AMSTRAD EDUCATIVO

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

295 Ptas.

Nº 3

TODO SOBRE EL

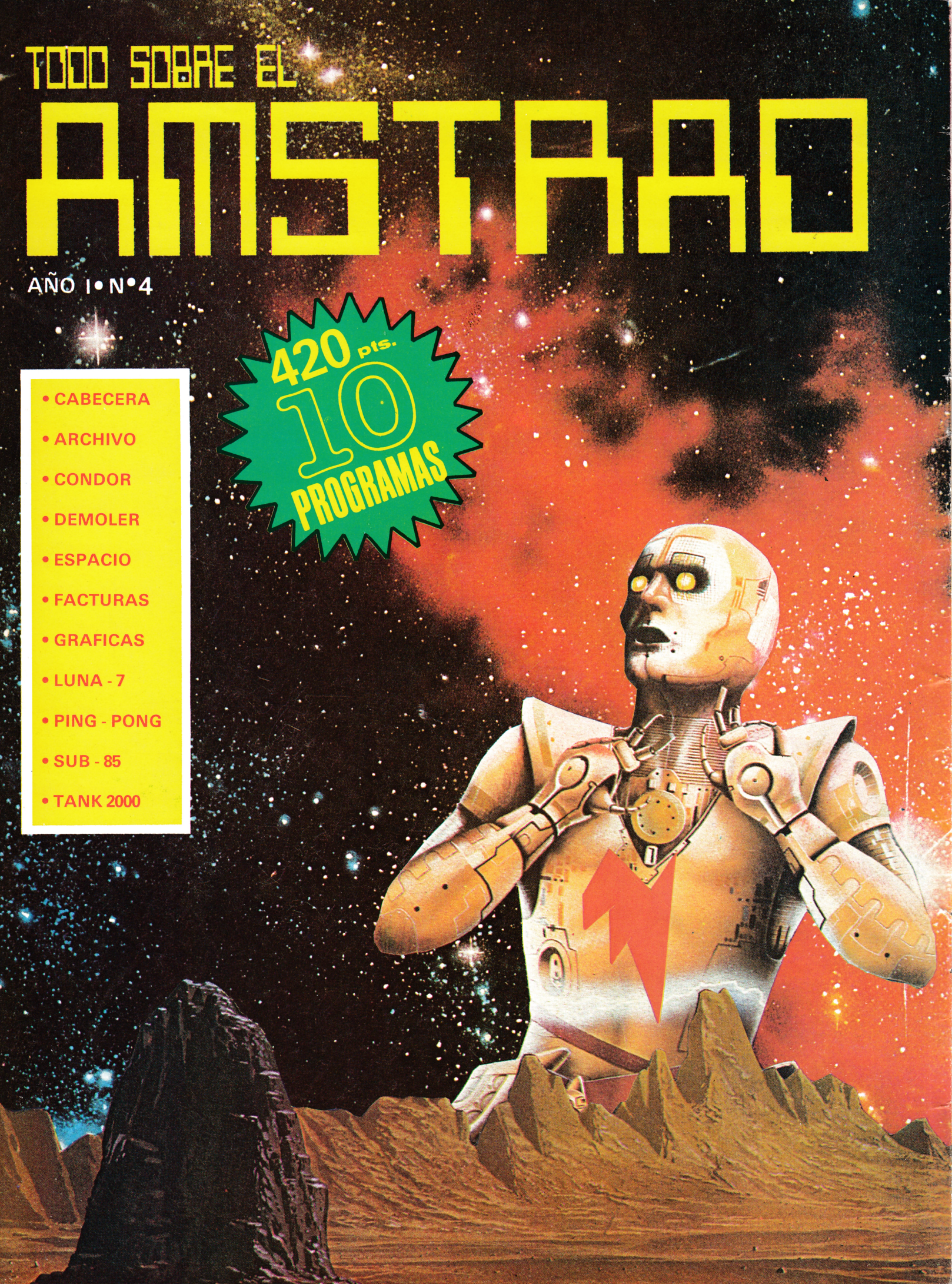
AMSTRAD

AÑO I • N°4

420 pts.

10 PROGRAMAS

- CABECERA
- ARCHIVO
- CONDOR
- DEMOLER
- ESPACIO
- FACTURAS
- GRAFICAS
- LUNA - 7
- PING - PONG
- SUB - 85
- TANK 2000



EDITORIAL

Es fácil escribir cuando uno se dirige a unos amigos. Este es un momento feliz para toda la redacción de AMSTRAD EDUCATIVO. La gran avalancha de cartas recibidas nos ha demostrado el gran interés por mejoraros en el manejo de vuestro ordenador, lo cual nos ha servido de estímulo para tratar de superarnos día tras día.

En este número hemos incluido en el apartado técnico, el programa "Leyes de momentos de una viga", que basado inicialmente en el método de Cross nos calcula el momento resultante de la curva obtenida.

El juego que os ofrecemos listado este mes dentro de la sección PROGRAMANDO EN BASIC es el conocido "Juego de los Barcos". En él podéis introducir una serie de mejoras que esperamos os gusten.

Que os divirtáis y hasta el próximo número.

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: Susana M. Villalba.
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
Polig. 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 CM/P

Referente a la denominación de ficheros en CP/M

Todo manual de un ordenador que trabaje, o pueda trabajar, bajo el control del CP/M, tendrá un apartado dedicado a la forma de aludir, o referirse, a los ficheros (**file references**) contenidos en disco.

Por diseño, CP/M permite referirse a un sólo fichero de entre los almacenados en el disco insertado en una unidad de disco predeterminada, o a un grupo de ficheros de ese disco.

Dicho esto, recordemos que a todo fichero se le debe asignar a un nombre.

CP/M tiene establecidas unas reglas generales para dar nombre a un fichero, las cuales han de ser tenidas en cuenta en el momento de la denominación, y que son las que se exponen a continuación.

En primer lugar, el nombre debe tener al menos un caracter y, como máximo, ocho.

Por otra parte, este nombre puede tener un apéndice a su derecha compuesto por un punto y haste tres caracteres.

Este apéndice, dependiendo del manual o publicación que hable del tema, recibe diferentes apelaciones, como son "extensión de fichero", "nombre secundario" o "tipo de fichero"

Nosotros aquí, lo mencionaremos por **tipo de fichero** , ya que está más acorde con la función que cumple.

En cuanto a los caracteres de uso prohibido al dar el nombre a un fichero, y a su tipo, depende de lo que cada manual diga al respecto.

