIC permiten hasta 255 dimensiones.

10.° Cuando en un programa, o en modo directo, se ejecuta una orden DIM, todas las variables de la matriz en cuestión toman el valor inicial cero en el caso de las numéricas o la cadena vacía en el de las alfanuméricas.

11.° La orden RUN borra las matrices. Por tanto, una vez ejecutado un programa con RUN por primera vez, y si no se quiere perder la información contenida en las matrices, el programa deberá correrse mediante el mandato —en modo directo— GOTO seguido del número de la línea a partir del cual se le quiere expistar y, en todo caso, posterior a las sentencias DIM que se quieren preservar.

ASIGNACION DE VALORES A LAS VARIABLES CON SUBINDICE

Como quedó dicho en la última nota, tras un orden DIM las variables de la matriz afectada están pendientes de recibir valores. Existen tres métodos fundamentales: Mediante IMPUT, mediante READ/DATA y mediante el volcado de una matriz en otra. Veamos algunos listados que puedan servir de prototipos de los tres casos:

Mediante INPUT:

```
10 CLS
20 DIM L$ (14), P (14)
30 P = 0
40 INPUT "Libro"; (L$ (P)
50 INPUT "Precio"; P (P)
60 P = P + 1: IF P   I   = 14 = THEN 40
70 END
```

Comentarios al listado anterior:

En la línea 20 se dimensionan dos matrices una alfanumérica —L$( )— y otra numérica —P( )— ambas de 15 elementos: Del L$(0) al L$(14).

En la 30, se inicializa la variable que actuará de controlador gracias al cual los subíndices irán tomando valores. Se ha introducido deliberadamente esta variable —P— con el objeto de dejar claro que pueden subsistir una variable sencilla con el mismo nombre que el correspondiente a las variables con subíndice de una matriz, sin que ello provoque confusión en el BASIC.

En la línea 40, se permite la entrada por teclado a una cadena de caracteres que dará contenido a la variable de nombre genérico L$(P).

En la línea 50 se permite la entrada por teclado a un número que dará valor a la variable numérica P(P).

En la línea 5C aumenta el contador en una unidad con lo cual se actualiza la variable que da valor a los subíndices, comprobándose a continuación si éstos superan el número más alto permitido por la matriz.

Mediante READ/DATA:

```
10 CLS
20 DIM L$(14), P(14)
30 P = 0
40 READ L$(P), P(P)
50 P = P + 1: IF P    = 14 THEN 40
60:
1000) DATA EL QUIJOTE, 1200, LA DIVINA COMEDIA,
900, FAUSTO, 950, OTELO, 750, HAMLET, 750, CALIX-
TO Y MELIBEA, 600, LA GALATEA, 500 EL LAZARILLO
DE TORMES, 600, LAS ALEGRES COMADRES DE
WINDSOR, 700, FUENTEOVEJUNA, 800, EL MEJOR
ALCALDE, 800, EL CABALLERO DE OLMEDO, 800, PE-
RIBAÑEZ, 800, EL BUSCON, 800
```

Comentarios al listado anterior:

Al igual que en el caso anterior, en la línea 20, se dimensionan las matrices L$( ) y P( ), y en la L40 se inicializa el contador P.

En la 40 se organiza la lectura de los datos contenidos en el DATA de la línea 1100, según lo cual las matrices L$( ) y P( ) irán tomando un título y un precio conforme aumenta el contador.

En la línea 50 se efectúan la actualización del contador y la comprobación de borde oportuna.

Mediante el volcado de una matriz en otra:

Para estudiar esta alternativa suponemos que el listado se inicia en la línea 10 del programa del caso anterior, con lo cual la matriz L$( ) ha sido dimensionada y sus elementos valorados.

```
60:
70 DIM R$(14)
80 FOR X =0 to 14
90 R$(X) = L$(X)
100 NEXT X
```

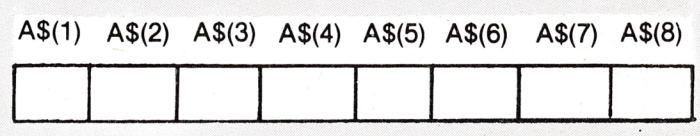
Comentarios al listado anterior:

Tras la ejecución de estas cuatro líneas, los elementos de la matriz R$( ) han recibido la asignación del contenido de los elementos correspondientes de la matriz L$( ).

Tras este breve repaso a las matrices, veamos algunas manipulaciones usuales realizadas sobre ellas.

DESLIZAMIENTOS Y ROTACIONES

Para estudiar los dos conceptos comprendidos en el encabezamiento, vamos a considerar una matriz de nombre A$ y ocho elementos que representaremos así:



Según este gráfico, el contenido de los elementos de la matriz A$( ) es la cadena vacía.

Para dar a entender que A$(1) es igual a "A", A$(2) igual a "B", A$(3) igual a "C", etc., se representará de esta forma:

A$(1)	A$(2)	A$(3)	A$(4)	A$(5)	A$(6)	A$(7)	A$(8)
A	B	C	D	E	F	G	H

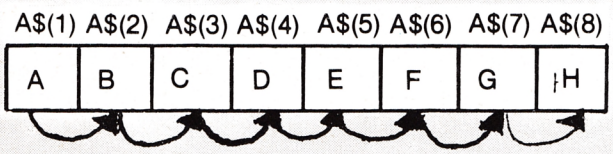
Hechas estas prevenciones, vamos a ver en que consiste un deslizamiento.

DESLIZAMIENTOS

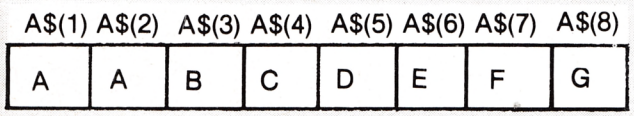
Cuando, por las razones que sean, interesa desplazar el contenido de cada elemento de una matriz a su inmediatamente contiguo, bien a derecha bien a izquierda, se da lugar a un deslizamiento. Esto mismo representado gráficamente, da lugar a la siguiente serie de alternativas:

Deslizamiento a la derecha

Antes del deslizamiento:



Después del deslizamiento:

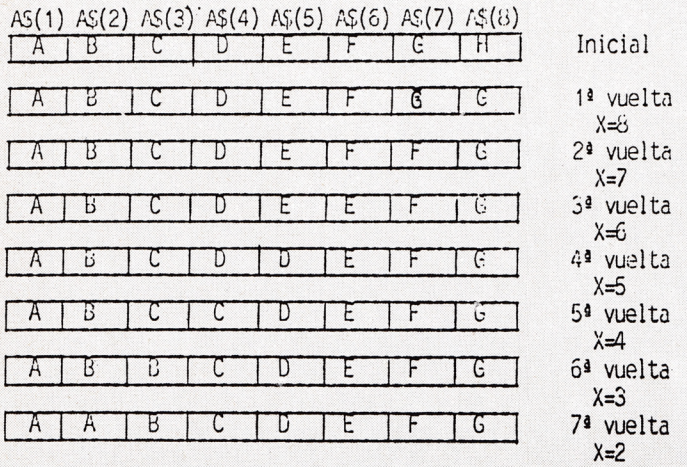


Como se puede observar, en este tipo de deslizamiento a la derecha se pierde el contenido de la variable situada más a la derecha, que equivale a la de subíndice más alto.

El programa que hace posible un deslizamiento a la derecha, en estructura y considerando dimensionada la matriz y valorados sus elementos, es el siguiente:

```
100 FOR X = 3 TO STEP-1
110 A$ (X) = A$(X-1)
120 NEXT X
```

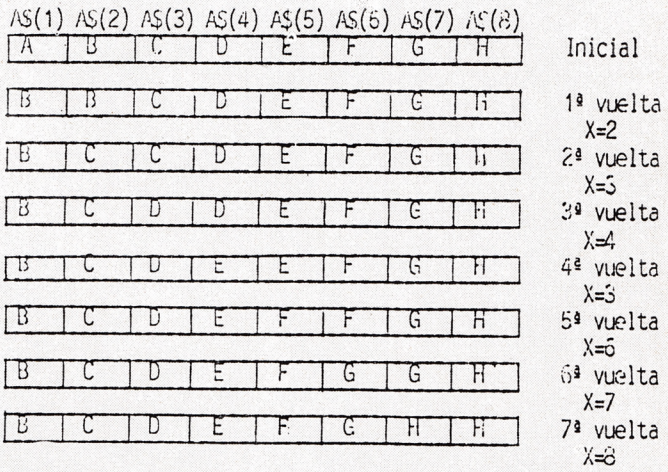
La ejecución de este programa da lugar a la serie de desplazamiento que se indican en la figura, resultando finalmente en el deslizamiento deseado.



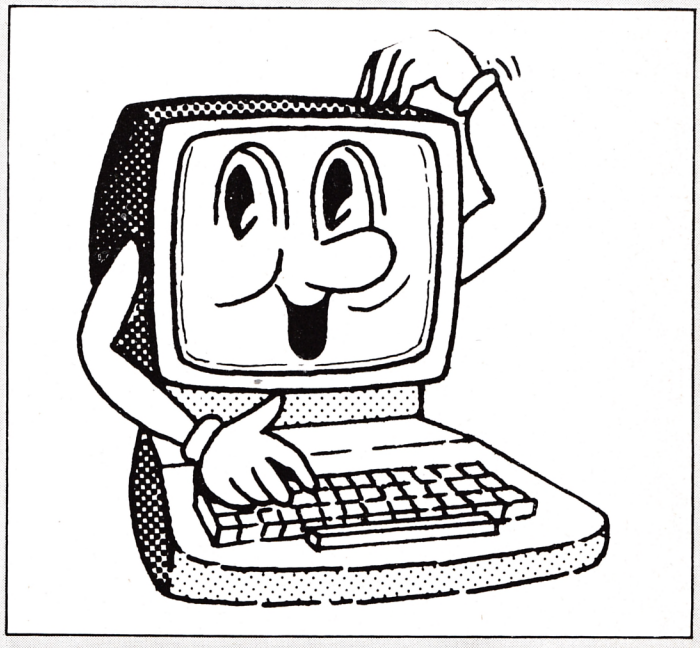
Para conseguir este mismo efecto, pero a la izquierda:

```
100 FOR X = 2 TO 8
110 A$ (X-1) = A$ (X)
120 NEXT
```

La ejecución de este programa da lugar a la serie de desplazamientos que se indican en la figura resultando finalmente en el desplazamiento deseado.



Para evitar la pérdida del contenido de cualquiera de las variables más extremas, izquierda o derecha, según el tipo de deslizamiento, bastaría asignar previamente ese valor a una variable auxiliar. Esto se puede representar de la siguiente manera:



EL BASIC DE

Por A. B

CONFIGURACION DE LOS CAPITULOS

En los capítulos siguientes se estudia el vocabulario BASIC más usual.
Cada capítulo responde a la siguiente estructura:

PALABRA	SINTAXIS DE LA PALABRA	TRADUCCION

INTERPRETACION — **Explica el uso general de la palabra**

POSIBILIDADES — **Explica las diferentes posibilidades de utilización**

FORMA DE TECLEAR LA INSTRUCCION

EJEMPLOS — **Para aclarar el campo de utilización**

EJERCICIOS — **Sugiere y/o plantea ejercicios para el aprendizaje.**

Antes de que inicie el estudio del primer capítulo que le ayudará a introducirse en el lenguaje de programación BASIC, permítanos darle...

Un consejo: Si no tiene microordenador, procure proveerse de uno. Sin él tendrá dificultades para asimilar cuanto a continuación se expone.

Otro consejo: Siempre que piense qué podría suceder si hace tal o cual cosa, ¡NO LO DUDE Y HAGALA!, ésta es la mejor vía para aprender.

Y un consejo final: Provéase de un cuaderno y escriba en él todos los ejercicios que realice. Le será muy útil a la hora de confeccionar sus propios programas.

ELLIDO

PRINT	PRINT EXPRESIONES	IMPRIMIR

INTERPRETACION:

Significa *''imprimir''* y ordena la impresión en pantalla de la expresión o expresiones que figuran, optativamente, en las expresiones.

POSIBILIDADES:

* Si las expresiones son emitidas, una línea de pantalla salta en blanco.

** Si las expresiones son incluidas y estas expresiones son cadenas de caracteres, debemos escribirlas entrecomilladas.

*** Si las expresiones son incluidas y estas expresiones son matemáticas, aparecerán en pantalla sus valores. Recuerde que las expresiones matemáticas no se escriben entre comillas.

La posición de impresión de cada expresión de expresiones viene determinada por los símbolos que separan una expresión de otra:

— Un punto y coma (;) hace que la siguiente expresión se imprima justo a continuación de la inmediata anterior.

— Una coma (,) obliga a imprimir a partir de la siguiente zona de tabulación o al comienzo de la línea siguiente si ya se ocuparon todas las zonas.

Si las expresiones finalizan con una (,) o un punto y coma (;), la siguiente instrucción PRINT, que pudiera aparecer a lo largo del programa, comenzaría la impresión de sus expresiones conforme al criterio expuesto más arriba.

En caso de no acabar con una coma o un punto y coma, la siguiente instrucción PRINT comenzaría la impresión de sus expresiones en la siguiente línea de pantalla que esté libre.

FORMA DE TECLEAR LA INSTRUCCION

Teclear PRINT, seguido de ENTER, para dejar en blanco una línea en la pantalla.

Teclear PRINT y la expresión entre comillas ("), seguido de ENTER para que aparezca en pantalla *la expresión alfanumérica.*

Teclear PRINT y la expresión, seguido de ENTER, para que aparezca en pantalla *la expresión numérica.*

EJEMPLO:

1. Escriba un programa que imprima en pantalla la cadena de caracteres: **"La loba lamía, lentamente, los lobeznos 3 veces".**

SOLUCION:

PROGRAMA	COMENTARIOS
10 PRINT "La Loba"	La forma de teclear esta línea de programa, respetando las normas dictadas por cada MANUAL, sería: • Teclear 10 • Teclear PRINT • Teclar "(comillas) • Escribir el texto. • Dar un espacio. • Teclar "(comillas). • Apretar ENTER.

Una vez seguidas las instrucciones dadas en COMENTARIOS, la línea número 10 de nuestro primer programa estará en la memoria del ordenador. En este caso, tenemos un programa, compuesto por una sola línea, cuya misión es imprimir el texto indicado cada vez que se le ordene al computador.

De una forma similar iríamos tecleando las siguientes líneas del programa:

PROGRAMA	COMENTARIOS
20 PRINT "lamía, lentamente"; 30 PRINT "los lobeznos"; 40 PRINT "3 veces"	• Teclear 20 • Teclear PRINT • Teclear "(comillas) • Escribir el texto

- Dar un espacio
- Teclear " (comillas)
- Teclear; (punto y coma)
- Apretar la tecla ENTER

- Teclear 30
- Teclear PRINT
- Teclear "
- Escribir el texto
- Dar un espacio
- Teclear "
- Teclear ;
- Apretar ENTER

- Teclear 40
- Teclear PRINT
- Teclear
- Escribir el texto
- Teclear "
- Apretar ENTER

Para ordenar al computador que EJECUTE el programa que tiene en memoria, el BASIC dispone del comando RUN.

Así, en el caso que nos ocupa, si tecleamos RUN y, a continuación, ENTER, inmediatamente aparecerá en la pantalla:

La loba lamía, lentamente, los lobeznos 3 veces.

Una vez en este punto, es conveniente observar que la definición del ejemplo no fue del todo correcta, ya que decíamos: "Escriba un programa que imprima en...", cuando realmente deberíamos haber dicho: "Escriba un programa que, una vez ejecutado, determine la impresión en pantalla de..."

Esta precisión tiene por objeto fijar las ideas sobre las diferentes expresiones y conceptos que estamos manejando. Pasemos ahora a otros ejemplos.

EJEMPLO:

2. Escriba un programa que, una vez ejecutado, imprima en la pantalla la palabra **"Juan"**, que deje la siguiente línea en blanco e imprima en la siguiente **"excelente"**.

SOLUCION:

PROGRAMA	COMENTARIOS
10 PRINT "Juan" **20 PRINT** **30 PRINT "excelente"**	La única novedad de este programa de tres líneas se presenta en la 20, con el comando PRINT sin ninguna expresión a continuación. Al ser ejecutado el programa, dejará una línea en blanco entre el anterior PRINT y el siguiente: Ejecute el programa con RUN y ENTER, como ya sabe.

EJEMPLO:

3. Escriba el programa que, una vez ejecutado, imprima al comienzo de una línea de pantalla "MUY" y, en la siguiente zona de esta línea, "BIEN".