No obstante, no se deben usar en general los siguientes:

< > . , ; : = ? [ ] $

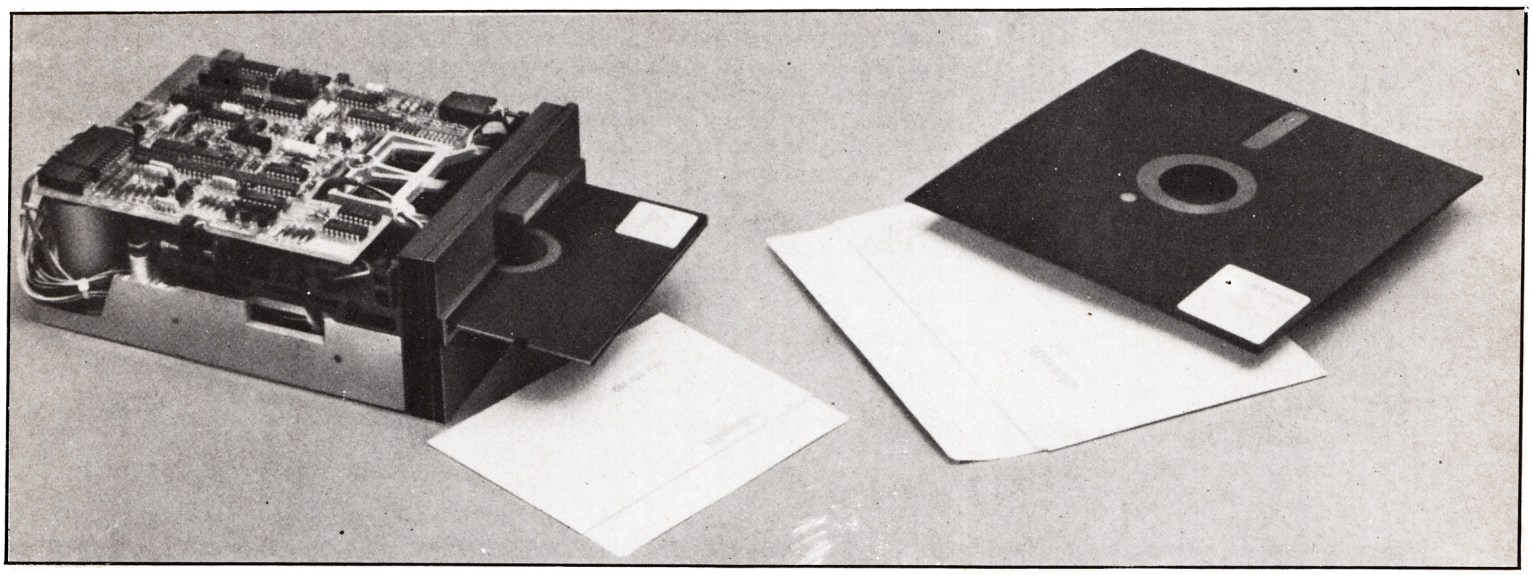
Como por ejemplo para aplicar lo dicho, supongamos que se ha de poner nombre a un fichero dedicado a controlar una biblioteca, con libros dedicados a cualquier materia. En estas condiciones, podrían valer tanto LIBROS como BIBLIOTE.

Más, si el caso fuera reducido a volúmenes conteniendo exclusivamente temas geográficos, quizá fuera conveniente usar un nombre de fichero seguido de un tipo de fichero, de este modo:

LIGROS. GEO o BIBLIOTE. GEO

En este sentido, es conveniente añadir que, si bien el operador puede asignar a su juicio, el tipo de fichero, algunos lenguajes y aplicaciones desarrolladas baja CP/M asignan el tipo de fichero automáticamente, y suelen responder a alguna de las denominaciones dadas en la siguiente lista:

TIPO de fichero	Contenido
* ASC	Caracteres ASCII.
* ASM	programa en lenguaje de asamblea del tipo Z-80.
* A86	programa en lenguaje de asamblea del tipo 8086.
* BAK	copia de un fichero existente con el mismo nombre.
* BAS	programa en BASIC.
* CAL	datos de Supercalc.
* CMD	programa transitorio ejecutable, en CP/M-86.
* COB	programa en COBOL.
* COM	programa transitorio ejecutable, en CP/M-80.
* DAT	fichero de datos.
* FOR	programa en FORTRAN.
* HEX	fichero hexadecimal producido por ensamblador (CP/M-86).
* H86	fichero hexadecimal producido por ensamblador (CP/M-80).
* ITN	programa compilado en c BASIC.
* LST	listado de compilación o asamblea.
* OBJ	programa objeto.
* PAS	programa en PASCAL.
* PRN	listado de compilación o asamblea.
* REL	programa objeto reubicable.
* SUB	fichero de órdenes a ejecutar por SUBMIT.
* TXT	textos.
* $$$	temporal.



Hasta el momento, se está aludiendo a un sólo fichero por su nombre completo, sin ninguna clase de ambigüedad.

Este tipo de referencia a ficheros, se conoce en inglés como **unambiguous file name,** y de aquí el acrónimo **ufn** que será utilizado como sinónimo de "nombre fichero sin ambigüedad".

COMODINES (WILD-CARDS)

Veamos, a continuación, como seleccionar un ficheo o un grupo de ficheros de forma ambigüa: **ambiguorus file nafe (afn).**

Para cubrir este objetivo, CP/M pone a nuestra disposición ciertos caracteres, llamados "comodines" **(wild-cards),** de los que se da detalle a continuación:

?

Función asignada: Este comodín vale por cualquier caracter de los admitidos para conformar el nombre y tipo de un fichero.

EJEMPLOS:

1.°— Supongamos que se ha creado una serie de ficheros con el mismo nombre pero los tipos varían de esta forma:
BIBLIOTE. G11
BIBLIOTE. G12
BIBLIOTE. H11
BIBLIOTE. H12
BIBLIOTE. L11
BIBLIOTE. L12

En esta situación, si interesara referirse a todos los ficheros cuyo nombre fuera BIBLIOTE y su tipo comenzara por cualquier caracter y los dos últimos fueran 11, definiríamos este grupo mediante el siguiente **afn:**
BIBLIOTE. ?11
Con lo cual estaríamos aludiendo a:
BIBLIOTE. G11
BIBLIOTE. H11
BIBLIOTE. L11
.....
.....

2.°— Si ahora, con el fin de fijar la idea, admitimos, además, este otro conjunto de ficheros:

BIBCAT. G11
BIBCAT. G12
BIBCAT. H11
BIBCAT. H12
BIBCAT. L11
BIBCAT. L12

En esta ocasión, nuestro deseo es localizar todos los ficheros cuyos nombres comiencen por BIB y su tipos comiencen por H y acaben con cualquier caracter. Para ello tendríamos que dar la siguiente referencia:
BIB?????. H1?
El grupo seleccionado así es:

BIBLIOTE. H11
BIBLIOTE. H12
BIBCAT. H11
BIBCAT. H12

3.°.—Observar que no es la misma la referencia.
BIBCAT. ?1 que BIBCAT. ??1
El primero simplemente no existe. El segundo está haciendo referencia a todos los ficheros BIBCAT que acaben en 1.
Utilizando este comodín podemos aludir a todos los ficheos de un disco, mediante la referencia
????????. ???
Lo cual dará el conjunto de los ficheros contenidos en el disco.
Pero, para conseguir esto, es más cómo recurrir a otro comodín.

*

Función asignada: Este comodín vale por el nombre y/o el tipo de un fichero.

Ejemplos:

1.°— Con todos los ficheros citados anteriormente colocados en un disco, la referencia:
* . L12
nos seleccionaría:
BIBLIOTE. L12
BIBCAT. L12

2.°— En las mismas circunstancias, la referencia:
BIBCAT . *
daría
BIBCAT... G11
BIBCAT... G12
BIBCAT... H11
BIBCAT... H12
BIBCAT... L11
BIBCAT... L12

EJERCICIOS CON NOMBRES DE FICHEROS

1.º— ¿Es válido este nombre de fichero?: 1412DC.PRU ...
Respuesta.—No. Porque comienza con un caracter distinto a una letra.

2.º— ¿Es válido este nombre de fichero?: DONQUIJOTE ..
Respuesta.—Sí, pero sólo se aceptan los ocho primeros: DONQUIJO.

3.º— ¿Es válido este nombre de fichero?: QUIJOTE. CER .
Respuesta.—Sí.

4.º— ¿Es válido este nombre de fichero?: SEAT.127 .....
Respuesta.—Sí.

5.º— Cuál es el tipo de fichero MOVCPM.COM? .....
Respuesta.—COM.

6.º— ¿Qué ficheros quedan referenciados por *.*? .....
Respuesta.—Todos los del disco.

7.º— ¿Qué ficheros quedan aludidos mediante *.COM? .
Respuesta.—Todos los de tipo .COM contenidos en el disco.

8.º— ¿A qué ficheros aludimos con PRUEBA.12? .....
Respuesta.—A todos cuyo nombre y dos primeros caracteres coincidan y estén contenidos en el disco.

9.º— ¿Es la misma referencia * PRUEBA.*23 que *.*? ...
Respuesta.—Sí. Los caracteres a la derecha de un asterisco no son considerados.

10.º— ¿Es la misma referencia PRU*.B* que PRU?????.B?? ? .....
Respuesta.—Sí. Los caracteres a la izquierda de un asterisco si son considerados.

EJERCICIOS CON UNIDADES DE DISCO

1.º— Cuál es la unidad de disco por defecto si la impronta en pantalla es "A>" .....
Respuesta.—A.

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

3.º ¿Cuál será la impronta tras la orden anterior? .....
Respuesta.—B>.

4.º ¿Es correcta esta orden? Ñ:
Respuesta.—No, porque la Ñ no es caracter estandar ASCII y, además, y en todo caso, la versión 2.0 sólo permite en línea 16 unidades de disco.

REFERENTE A LAS UNIDADES DE DISCO

Páginas atrás comentábamos que las unidades de disco conectadas a un computador son reconocidos por el sistema mediante letras.

Estas letras, suelen ser A, B, C y D, aunque cada fabricante impone sus criterios. Por esta razón, se deberá confirmar este punto en el manual de cada ordenador. Aquí se seguirá el criterio anterior.

También se dijo, que inmediatamente después de la conexión, la unidad de disco elegida por CP/M es la A. Esta situación queda reflejada en la pantalla al mostrar

A>

Como sabemos, esta es la impronta indicativa de que CP/M está a nivel operador o, dicho de otro modo, el CCP está a la espera de órdenes.

Pero, en estas circunstancias, todas las órdenes CP/M relativas a ficheros serán sobreentendidas y dirigidas a la unidad de disco A, y al disco instalado en ella.

Esta es la "unidad por defecto" o "unidad implícita".

El operador, cuando la CP/M está a su nivel, puede dar la orden oportuna para cambiar a otra unidad de disco, y convertirla, así, en la unidad de disco por defecto actual.

Esta orden consiste en escribir la letra correspondiente a la unidad de disco seleccionada seguida de dos puntos (:). Ejemplo:

A>B:

Al pulsar <c r >, la pantalla indicará el cambio de esta forma

A>B:
B>

A partir de este momento, todas las referencias a ficheros serán dirigidas al disco colocado en la unidad B.

Cerraremos este epígrafe diciendo que todas las minísculas introducidas al nombrar un fichero o seleccionar una unidad de disco son transformadas a mayúsculas, al ser procesadas por CCP.

COMO CREAR FICHEROS EN DISCO

Cuando se introduce el disco en la unidad de disco, la corredera indicada con la letra F en la figura 1 de desplaza hacia atrás, permitiendo a la cabeza magnética encargada de la lectura-escritura en la superficie del disco, tomar contacto con ella a través del taladro alargado C.

El cabezal se mueve radialmente aldelante y atrás a lo largo de esta abertura, para poder acceder a cualquier punto del disco.

En este sentido debemos resaltar la gran rapidez de maniobra que implica el doble movimiento de giro del disco y radial de la cabeza, con el consiguiente ahorro de tiempo en la lectura-escritura de información.

D) * Los taladros denominados de alineación son los encargados de ajustar la carcasa del disco en su correcta posición dentro del *disk-drive,* haciéndolo, de esta forma, solidario con él.

E) * Finalmente, el taladro de índice permite al dispositivo adecuado de la unidad de disco «ver» dónde comienzan los sectores de las pistas, mediante una pequeña perforación existente en la superficie del disco. El dispositivo en cuestión espera hasta detectar la perforación citada y, a partir de aquí, lleva la cuenta de los **bytes** de información.

Los conceptos de sector y pista se estudian más abajo.