SOLUCION:

PROGRAMA	COMENTARIOS
10 PRINT "MUY", "BIEN"	La coma introducida entre las dos cadenas entrecomilladas hará que la segunda cadena comience su impresión al principio de la siguiente zona de la misma línea. Ejecute el programa.

EJEMPLO:

4. ¿Cómo se producirá la impresión en pantalla de las cadenas de caracteres que figuran en el siguiente programa una vez ejecutado?

SOLUCION:

PROGRAMA	COMENTARIOS
10 PRINT "JUNTO" **20 PRINT "A TI"** **JUNTO A TI**	El punto y coma con que acabamos la primera instrucción PRINT obliga a la expresión PRINT siguiente a imprimirse justo a continuación de la anterior. Observe que se ha dejado deliberadamente un espacio al final de la primera cadena (línea 10).

EJEMPLO:

5. Escriba un programa que, una vez ejecutado, imprima en pantalla el resultado de sumar 2 y 2.

SOLUCION:

PROGRAMA	COMENTARIOS
10 PRINT 2 + 2	Como observará, la expresión que sigue al comando PRINT va sin comillas, ya que es una expresión matemática. Las comillas sólo se usan para indicar al ordenador que deseamos que una expresión sea tratada

como una cadena de caracteres.
En el caso que nos ocupa, ordenamos al computador que imprima el resultado de la operación indicada.

EJEMPLO:

6. Escriba un programa que, una vez ejecutado, imprima en pantalla: 2 + 2 *son 4.*

SOLUCIONES:

PROGRAMA	COMENTARIOS
10 PRINT "2 + 2 son"; **2 + 2** **o también** **10 PRINT "2 + 2 son ",** **20 PRINT 2 + 2**	La primera expresión, por ser entrecomillada, será tratada como una cadena de caracteres. Recuerde que, entre el último caracter de la cadena y las comillas, debe dejar un espacio. Así evitará que al ejecutar el programa aparezca en pantalla 2 = 2 *son 4.* ¡Cuide la correción en la escritura! El punto y coma que separa las dos expresiones PRINT obliga al ordenador a que imprima el resultado de la operación matemática justo a continuación del último caracter de la cadena anterior.

Para dejar la pantalla limpia de los caracteres impresos con anterioridad, se dispone del comando CLS.
Ambos serán estudiados posteriormente, pero hasta ese momento, usted puede **limpiar** la pantalla tecleando **CLS** y **ENTER.**

Realizar dibujos esquemáticos:

1. 10 PRINT "XXXXXX"
   20 PRINT "X X"
   30 PRINT "X X"
   40 PRINT "X X"
   50 PRINT "X X"
   60 PRINT "XXXXXX"
2. Trace las diagonales del cuadrado anterior.
3. Borre las diagonales del cuadrado anterior y uno de los lados. Transfórmelo en un rectángulo de doble base que altura.
4. Dibuje un triángulo rectángulo de catetos iguales.
5. Dibuje libremente.

Es conveniente diseñar los dibujos en un papel milimetrado antes de imprimirlos en pantalla: evitará muchos errores y pérdidas de tiempo... ¡y de paciencia!

SOLUCIONES:

2.
```
 1 REM Ficha "PRINT" Ej.: 3.2"
10 PRINT "XXXXXX"
20 PRINT "XX  X"
30 PRINT "X X  X"
40 PRINT "X  X X"
50 PRINT "X  XX"
60 PRINT "XXXXXX"
```

3.
```
 1 REM Ficha "PRINT" Ej. 3.3.
10 PRINT "XXXXXXXXXXX"
20 PRINT "X          X"
30 PRINT "X          X"
40 PRINT "X          X"
50 PRINT "X          X"
60 PRINT "XXXXXXXXXXX"
```

4.
```
  1 REM Ficha "PRINT" Ej. 3.4
 10 PRINT "X"
 20 PRINT "XX"
 30 PRINT "X X"
 40 PRINT "X  X"
 50 PRINT "X   X"
 60 PRINT "X    X"
 70 PRINT "X     X"
 80 PRINT "X      X"
 90 PRINT "X       X"
100 PRINT "XXXXXXXXX"
```

FICHAS DEL

GUIA DE PAL

IF comparación THEN sentencia

Donde el resultado de **comparación** sólo puede ser **cierto o falso.**

Obliga a pasar la secuencia de lectura a **sentencia** si el resultado de la comparación es **cierto,** en otro caso continúa en la siguiente línea de programa.

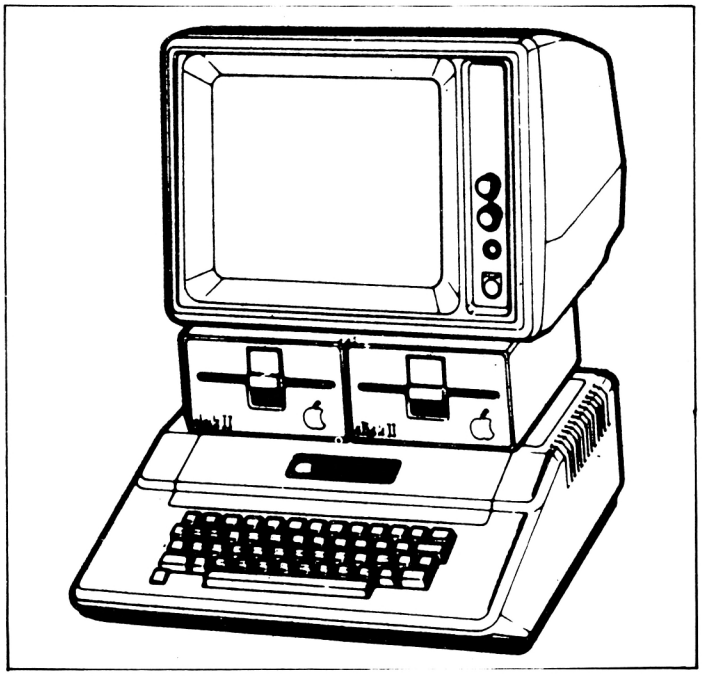
```
10 INPUT A
20 IF A = 3 THEN PRINT "TRES"
30 PRINT "NO ES TRES".
```

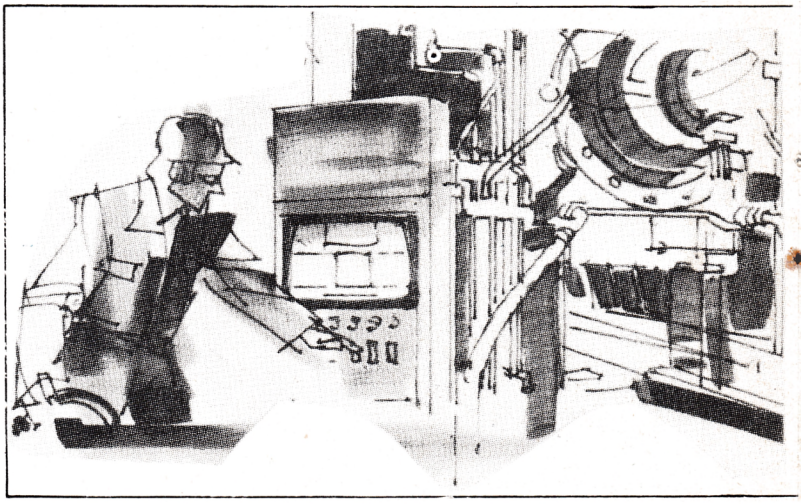
INKEY$

Capta el caracter de la tecla apretada en el momento de ser procesada esta instrucción, si no hubiera ninguna, se considera la cadena vacía.

```
10 PRINT "Apretar una tecla".
20 IF INKEY$ = " "THEN GOTO 20
50 PRINT "Tecla apretada".
```

En la línea 20 se produce un bucle sobre sí misma, ya que mientras no se apriete una tecla, el computador capta la cadena vacía, y consiguientemente, pasa la lectura a la línea 20.





INPUT "texto"; ¡Variable!

Provoca una parada en el programa para, de esta forma, dar un contenido a la variable.

Si el "texto" ha sido escrito, éste aparecerá en la pantalla durante la interrupción y hasta que el dato solicitado haya sido tecleado y se apriete la tecla ENTER.