Veamos ahora la forma en que se organiza el almacenamiento de la información sobre la superficie magnética del disco, y qué y cómo lo maneja.

Al comprar un disco flexible o compacto, lo que realmente hemos adquirido es una lámina de plástico circular, recubierta de un material susceptible e ser magnetizado, ofreciéndonos su superficie para ser usada como soporte de información. Algo de eso se dijo la principio de este epígrafe, pero se omitió que, por lo demás, no sirven para nada... hasta que se les somete el proceso de **formateado.**

La expresion **formateado**es una anglicismo derivado de la voz inglesa **format**, con lo cual se quiere dar a entender que, mediante éste, el disco va a adquirir determinadas características. FORMAT es un programa de utilidad tratado en otro lugar detalladamente.

Bien. Pero, ¿qué ha cambiado en el disco, después del formateado?

ESQUEMA DE UN DISCO FORMATEADO POR UN AMSTRAD

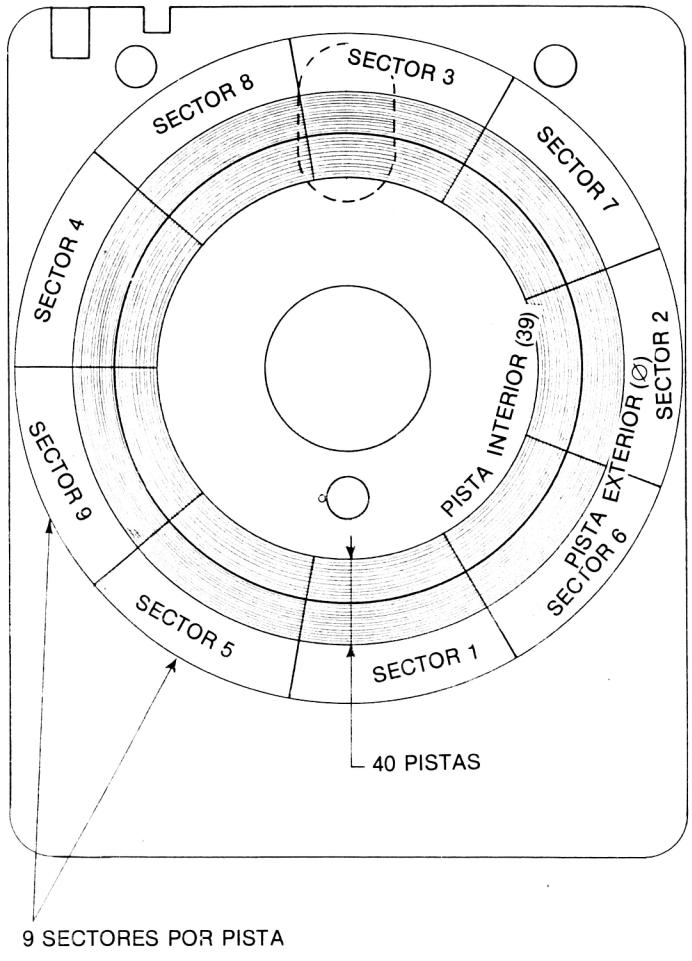


Figura 2: DISCO FORMATEADO (INTERIOR)

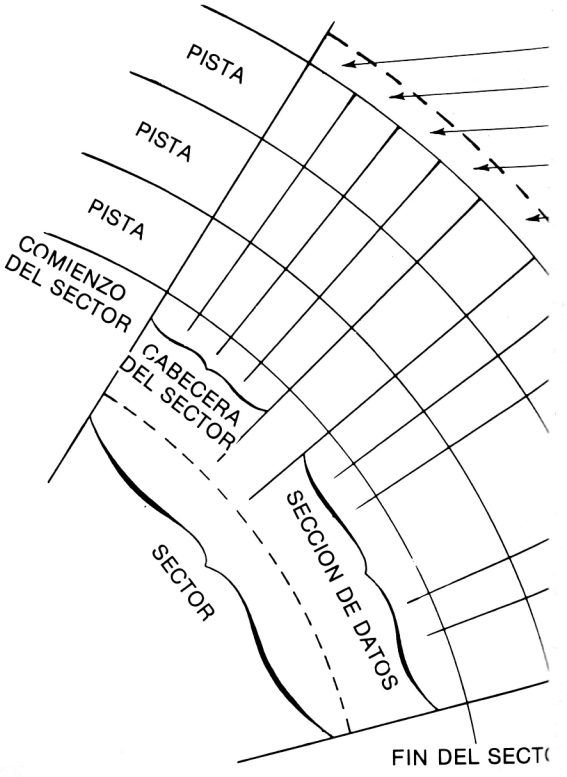
Resumen para un disco formateado por un AMSTRAD.

* 40 pistas. Numeradas de 0 (exterior) a la 39 (interior)
* 9 sectores por pista > <360 sectores por cara.
* 512 **bytes** de espacio para datos por sector = 4608 bytes pora pista > <4,5 bytes.
* 64 ficheros, como máximo, se pueden almacenar en un disco, pero aquellos ficheros que ocupen más de 16 K bytes reducirán este número en una unidad por cada 16 K.
* 1024 bytes que, como mínimo, ocupará un fichero. También denominado tamaño del bloque.
* La capacidad del disco variará de 180 K bytes a 171 K bytes, según el disco se formatee con los comandos CP/M incluidos, o sólo sea utilizados para datos.

ESQUEMA TIPICO DE LA DISTRIBUCION LONGITUDINAL DE UN SECTOR

1.— Comprueba si la velocidad de rotación es la adecuada.
2.— Identifica pista.
3.— Indentica sector.
4.— Comprueba número de bytes grabados en la cabecera (check-sum).
5.— Longitud en blanco para separar cabeza.
6.— Similar a 1.
7.— Idenfica siguiente sector.
8.— Información almacenada.
9.— Similar a 4 pero relativo a la Sección de datos.
10.—Longitud en blanco para separar el Sector del siguiente.

ESQUEMA TIPICO DE LA DIST



Para describir el cambio supondremos que estamos trabajando en una máquina específica:el AMSTRAD.

En primer lugar, la cabeza magnética de la unidad de disco escribe cierta información, borrando cualquier otra que pudiera existir, en una serie de anillos circulares denominados **pistas.** El AMSTRAD configura 40 de estas pistas, numeradas del 0 (la más externa) al 39 (la más interna).

Cada **disco** a su vez, es dividido en un número predeterminado de sectores circulares de igual superficie entre sí.

Por extensión denominamos **sector** a los diferentes trozos de pista comprendidos en un sector (bloque). El AMSTRAD los fija en 9, numerados del 1 al 9, y distribuidos según indica el dibujo 2.

En la figura 3 se puede ver un esquema típico de la información depositada en un sector cualquiera, después del formateado.

De toda ella lo que más nos interesa es la longitud disponible para datos, equivalentes a 512 bytes en el AMSTRAD. Físicamente aquí es donde se "memoriza" la información procedente del computador y recibida a través de la oportuna transmisión; el resto del espacio del sector está destinado a contener la información que permitirá al DOS el control sobre todos y cada uno de los sectores, y, finalmente, el disco en su calidad de soporte externo de memoria.

Observese, en el dibujo 2, que tras el formateado, sólo un tercio de la superficie de cara cara es utilizada.

En el AMSTRAD, en términos de transmisión entre memoria RAM y disco, la unidad más pequeña de información que se puede manipular es un sector o, lo que es lo mismo, **512 bytes.** Para trabajar a nivel de bytes hay que pasar del disco a RAM el sector que le contenga y allí manipularlos.

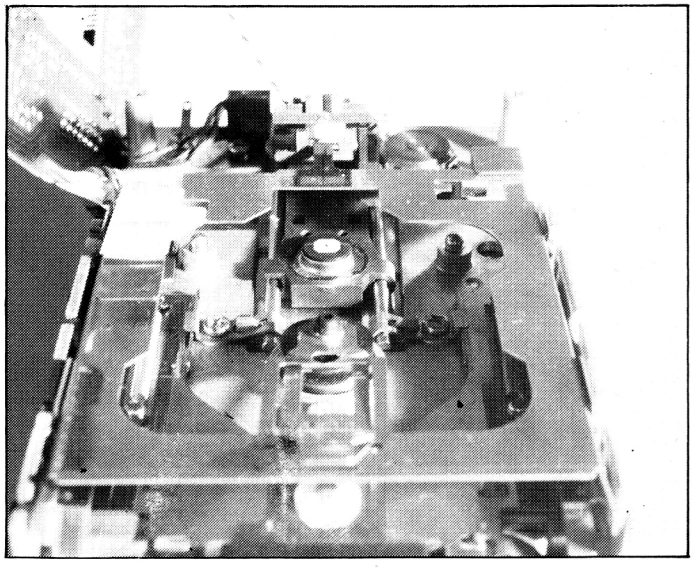
DISK-DRIVE

Con estos conocimientos en nuestro haber, vamos a aproximarnos al dispositivo periférico como **disk-drive** o unidad de disco.

Dediquemos una líneas a la **pista catálogo o directorio.**

Esta pista suele ser la central, como muestra la figura 2, o la número 2 contada de sde el exterio (0), y su misión es, contener la información necesaria y suficiente a disposición del DOS, para que este pueda, en su momento y según los casos, organizar los procesos de lectura y escritura en el propio disco. En MS-DOS, el directorio ocupa los sectores 4, 5, 6 y 7 de la pista ∅.

El Sistema operativo, para cumplir esta misión, al almacenar en un disco un nuevo fichero, lo cataloga,



RIBUCION LONGITUDINAL DE UN SECTOR

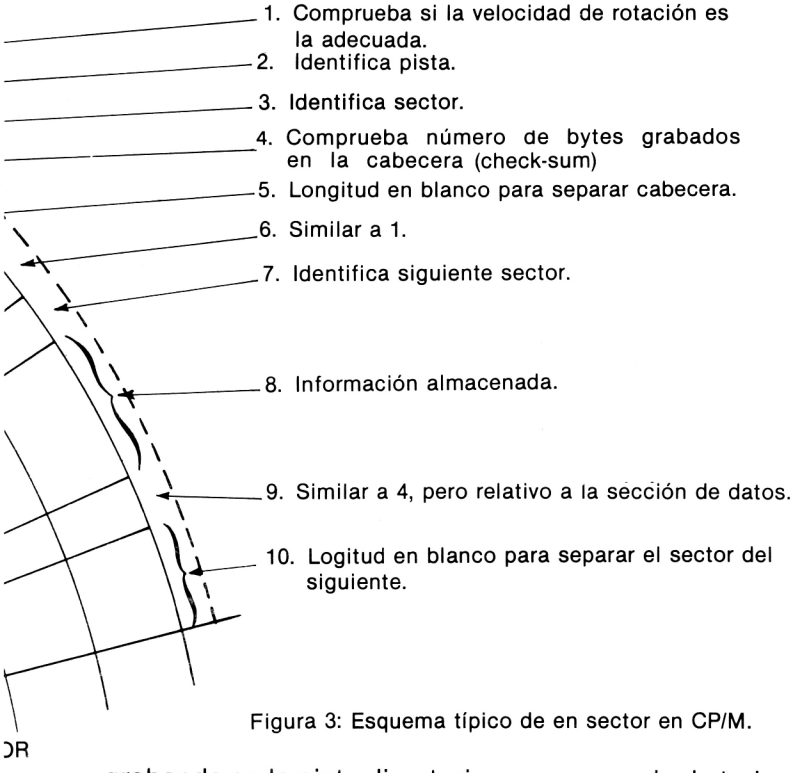


Figura 3: Esquema típico de en sector en CP/M.

)R

grabando en la pista directorio una secuencia de treinta y dos **bytes**, con lo cual, y entre otras cosas que se verán con detalle más adelante, se deja constancia del nombre del fichero y de la posición y espacio que ocupa.

En el lenguaje cotidiano, inglés, la estructura mental de la palabra **drive**, soporta significados tales como **conducir, transmitir** e **impulsar**, por tanto, en términos informáticos, deberíamos interpretar que un **disk-drive** es un dispositivo para transmitir datos entre un computador y un disco, y, además, maneja —conduce e impulsa— el propio disco, para lo cual está dotado de la electrónica y mecánica necesaria.

Así pues, un disk-drive es un dispositivo electromecánico que permite el intercambio de información entre el computador al cual está conectado, y un disco instalado en él.

Nosotros nos referiremos a este periferico como a una unidad de disco.

Dentro de la unidad de disco, no podemos decir que haya una parte más importante que otra ya que del preciso y acorde trabajo de todas ellas, depende la eficacia de este periférico.

No obstante, la cabeza magnética parece la pieza del sistema y a ella nos referiremos en primer lugar.