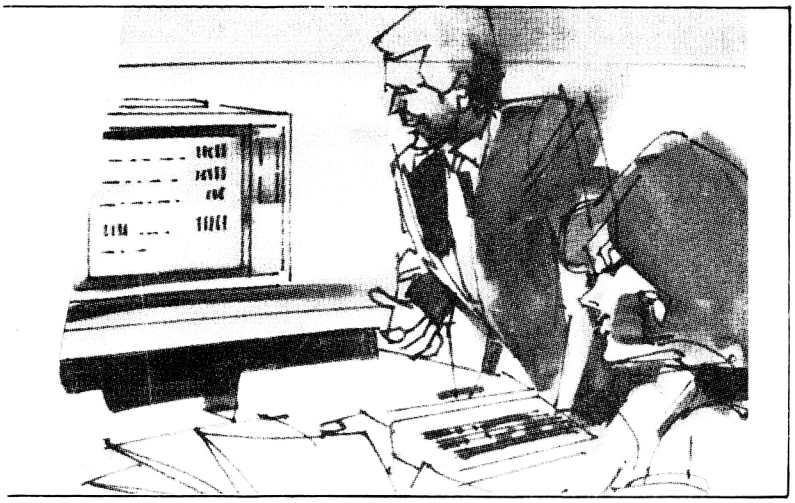
Si el "texto" no existe, sólo el **prompt** (con la expresión **prompt** se viene a indicar que el ordenador está preparado para hacer su trabajo y, para ello, hace aparecer en pantalla un símbolo, al que por extensión se llama así) típico del ordenador aparecerá en pantalla, indicando que está a la espera de información. En este caso, el dato que se introduzca será asignado a la variable.

Esta variable puede ser numérica o de caracteres y, consiguientemente, el dato introducido tiene que estar en consonancia con el tipo de variable. Es decir, si la variable se ha considerado como numérica, no se aceptará una cadena de caracteres como entrada del INPUT.

Escriba un programa que, una vez ejecutado, nos pida el número de "pasos" consumidos en una conferencia telefónica y, a continuación, el valor de cada "paso" para obtener finalmente el importe de la conferencia.

```
10 INPUT "¿Número de pasos?"; N
20 INPUT "Valor de unpaso?"; P
30 PRINT "El importe de su conferencia es";
40 PRINT N*P.
```

Con las dos primeras líneas provocaremos, al ejecutar el programa, dos interrupciones para asignar valores a las variables N y P.



INSTR (A$,B$)

Localiza y muestra la posición donde comienza la primera coincidencia de la cadena de caracteres B$ dentro de la cadena A$.

```
10 $ = "ABCD"
20 B$ = "C"
30 PRINT INSTR (A$, B$)
```

El resultado es 3.

INT ¡expresión!

Se calcula el menor de los números enteros del valor que da —o es— **expresión.**

```
10 PRINT INT 1,2
20 PRINT INT —1,2
```

LEN ¡expresión!

Calcula el número de caracteres que componen la cadena definida en **"expresión".**

PRINT LEN "PLUS".

LET ¡variable = expresión

Asigna a una variable el valor de una expresión.

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

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

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

Las variables de caracteres se representan por una sola letra seguida del símbolo $. De esta forma, el computador sabe que el contenido de la variable sería un texto.

LEFT$ ¡("expresión", expresión)!

Extrae una subcadena de **"expresión"** formada por tantos caracteres com indique la **"expresión"** numérica y contados desde la izquierda.

LIST ¡expresión1 - expresión2!

Proporciona un listado en pantalla del programa que actualmente esté en memoria, y, a partir de la línea de programa que indique **expresiones.** Si las **expresiones** no se escriben entonces se obtiene el listado completo.

LOAD ¡"expresión"!

Carga desde el disco a la memoria el programa cuyo nombre viene dado por **"expresión".**

LOCATE ¡expresión1, expresión2!

Posiciona el cursor en la columna dada por **expresión1** y fila dada por **expresión2.**

LOG ¡(expresión)!

Calcula el logaritmo natural —en base e— del valor que da —o es— **expresión.**

PRINT LN 1,1.

LOG10 ¡expresión!

Calcula el logaritmo —en base 10— del valor que da —o es— **expresión.**

PRINT LN 1,1.

MASTE

Este mes presentamos una versión sencilla del conocido juego "Master Mind". Como seguramente conocerá se trata de adivinar una clave, —en este caso numérica—, de forma que además de acertar los números que la componen, estos deben estar en la debida posición.

REGLAS DEL JUEGO

Si un determinado número existe en la clave pero no se encuentra en la posición exacta, el ordenador lo indicara con el símbolo 0 , si el número existe y además se encuentra en su sitio lo indicara con , en el otro caso posible, número no existente en la clave el ordenador no mostrará ningún signo, es decir, sólo se señalan los aciertos.

Los aciertos son totalizados en cada jugada y señalados a la derecha del número introducido, pero en ningún caso la posición de estos símbolos coincide con posiciones reales de los números, es decir indica únicamente cuántos números hay acertados y cuántos son plenos, pero no cuales lo son.

La clave está compuesta por cuatro cifras numéricas, cada una de ellas puede ser un número comprendido entre 1 y 9, estos números pueden repetirse e incluso en un caso límite (poco probable) pueden ser todos iguales.

Para la obtención de los cuatro números que componen la clave se utiliza un procedimiento aleatorio. El ordenador elabora la clave y nosotros debemos adivinarla.

COMENTARIOS

Dado que suponemos que el lector conoce las instrucciones más elementales del BASIC, como PRINT, REM, etc., únicamente comentaremos aquellas líneas o grupos de líneas más importantes.

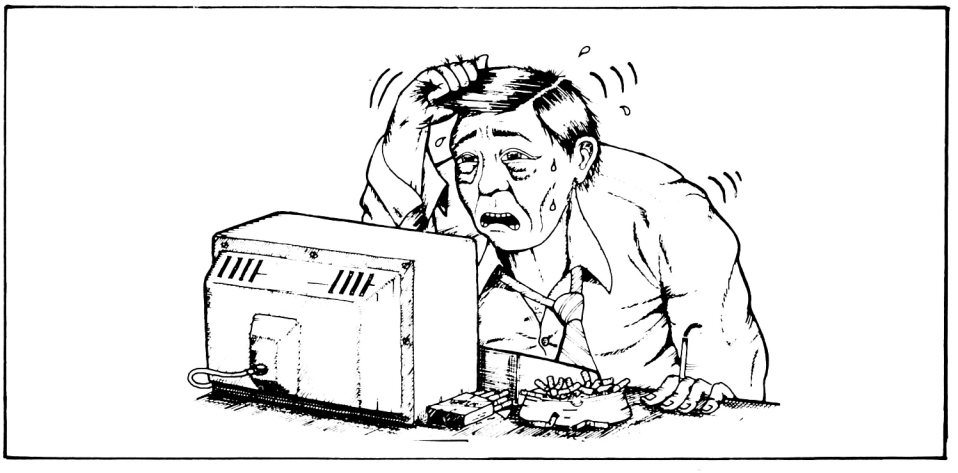
30 Establece una ventana en la parte inferior de la pantalla para la emisión de mensajes por la misma. Ocupa los 40 caracteres de la línea 25 (última de la pantalla).

60-70 CHR$ (164) Es el símbolo de acierto pleno, y CHR$ (230) el de número sólo. Ambos son caracteres standard del Amstrad, incluidos en el manual.