Una cabeza lectora-grabadora es, esquemáticamente, un anillo metálico con una abertura en un lugar, y en el opuesto una bobina. Cuando por esta bobina pasa una corriente eléctrica, en la abertura se genera un campo magnético cuya intensidad variará conforme a determinadas alteraciones en las características de la corriente que circule por la bobina.

En sentido contrario, si en las proximidades de la abertura se producen variaciones magnéticas, en la bobina se genera una corriente eléctrica.

Al hacer girar un disco en el **disk-drive**, estamos poniendo a disposición de un artificio capaz de magnetizar, tal cual es la cabeza de lectura/escritura, una superficie susceptible de ser magnetizada.

Cuando se pretende grabar —escribir— alguna información en el disco, lo que hacemos, mediante las instrucciones oportunas es obligar al computador a emitir una serie de impulsos eléctricos —bits— que, en función de sus características —ausencia/presencia de energía—, producen en la cabeza variaciones del campo magnético que quedan registradas en la superficie del disco.

Si, por el contrario, se desea recuperar la información almacenada en un disco, le pondremos a girar para que, al pasar enfrente de la cabeza, las partículas que componen la superficie magnética de la cara en la cual se quiere leer —magnetizadas con diferente intensidad en el proceso de grabación explicado anteriormente—, produzcan una corriente eléctrica cuyas variaciones serán transmitidas al computador.

Con el fin de estas variaciones respondan a un mismo criterio, es imprescindible que el disco gire a una velocidad constante, lo cual se consigue gracias a una electrónica y una mecánica sofisticadas.

Si estas variaciones alcanzan ciertas características estamos en presencia de un 1, en particular otro caso de un 0.

Por otra parte, y como ya sabemos, tanto el cero como el uno, son **bits** o unidades elementales de información y, una sucesión de bits, almacenados en forma de **bytes,** es, en definitiva, el contenido de un disco.

La cabeza de lectura está situada en un cabezal que se desplaza hacia delante y hacia atrás en la dirección del radio del disco, y a través del taladro alargado que a este fin tiene la funda de esta. El motor encargado de este movimiento es de extrema precisión consiguiendo desplazamientos, sobre la superficie del disco, muy pequeños y calculados.

En el caso de intentar la grabación de una información en disco, la primera acción que lleva a cabo el cabezal, es desplazarse hasta situar la cabeza sobre la pista catálogo, para que aquella "lea" el contenido de ésta y, determine si hay algún fichero con el mismo nombre y sitio en el disco.

La pista catálogo, contiene todos los datos necesarios para controlar la información que, hasta el momento esté grabada en el disco.

En el supuesto de haber espacio suficiente, se inicia la grabación en un sector vacío y si son necesarios más se utilizan nuevos sectores. Los datos oportunos para controlar esta nueva información a grabar, son escritos en la pista catálogo y en los propios sectores.

Un proceso similar se utiliza para recuperar la información contenida en el disco. Una vez localizada la pista y sector donde comienza, se lee esta, y si ocupa más de un sector se pasa al siguiente hasta completar la lectura de todo el espacio ocupado.

En el proceso de borrado de una información, los datos de esta desaparecen de la pista catálogo, dando por libres los sectores que estuvieran ocupados.

Todas estas funciones, y otras muchas, están dirigidas por el DOS.

EL BASIC DE

COMO SE LISTA UN PROGRAMA

Hacer, o sacar, un listado de un programa ya introducido en el computador, como es el caso del ejemplo anterior, consiste en hacer imprimir en pantalla todas las líneas que lo componen y en el orden de menor a mayor, según sus números.

El BASIC dispone del comando LIST que cumple esta función; más adelante se verá en profundidad, de momento vamos a aplicarlo al pequeño programa que hemos tecleado. Escribe LIST seguido de ENTER y verás aparecer el listado sin la línea 20.

COMO SE EJECUTA UN PROGRAMA

El pequeño programa anterior —actualmente con dos líneas: la 10 y la 30— está en memoria, pero nada le obliga al computador a ejecutarlo, o, dicho de otro modo, a obedecer las instrucciones que lo componen.

El comando BASIC encargado de esto es RUN, del cual anticipamos lo dicho con objeto de ver el efecto que causa sobre nuestro ejemplo.

Escribe RUN seguido de ENTER.

Lo primero que salta a la vista es el mensaje de error de sintaxis que emite el computador, refiriéndose a la línea 30, la cual, por otra parte, aparece completa a renglón seguido, y con el cursor a continuación del número de línea.

Esto es lógico si consideramos que el comando BASIC correcto es PRINT y no PRRINT. La acción lógica subsecuente por nuestra parte será borrar una de las erres que sobran; para ello desplacemos el cursor, mediante la teclar marcada >, hacia la derecha y hasta colocarlos en la posición elegida. Ahora, basta con apretar la tecla marcada CLR para que el carácter, sobre el cual está el cursor, desaparezca.

Pulse ENTER y pida un nuevo listado (LIST y ENTER).

Verá que el defecto está subsanado.

Ejecute el programa (RUN y ENTER) y el programa será obedecido sin más problemas. De una forma deliberada hemos introducido otros errores, ahora relativos a las primeras frase de el Quijote. Así, por ejemplo, en la línea 10 nos sobra una r en lugarr. Teclee EDIT 10 seguido ENTER y verá aparecer la línea 10 con el cursor sobre el primer carácter. Desplace el cursor con la tecla adecuada (>) y proceda como en el caso anterior.

En la línea 30, tenemos una doble modificación, ya que por una parte hay que añadir **no** entre **nombre** y la palabra que la sigue, y, por otra, debemos separar **quiero** de **acordarme.**

Empecemos por editar la línea 30 (EDIT 30 y ENTER). Cuando la línea en cuestión haya aparecido, desplacemos el cursor hasta coiocarlo sobre la q, y una vez allí teclea no y un espacio. Después coloca el cursor sobre la primera a de **acordarme** y da un espacio. Cuando todo ha sido hecho, pulsa ENTER y pide un nuevo listado. Comprobarás que todas las correcciones han sido efectuadas.