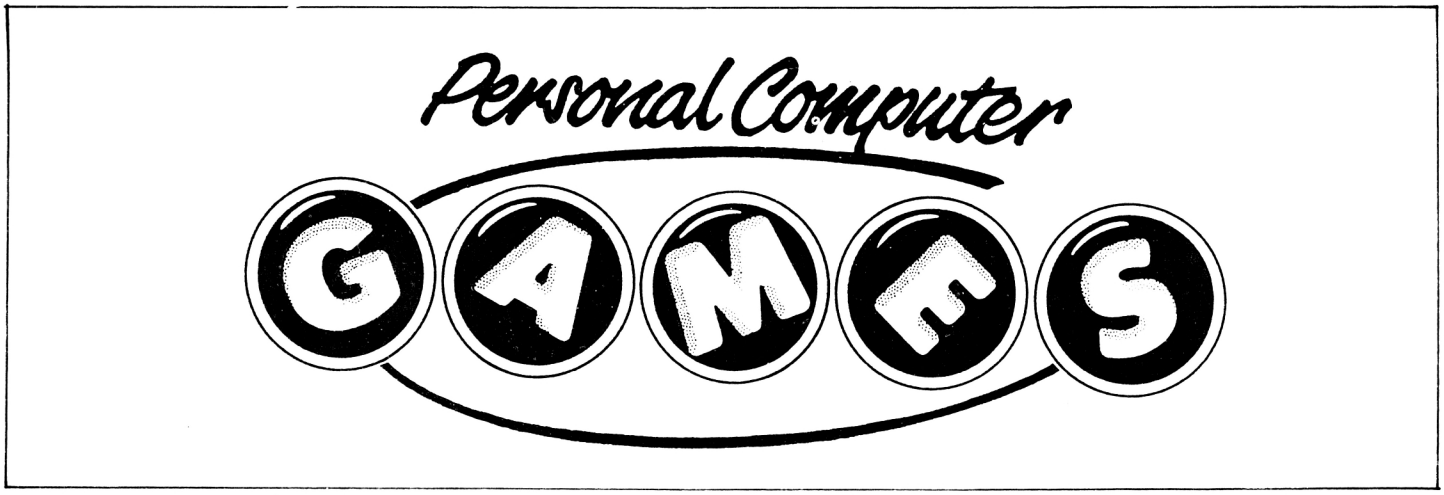
80 Mensaje en ventana, precedido por un pitido provocado por CHR$ (7). La llamada a la rutina &BB18 provoca una interrupción del programa hasta que se presione una tecla.

105 CHR$(22) Controla la impresión transparente (sobreimpresión), con CHR$(1) se activa y con CHR$(0) se anule.

120 Mejora la aleatoriedad del proceso de generación de la clave, al hacer depender ésta del la var. TIME (tiempo que lleva encendido el ordenador).



R MIND



130 Introduce un número aleatorio generado por RND, comprendido entre 1 y 9 en la matriz (ver. indexada) CLAVE, que no es previamente dimensionada, ya que sólo tiene 4 elementos (10).

170 Bucle para la entrada de los cuatro números de cada jugada.

180-190 Entrada de los números de la jugada, en forma de cadena (textos y no números propiamente), utilizando la función INKEY$ en vez de INPUT.

200-210 Si el caracter introducido es válido, se almacena en la matriz NUM, que tiene cuatro elementos como la CLAVE. La función VAL$ permite el paso de los números introducidos como textos a números propiamente dichos.

220-225 Confirmación de si los números introducidos en la jugada son los deseados (evita los errores al teclear).

255-270 Si los números no eran válidos, se hace retroceder el cursor hasta la posición inicial, para introducir nuevamente la jugada, tal como la indica el mensaje de la 270.

280 Si los números introducidos son válidos, a poner a cero unos contadores de aciertos P y Q.

300 Si un número coincide con uno de la clave y en posición, se incrementan los contadores y a la variable T$ se le agregan los símbolos correspondientes.

320 Si T$ contiene 4 caracteres se translada el control a la línea 500, a fin de comprobar si ha acertado la clave o no.

330-400 Se comprueba la existencia de números iguales a los de la clave, aunque no estén en su sitio.

420 La var. CONTADOR controla el número de jugadas, establecido en un máximo de 10. Si es menor podemos seguir jugando por la transferencia a la línea 160.

430 En caso contrario, —se acabó el tiempo—, y el ordenador muestra la clave no acertada. El CHR$ (24) invierte los colores del vídeo.

500-520 Controla la existencia o no de acierto pleno (los cuatro números y en su sitio), en caso contrario devuelve el control a la 410 para proseguir el juego si no se han agotado las jugadas disponibles.

MEJORAS

Además de las derivadas de una mejor presentación, puede incrementarse el nivel de dificultad añadiendo un dígito más, es decir que los números vayan del 0 al 9. Modifique para ello las líneas.

40 del 0 al 9
130 RND * 10
190 IF Z$ "0"

Los caracteres utilizados para mostrar los aciertos de las jugadas puede cambiarlos por otros más a su gusto, elíjalos en el manual, tome nota de su código y cambie estos valores en las líneas 60, 70, 300 y 500.

Si dispone de monitor en color puede mejorar mucho la presentación, añadiendo una buena combinación de colores.

```
10 REM JUEGO DEL MASTER MIND
20 MODE 1: PRINT TAB(15)"MASTER MIND"
30 WINDOW #1,1,40,25,25
40 PRINT "   El objeto de este juego es descubrir un codigo de cuatro cifras, de
l 1 al 9"
50 PRINT
60 PRINT CHR$(164);" = CORRECTO NUMERO Y SITIO"
70 PRINT CHR$(230);" = CORRECTO SOLO EL NUMERO"
80  PRINT CHR$(7):PRINT #1,"   PRESIONE UNA TECLA PARA CONTINUAR":CALL &BB18
90   Comienzo del juego
100 'Generar la clave (aleatoriamente)
105 PRINT CHR$(22);CHR$(0)
110 A$=""
120 RANDOMIZE TIME
130 FOR K=1 TO 4:CLAVE(K)=INT(RND*9)+1: NEXT K
140 'Entrada, de la jugada
150 CLS: PRINT CHR$(7): PRINT #1,"   INTRODUZCA LOS CUATRO NUMEROS       "
160 PRINT
170 FOR K=1 TO 4
180 Z$=INKEY$
190 IF Z$<"1" OR Z$>"9" THEN 180
200 PRINT Z$;:NUM(K)=VAL(Z$)
210 NEXT K
220  PRINT CHR$(7):PRINT#1,"Es correcta la entrada (S/N)?"
230 z$=INKEY$:IF Z$="" THEN 230
250 IF Z$="S" OR Z$="s" THEN CLS #1: GOTO 280
255 PRINT CHR$(8)+CHR$(8)+CHR$(8)+CHR$(8)
260 PRINT #0,CHR$(22)+CHR$(0);CHR$(11)+CHR$(11)+CHR$(8)+CHR$(8)+CHR$(8)+CHR$(8)
```

```
270 CLS #1:PRINT CHR$(7):PRINT #1,"    INTRODUZCA DENUEVO LOS NUMEROS     ":GOTO
 170
280 T$="":FOR K=1 TO 4:P(K)=0:Q(K)=0: NEXT K
290 FOR K=1 TO 4
300 IF NUM(K)=CLAVE(K) THEN T$=T$+CHR$(164):P(K)=1:Q(K)=1
310 NEXT K
320 IF LEN (T$)=4 THEN  PRINT CHR$(11);CHR$(22);CHR$(1);"       ";T$: GOTO 500
330 FOR I=1 TO 4
340 IF P(I)<>0 THEN 400
350 FOR J=1 TO 4
360 IF Q(J)<>0 THEN 380
370 IF T<>J AND NUM(J)=CLAVE(I) THEN T$=T$+CHR$(230):Q(J)=J:J=4
380 NEXT J
400 NEXT I
410 PRINT CHR$(11);CHR$(22);CHR$(1);"       ";T$
420 IF CONTADOR<10 THEN CONTADOR=CONTADOR+1:PRINT #1,"    INTRODUZCA LOS NUEVOS
NUMEROS     ": GOTO 160
430 PRINT #1,CHR$(24);" LO SIENTO LA CLAVE ERA ";CLAVE(1);CLAVE(2);CLAVE(3);CLAV
E(4);CHR$(24)
440 END
500 FOR K=1 TO 40:IF MID$(T$,K,1)=CHR$(164) THEN  AC=AC+1: NEXT K
510 IF AC=4 THEN  PRINT #1,CHR$(24);" ACERTO A LOS ";CONTADOR;" INTENTOS ";CHR$(
24):END
520 GOTO 410
```

Boletín de suscripción

A remitir a GTS. S.A. C/Bailén, 20. 1.º Izqda. 28005 Madrid.

Deseo suscribirme a los 11 números anuales de Amstrad Educativo por sólo 2.500 ptas. a partir del próximo número.

El importe lo haré efectivo:

- ☐ **Por giro postal n.º** .....
- ☐ **Por talón nominativo adjunto.**
- ☐ **Contra reembolso a la recepción del primer ejemplar, más gastos de envío.**

Nombre y apellidos: .....

Domicilio: .....

Ciudad: .....Teléfono .....

Fecha: .....Firma .....

LA PANTALLA DE TEXTO

Por JUAN MARTINEZ PINTOR

En el capítulo I se vieron los conceptos de palabra y lista. Se dijo entonces que una palabra es una serie de caracteres consecutivos ninguno de los cuales puede ser un separador y que una lista es una serie de objetos-logo con indicación expresa de donde comienza y donde termina siendo, a su vez, un objeto-logo con indicación expresa una palabra o una lista. Resulta pues que una lista elemental sería la formada exclusivamente por palabras. Esta lista de nivel 1 podría ser uno de los objetos (de los elementos), constituyentes de otra lista (nivel 2), en la que otros objetos podrían ser palabras o listas. Se indicó también entonces que el orden en que estén los elementos de la lista es fundamental.

EJEMPLO IV.1.

[CASA LAPIZ MESA] es una lista cuyos elementos son CASA el primero, LAPIZ el segundo y MESA el tercero. Por ello, [FOLIO [CASA LAPIZ)] es una lista cuyo primer elemento es la palabra FOLIO y cuyo segundo elemento es la lista anterior. Lógicamente la lista anterior y la lista [(FOLIO) (CASA LAPIZ MESA)] son distintas, dado que en el primer elemento de esta última es también una lista pero formada por un sólo elemento, que es una palabra. Todos los elementos de la última lista son, a su vez, listas.

Como Logo es un lenguaje extraordinario capacitado para manejar símbolos, es preciso en muchas ocasiones distinguir entre, por ejemplo, una palabra tomada como tal, y la misma palabra tomada como el objeto representado por ela. Existen órdenes que aplicadas a la palabra como tal producen un resul-

tado muy distinto del obtenido aplicándola al objeto representado. Tendremos ocasión de ver muchos ejemplos de ello.

En consecuencia existe un modo para indicar a Dr. Logo que una palabra ha de ser tomada ella misma como sujeto de la orden. Este modo consiste sencillamente en anteponer el carácter **comillas** (") a la palabra. Esto mismo puede pensarse teniendo en cuenta el concepto de operación estudiado en el punto II.1: Cuando una palabra va precedida por el caracter ", se ejecuta una operación sobre ella cuyo resultado es precisamente el objeto-logo que consiste en esa palabra precisamente. Del mismo modo podemos pensar en los caracteres de delimitación de listas ([y]).

EJEMPLO IV.2

Por ello si se escribe directamente en el ordenador, por ejemplo, **fd 50** la orden será ejecutada de inmediato porque se toma la palabra fd como lo que representa (un primitivo); pero si se escribe **[fd 50]** entonces se escribe esta lista en pantalla sin que ninguna acción se lleve a cabo (en según que versiones de Logo podría ni tan siquiera escribirse en pantalla la lista). Parecido resultado se obtiene escribiendo **"fd 50"** porque fd es tomado ahora como la palabra y no como el primitivo que representa.

De todo lo anterior podemos sacar la conclusión de que Logo dispone de muchos instrumentos para manejar texto considerado como tal. En efecto, ello es así y poco a poco iremos teniendo ocasión de comprobarlo.

IV. 2. IMPRESION DE TEXTOS EN PANTALLA

En el punto III.3 se ha visto que la pantalla puede estar en uno de los estados siguientes: sólo texto, sólo gráficos o una mezcla de texto en todos los modos posibles de la pantalla incluso entre los gráficos. Existen instrucciones que permiten escribir textos gráficos). Vamos a ver a continuación las órdenes que permiten escribir en las pantallas de texto y mixta.

La instrucción

pr

requiere un objeto-logo como dato sobre el que actuar y, cuando se ejecuta, el objeto es escrito en pantalla.

EJEMPLO IV.3.

Si se ejecuta la orden

pr "Hola

en pantalla se escribe la palabra "Hola". Del mismo modo la instrucción,

pr [Hola buenos días]

escribe todos los elementos de la lista anterior en la pantalla. Obsérvese que el resultado de ejecutar esta orden no incluye la escritura de los corchetes exteriores de la lista. Por el contrario, la instrucción,

pr [Hola buenos días [tenga usted]]

no escribe los corchetes exteriores de la lista principal, pero si los de la lista interior; el resultado sería:

Hola buenos días [tenga usted].

En realidad el primitivo pr admite como dato una lista. De este modo la lista más exterior no ha de delimitarse entre corchetes, y el resultado será que se escribirán en pantalla todos los objetos de la lista pero después de cada uno de los cuales el cursor pasará a la línea inmediatamente inferior (se produce un retorno de carro).

EJEMPLO IV.4.

Si ejecutamos,

pr "Hola "buenos "días

el resultado será

Hola
buenos
días

pero si ejecutamos

pr "Hola "buenos "días [tenga usted]

el resultado será

Hola
buenos
días
[tenga usted]

Observemos en el ejemplo anterior que cada palabra ha se hacerse preceder del caracter "para que no se interprete, cuando no se incluyen en una lista delimitada con corchetes. Para que no se ejecute un retorno de carro tras la impresión de cada uno de los elemento, basta encerrar toda la orden entre paréntesis (incluso la palabra pr). De este modo, sólo después de escrita toda la lista que se pase a pr como dato, se ejecutará el retorno de carro.

EJEMPLO IV.5

(pr "Hola "buenos "días [tenga usted])

escribe en pantalla

Hola buenos días tenga usted

y el cursor pasa entonces a la línea inferior.

Todo lo dicho para la orden pr es válido para el primitivo

show

con la única diferencia de que este último muestra en pantalla los paréntesis exteriores de la lista. Ya se verá que ésto puede ser de gran utilidad para el estudio de procedimientos, cuando se tratan como listas.

Por lo que a presentación de texto respecta, en la pantalla de texto, sólo resta la instrucción.

type

cuya única diferencia con pr es que type no ejecuta un retorno de carro después de ejecutada. Ello significa que dos órdenes type consecutivas presentarían sus resultados en la misma línea (si ello es posible), uno a continuación del otro. Hemos, no obstante, de tomar nota de que el comportamiento anteriormente descrito de type se refiere a su ejecución en un procedimiento. En nivel superior el comportamiento es exactamente igual que el de pr.

Aunque el desarrollo completo de las operaciones numéricas será desarrollado por completo en el capítulo XIV, es ésta una buena ocasión para presentarlas y así poder realizar cálculos con Dr. Logo. En efecto, se dispone de las operaciones aritméticas elementales, designadas con los caracteres:

+ para la suma,
— para la resta,
* para el producto y
/ para la división.

EJEMPLO IV.6.

```
pr 2 + 5
7
pr 10 — 4
6
pr 4 * 3
12
pr 120 / 60
2
```

El orden de prioridad de las operaciones así como el uso de paréntesis puede hacerse en Dr. Logo del mismo modo que se hace en otros lenguajes. De momento no insistimos más en ello.

IV.3. MODOS DE LA PANTALLA MIXTA

Como se vio en el punto III.3, y se ha recordado más arriba, existen tres posibles estados de la pantalla. La pantalla mixta es capaz de albergar gráficos y reserva las líneas más bajas de la pantalla para texto. A la pantalla mixta se accede con la orden ss como ya sabemos e inicialmente el número de líneas que se reservan para texto es 5. Lógicamente, si la tortuga entra en la zona de texto desaparece de la vista, pero continúa obedeciendo las instrucciones que reciba.