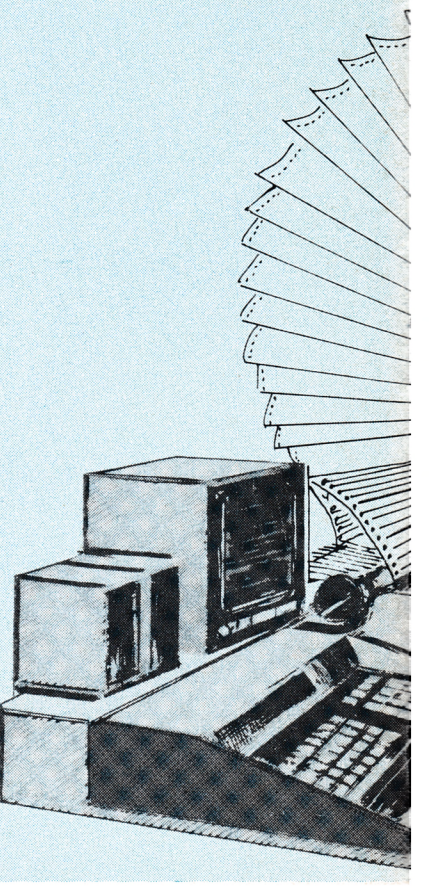
Hay otra forma de corregir lista dos, basada en el cursor de copia. Para ver esto en acción, supongamos nuevamente las tres líneas iniciales del programa ejemplo, errores incluidos.

Cuando hayas terminado de teclear el programa, el cursor estará situado al comienzo de la siguiente línea disponible.

Comencemos las correcciones, por el nuevo sistema, en la línea 10.

Manteniendo SHIFT apretada, pulse la tecla > las veces que necesite como para situar el nuevo cursor que aparece —cursor de copia— sobre el 1 del comienzo de la línea 10. Ahora aprieta la tecla donde figura COPY y verás desplazar este cursor sobre los caracteres de esta línea y, simultáneamente, el cursor normal se desplazará dejando aparecer los mismos caracteres. Para sobre la siguiente posición al carácter que desees borrar, la r en este caso, y pulsa DEL. Finalmente corre el cursor (con COPY) hasta el final de la línea y aprieta ENTER. La corrección ha sido efectuada.

Atención. Si el cursor de copia no se desplaza hasta el final de la línea, la parte de la línea situada a la derecha de este cursor, queda borrada al pulsar ENTER.



Es evidente que, para borrar una línea completa, es más cómodo el primero de los sistemas propuestos, no obstante, este es un ejercicio interesante ya que, al desplazar el cursor de copia hasta el extremo derecho de la línea 20, y proceder a su borrado mediante DEL, si el cursor llega a "eliminar" los números de línea, el proceso se cancela y la línea se mantiene, en otras palabras, sólo se pueden borrar las sentencias, o parte de ellas, situadas en una línea, nunca el número de línea.

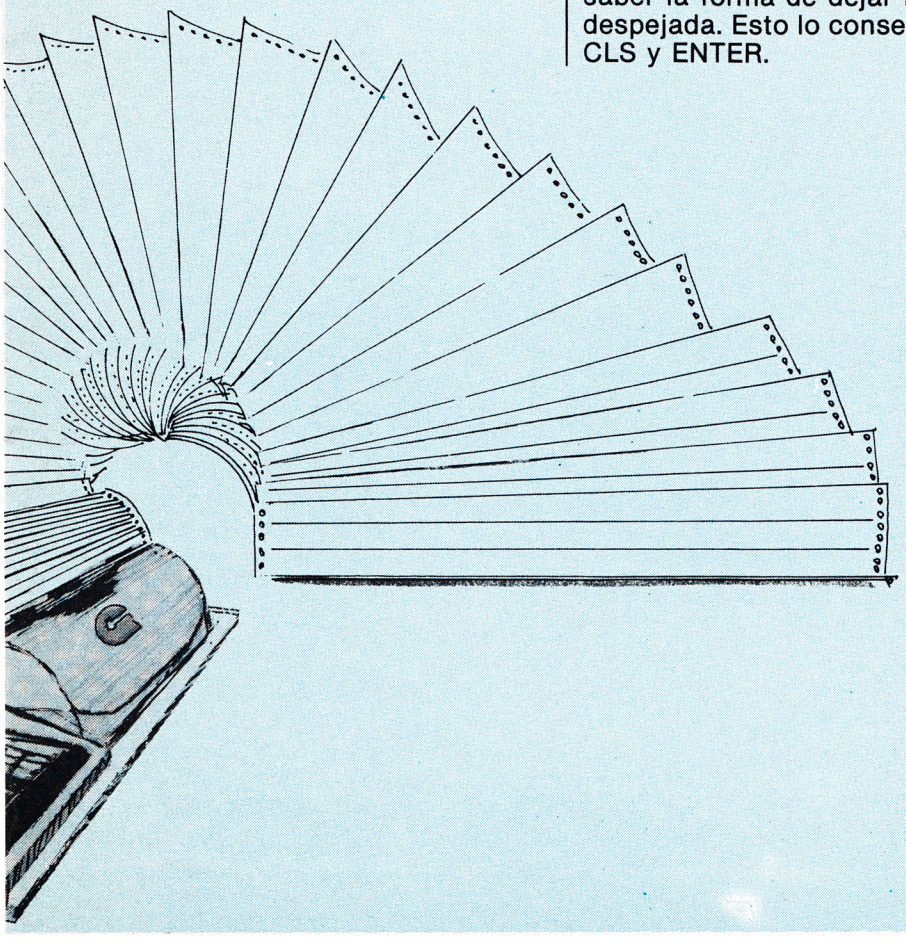
También debemos observar que la tecla CLR no es operativa con el cursor de copia.

Prueba a efectuar las modificaciones de la línea 30 mediante el cursor de copia, siguiendo las indicaciones anteriormente dadas.

COMO SE BORRA UN PROGRAMA EN MEMORIA Y COMO SE LIMPIA LA PANTALLA

Una vez que hayas hecho tus pruebas con el programa anterior y si, por alguna razón, deseas tener la memoria libre de cualquier listado, puede usar el comando NEW (seguido de ENTER, claro está), con la cual, si pruebas a pedir un listado, verás que no hay tal.

A modo de anticipación, debemos saber la forma de dejar la pantalla despejada. Esto lo conseguirás con CLS y ENTER.



CONEXION Y PUESTA EN MARCHA

Para hacer operativo tu sistema de computación AMSTRAD, necesita conectarlo correctamente.

Mantener el equipo desenchufado de la red mientras procedes a interconectarlo y siempre que no esté en uso.

El primer paso debe consistir en comunicar el computador con el monitor —o su televisión.