El número de líneas de texto que deben reservarse en la pantalla mixta puede modificarse con el primitivo.

setsplit N

donde N representa el número de líneas que se reservarán; debe estar comprendido entre 1 y 25. Si se escribe un número fuera del rango anterior se recibe el mensaje de error:

setsplit doesn't like 0 as input

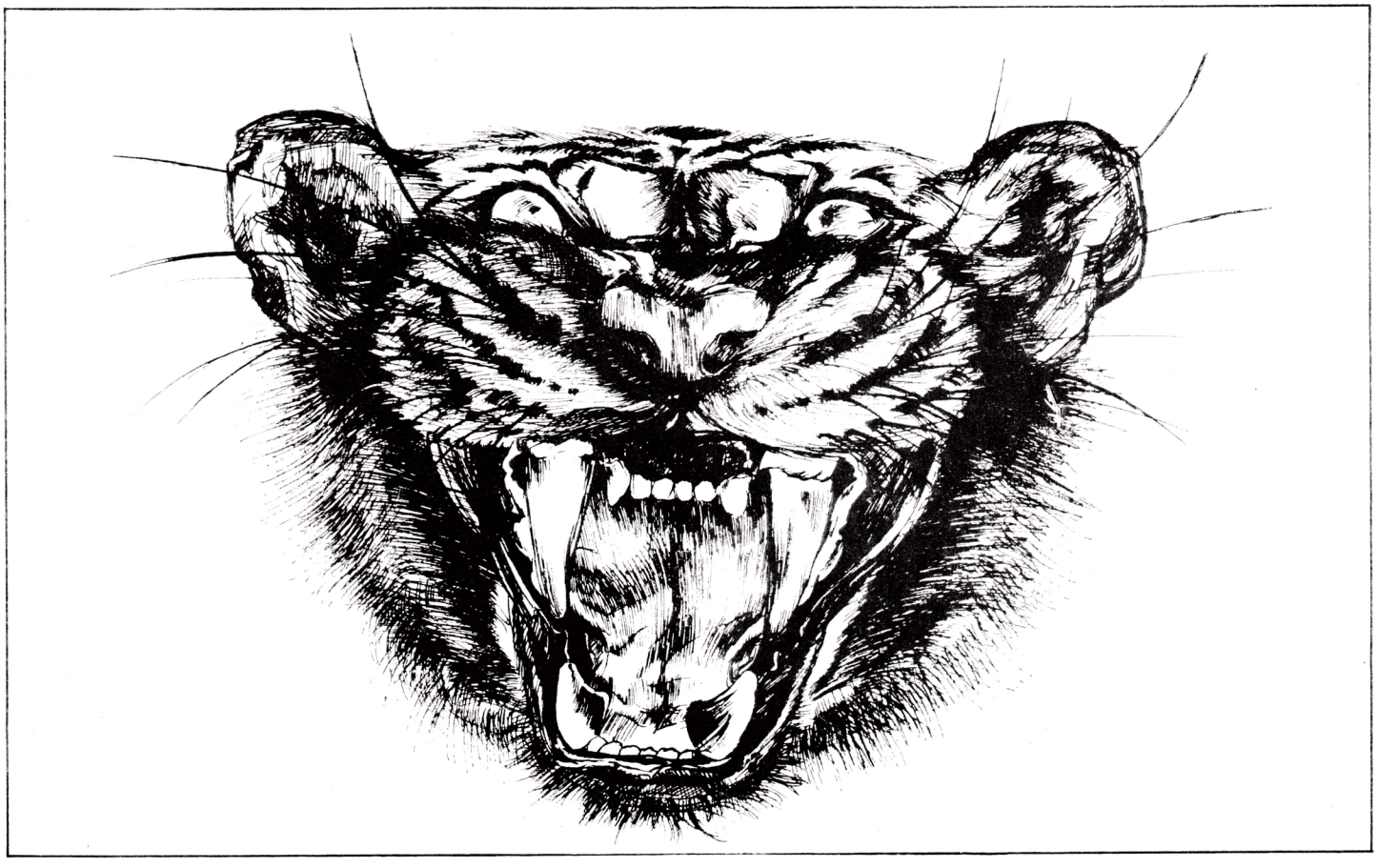
y el número manejado con esta instrucción no será alterado.

Observemos la gran utilidad que puede representar esta orden para escribir programas en los que el número de líneas de texto sea importante. Podemos citar como ejemplo de ello el programa DISPAROS correspondiente al número 2 de la revista. Estudiando su ejecución, y su listado, vemos que se ha usado una sola línea final para pedir los datos del disparo, reservando el resto de la pantalla para el movimiento de la tortuga.

Por otra parte, resulta que aunque existe una orden expresa para borrar los gráficos (cs y clean como se vio en el punto III.7), si con la orden setplit se invaden parte o todos los gráficos, esta parte será borrada; es decir, que no sólo desaparecerán de la vista al ser ocultada por las líneas de texto; si posteriormente se regresa al número de líneas de texto previas a la ejecución de la orden setsplit, los gráficos correspondientes a la zona invadida ya no estarán allí.

Para finalizar el capítulo indicamos que la instrucción siguiente borra los textos de la página de texto o pantalla mixta situando el cursor en la esquina superior izquierda de la zona correspondiente. De modo que en pantalla mixta, el cursor será situado en la primera de sus líneas. La orden a que nos referimos es:

ct



PROGRAMA COMENTADO EN LOGO

MASTER MIND

Por JUAN MARTINEZ PINTOR

El siguiente programa desarrolla una parte del juego: el Dr. Logo "piensa" un número que el jugador debe averiguar. El juego se completará en otro número desarrollando la parte en que es el jugador quien piensa el número y Dr. Logo quien intenta averiguarlo.

No se han utilizado gráficos tortuga para representar ninguna situación, pero se sugiere al lector que piense algún tipo de gráfico tortuga que vaya progresando a la vez que se averiguan números.

Se observará al jugar que si el número buscado contiene un determinado dígito varias veces (como el 8 en el número 88238) entonces será considerado herido (o muerto) tantas veces como aparece. Así si se escribe el número 678000 se obtendrá un resultado de 3 heridos.

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

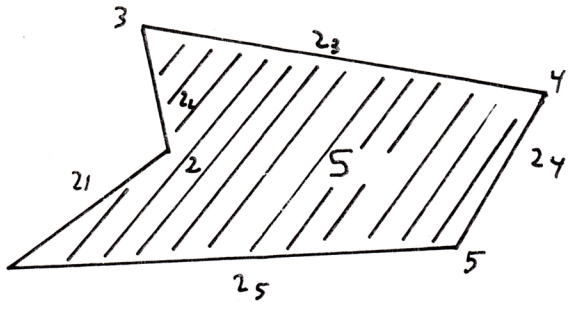
```
to master.mind    (* Definicion del procedimiento principal *)
presentacion      (* Procedimiento invocado *)
ct                (* Primitivo que borra los textos *)
jugar             (* Llamada a procedimiento *)
end               (* Concluye el modo definicion *)
------------
to  presentacion      (* Esta instruccion situa a Dr.  Logo en  el
                         modo  definicion de procedimientos.  Va a
                         definirse el procedimiento llamado
                         "presentacion. *)
ts ct                  (* ts selecciona la pantalla de texto y ct borra
                          los textos *)
pr [MASTER MIND]       (* Escribe en pantalla *)
repeat  4[pr []]       (* Repite 4 veces la escritura de una linea  en
                          blanco *)
pr [En cada jugada hay que escribir]
pr [un numero de 5 cifras, todas las]
pr [cuales pueden ser 0]
pr []
pr [En cada jugada se informa del]
pr [numero de muertos y heridos]
repeat 5 [pr[]]
type  [tecla  para seguir]        (* Escribe sin seguir de  retorno  de
                                     carro *)
make  "seguir  rc                 (* Espera hasta que  se  pulse  un
                                     caracter en el teclado y despues
                                     lo asigna a la variable "seguir *)
end
------------
to jugar
coge.num          (* Llama al procedimiento "coge.num *)
label "a          (* Define este punto como la etiqueta "a, para poder
                     enviar aqui el control *)
pedir.num         (* Llamada a procedimiento *)
comprobar (list :x1 :x2 :x3 :x4 :x5) (list :y1 :y2 :y3 :y4 :y5) 1 1
(*  Llamada al procedimiento llamado  comprobar. Este  procedimiento,
como puede verse mas abajo,  tiene 4 inputs: dos listas y dos numeros,
por ello su llamada indica las listas y los numeros *)
```

CALCULO DE AREAS Y PERIMETROS POR COORDENADAS

Víctor J. Campo López

Bajo este título presentamos este mes un programa que permite calcular el área y perímetro de una figura plana irregular, así como la longitud de cada uno de sus lados, a partir de las coordenadas cartesianas de sus vértices.

En una poligonal como se indica en la figura, cada uno de los vértices viene definido por una pareja de coordenadas, referidas a unos ejes cartesianos X e Y. Si numeramos estos vértices comenzando por el más próximo al origen y en sentido de las agujas del reloj, obtendremos todos los datos que son necesarios para el programa, a partir de ellos, se obtendrá como se ha indicado el área encenada por la poligonal, el perímetro de la misma y la longitud de cada uno de sus lados.



nº puntos = 5 + 1 = 6

OBSERVACIONES:

1) Para el programa el número de vértices es el total numerados más uno, ya que el primero se cuenta como origen y como fin, es decir dos veces.

2) Aunque se ha indicado que la numeración debe hacerse en sentido de las agujas del reloj, puede realizarse en sentido contrario, siempre que los puntos sean correlativos, la única diferencia reside en que el área vendrá expresada con un número positivo o negativo, pero idéntico si se toma el valor absoluto, si se desea puede modificarse la línea 470 añadiendo ABS (área/2).

3) A partir de la línea 500 se realiza un dibujo de la poligonal con indicación de sus vértices, a fin de visualizar la forma gráfica de dicha poligonal. Si no se desea utilizar esta parte pueden suprimirse las líneas 480 a 640, final del programa.

FORMULACION EMPLEADA:

La superficie viene dada por:

$$\$ = \sum_{i=2}^{i=n+1} (Xi\text{-}1 - Xi) * (Yi-1 + Yi)) / 2$$

La longitud de los lados es:

$$Li = \sqrt{(Xi\text{-}1 - Xi)^2 + (Yi\text{-}1 + Yi)^2}$$

Y el perímetro será: $P = \sum Li$

```
if  :muerto<5 [(pr "muertos :muerto "heridos :herido) pr [] go  "a][pr
"correcto]]
(*  Condicional.  Si la variable "muerto" tiene un valor menor que  5,
entonces  se escribe el numero de muertos y heridos,  se  escribe  una
linea  en  blanco y se transfiere el control al punto marcado  por  la
etiqueta "a (label "a).  En otro caso se escribe "correcto" y el juego
termina *)
end
------------
to coge.num
make "x1 random 10 make "x2 random 10 make "x3 random 10
(*  Asignaciones  de valores a las  variables  "x1..."x5.  Se  escogen
numeros al azar entre 0 y 9 *)
make "x4 random 10 make x5 random 10
end
------------
to pedir.num
pr []
(pr "escribir "el "numero)
make "y1 rc type :y1
(*  Se  pide al teclado el contenido de la variable  "y1.  Despues  se
imprime en pantalla sin que se siga con retorno de carro *)
make "y2 rc type :y2
make "y3 rc type :y3
make "y4 rc type :y4
make "y5 rc type :y5
pr []
make  "muerto  0
(*  Antes de comenzar las comprobaciones,  se inicializa el numero  de
muertos y heridos a 0 *)
make "herido 0
end
------------
to comprobar :lista1 :lista2 :n :m
(* El procedimiento "comprobar lleva como datos de entrada las  listas
que  ha  de  comprobar  y los numeros de los elementos  de  ellas  que
comprueba si son iguales *)
if item :n :lista1 = item :m :lista2 [make "test "i][make "test "d]
(*  Si el elemento numero :n de la lista llamada "lista1 es  igual  al
elemento  numero :m de la lista "lista2 entonces se asigna el valor "i
(de iguales) a la variable "test; de otro modo se asigna "d *)
if   :test="i   [if  :n=:m  [make  "muerto   :muerto+1] [make  "herido
:herido+1]]
(*  Si  la  variable "test contiene al valor "i entonces  si  en  esta
activacion  de "comprobar  es :n igual a :m se aumenta  el  numero  de
muertos de la jugada; si :n y :m son distintos se aumenta el numero de
heridos *)
if :m<5 [make "m :m+1][make "m 1 make "n :n+1]
(* Si el valor de la variable "m es menor que 5 en esta activacion del
procedimiento "comprobar, entonces se aumenta "m en 1; de otro modo se
asigna a "m el valor 1 y se aumenta "n en 1 *)
if or :n<5 :n=5 [comprobar :lista1 :lista2 :n :m]
(* Siempre que :n sea menor o igual que 5, el procedimiento "comprobar
se  llama a si mismo pero con los nuevos valores de "n y  "m.  Resulta
pues que este procedimiento es recursivo. *)
end
------------
```

```
10 REM CALCULO DE AREAS Y PERIMETROS
          POR COORDENADAS
20 ' Victor J. Campo Lopez  May-86
30 MODE 1
40 LOCATE 7,5: PRINT CHR$(24);"CALCULO D
E AREAS Y PERIMETROS"
50 LOCATE 12,7: PRINT "POR COORDENADAS";
CHR$(24)
60 LOCATE 1,12
70 PRINT "PARA INTRODUCIR EL NUMERO DE P
UNTOS DEBE  CONSIDERARSE EL ORIGEN DOS V
ECES, COMO  PUNTO DE PARTIDA Y COMO LLEG
ADA."
80 PRINT : PRINT
90 DEF FN A(P)=(X(P-1)-X(P))*(Y(P-1)+Y(P
))
100 DEF FN D(P)=SQR((X(P-1)-X(P))^2+(Y(P
-1)-Y(P))^2)
110 INPUT "NUMERO DE PUNTOS";N
120 DIM X(N),Y(N),DIS(N)
130 CLS: PRINT "INTRODUCIR COORDENADAS D
E LOS PUNTOS"
140 PRINT "-----------------------------
----------"
150  FOR P=1 TO N-1
160 PRINT "PUNTO";P;
170 INPUT "X,Y";X(P),Y(P)
180 NEXT P
190 PRINT "PUNTO";N;"cierre";x(1);y(1)
200 x(n)=x(1):y(n)=y(1)
205 PRINT : PRINT "PRESIONE UNA TECLA PA
RA VERIFICAR DATOS"
206 CALL &BB18
210 MODE 2: PRINT TAB(18)"RELACION DE CO
ORDENADAS DE LOS PUNTOS"
220 WINDOW #1,1,40,24,25
230 PRINT "  PUNTO"TAB(15)"X";TAB(28)"Y"
;TAB(42)"PUNTO";TAB(54)"X";TAB(67)"Y"
240 PRINT "=============================
========================================
=========="
250 FOR P=1 TO N:PRINT "   ";P,X(P),Y(P)
,: NEXT P
260 PRINT #0
270 PRINT #1,"D A T O S      C O R R E C
 T O S   (S/N)"
```

```
280 Z$=INKEY$:IF Z$="" THEN 280
290 Z$=UPPER$(Z$):IF Z$="S" THEN 360
300 CLS #1: INPUT #1,"CUANTOS PUNTOS ERR
ONEOS ";D
310 FOR P=1 TO D
320 CLS #0
330 INPUT "INTRODUCIR: PUNTO, X, Y ";PE,
X(PE),Y(PE)
340 NEXT P
350 GOTO 210
360 FOR P=2 TO N
370 AREA=AREA+FN A(P)
380 DIS (P)=FN D(P)
390 DISTANCIA=DISTANCIA+ FN D(P)
400 NEXT P
410 MODE 1
420 PRINT "PUNTO  DISTANCIA"
430 PRINT "================"
440 PRINT 1;TAB(9)" ----"
450 FOR P=2 TO N: PRINT P;:PRINT TAB(6)U
SING "####.###";DIS(P): NEXT P
460 PRINT
470 PRINT "PERIMETRO=";DISTANCIA,"AREA="
;AREA/2
480 LOCATE 20,2: PRINT "PRESIONE TECLA"
490 IF INKEY$="" THEN 490
500 REM    D I B U J O
510 PRINT CHR$(22);CHR$(1)
520 MODE 2
530 XMAX=X(1):YMAX=Y(1)
540 TAG
550 FOR P=2 TO N
560 IF X(P)>XMAX THEN XMAX=X(P)
570 IF Y(P)>YMAX THEN YMAX=Y(P)
580 NEXT P
590 KX=620/XMAX:KY=380/YMAX
600 FOR P=2 TO N:PLOT X(P-1)*KX,Y(P-1)*K
Y:DRAW X(P)*KX,Y(P)*KY
610 MOVER 5,12:IF P=N THEN  PRINT "1"; E
LSE PRINT P;
615 NEXT
620 TAGOFF
630 PRINT CHR$(22);CHR$(0)
640 IF INKEY$="" THEN 640
```

NUESTRO PRÓXIMO NÚMERO

AMSTRAD EDUCATIVO

N.º 5

EL AMSTRAD Y EL CMP

Ficheros en el AMSTRAD

Basic del AMSTRAD

Fichas del AMSTRAD

Programando en BASIC

Logo del AMSTRAD

Programa en LOGO

El Programa técnico del mes

LA "BIBLIA" DEL

¡ya está a la venta!

AMSTRAD

FASCICULO 5

295 PTAS.

DOMINARLO ES UN JUEGO

Sound-on-Sound

SUS MEJORES RECUERDOS