Las conexiones encargadas de esto, están en la parte posterior de la carcasa y a la altura del teclado numérico y marcadas con:

MONITOR
5 V DC
12 V DC

Tanto en el supuesto de que dispongas de un monitor AMSTRAD, como en el caso de que, desees utilizar la pantalla de tu TV, a través de un modulador, cada clavija tiene una sola conexión posible. Si se trata de monitor bastará con unir computador y monitor mediante los tres cables que, de uno u otro dispositivo emergen, conectar el monitor a la red y apretar el botón POWER situado en la esquina inferior izquierda del frontal de la pantalla. Si se usa el modulador, la explicación anterior es válida, pero relativa a este dispositivo; habrá claro está, que unir el modulador con la TV mediante el cable dirigido a la entrada de antena del televisor.

Dado que el transformador está situado en el interior del modulador, la toma de energía se efectuará conectando éste a la red.

En función del modelo de AMSTRAD de que se disponga, siempre existirá la opción de conectar un magnetófono a cassettes y un "floppy". Los lugares de conexión son evidentes y no requieren mayor explicación.

Las impresoras que de forma directa, se pueden utilizar deben incor-

porar una **interface** centronics. Su conexión en el computador está indicada con **PRINTER** en la parte posterior.

La función que cumple cada elemento del ordenador es la siguiente:

Computador: EL AMSTRAD contiene toda la electrónica necesaria para el manejo y almacenamiento de la información y posee la adecuada conexión, interna, al teclado.

Teclado: Gracias al teclado se consigue una comunicación directa entre el usuario y el ordenador, introduciendo información a través de él. Este dispositivo es sólo de entrada.

Televisor o **monitor:** Este dispositivo es sólo de salida, ya que permite al computador emitir sus respuestas por la pantalla del mismo.

Cassette o **disquete:** Es un medio de almacenamiento externo que permite transferir información desde una cinta magnética al computador y, en sentido contrario, desde el computador a la cinta.

Impresora: Es un dispositivo de salida que permite al usuario proveerse de copias en papel de información contenida en el computador.

CONSTANTES

Las **constantes** son aquellos elementos de un algoritmo cuyo valor permanece invariable.

En términos de programación hay dos clases de constantes: **numéricas** y **alfa numéricas.**

Las constantes numéricas pueden ser cualquier valor numérico positivo o negativo, entero o decimal. Como por ejemplo 3 ó —33,33.

Las alfanuméricas están compuestas por cualquier concatenación de caracteres escritos entre comillas.

EJEMPLOS: "AMSTRAD"
"24 de diciembre"

VARIABLES

Las variables son aquellos elementos de un algoritmo capaces de tomar cualquier valor.

En programación hay dos tipos de variables: **numéricas y alfanuméricas.**

Las primeras quedan representadas por cualquier cadena de caracteres con la única condición de que el primero de todos ellos sea una letra. EJEMPLOS: LET A = 1 o LET A2 = 1520.

Las alfanuméricas o de caracteres deben ser representadas por una cadena seguida del símbolo $, para que el computador la distinga, de las numéricas.

El valor que se asigne a un variable alfanumérica, debe ir entrecomillas.

EJEMPLO:
LET A$ = "AMSTRAD" o LET R$ = "24 de diciembre"

Denominamos genéricamente por nombre de la **variable** a la representación de las mismas, así, en los ejemplos anteriores, tenemos los siguientes nombres de variables:

A, A$, A$ y R$.

OPERADORES

Los operadores son los símbolos que determinan las operaciones que se han de realizar entre valores.

Hay cuatro tipos de operadores: **Aritméticos, de relación, lógicos, y funcionales.**

EXPRESIONES

Una **expresión** es un conjunto de valores relacionados entre sí por medio de operadores, cuyo resultado final es una cantidad.

Con respecto a las variables se dijo que eran elementos capaces de

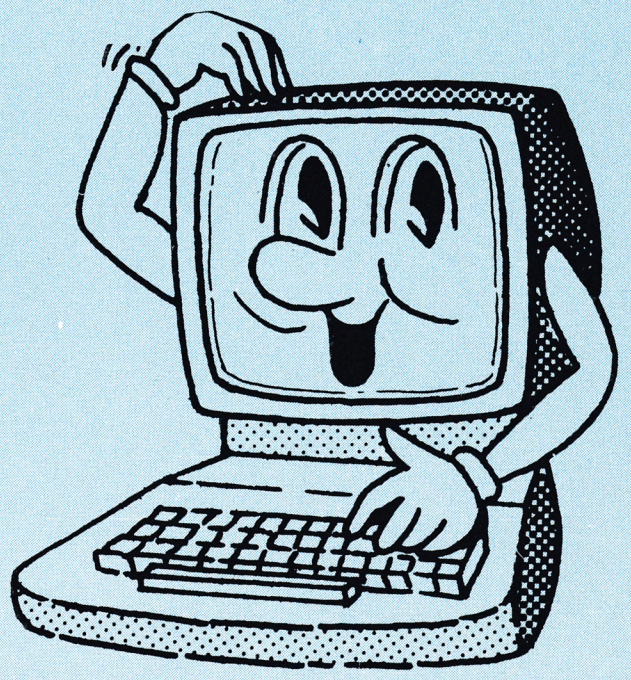
ALGORITMO

Un algoritmo es un conjunto de operaciones perfectamente definidas, gracias a las cuales se puede resolver un problema partiendo de unos datos conocidos.

Por ejemplo: Para hallar el área de un triángulo hay que multiplicar lo que mide la longitud de su base (b) por lo que mide la longitud de su altura (a) y dividir el resultado por 2. Esto mismo expresado en forma esquemática del proceso que conlleva es:

1.° R1 = b*a
2.° R2 = R1/2

Estas son las dos operaciones (o pasos) que exige el algoritmo propuesto.



contener un valor, una expresión, sin embargo, **es** un valor.

EJEMPLO: Definir una expresión que determine el valor de un viaje en taxi, sabiendo que cada "salto" del contador cuesta 25 Ptas.

Si admitimos que la variable X representa el número de "saltos" de contador, el importe del viaje vendrá dado por la expresión: 25*X.

El operador que relaciona ambos valores es "*".

Claro está, para cada valor de X tendremos un nuevo valor de la expresión 25*X.

Las expresiones matemáticas se escriben en BASIC prácticamente igual que con lápiz y papel, pero con las variables impuestas por los símbolos que figuran en el teclado del ordenador. Tal es el caso, por ejemplo de X^3 (equis elevado a 3), que nos obligará a teclear x↑3.

El orden de prelación de las diferentes operaciones matemáticas se verá más tarde, pero, antes de llegar al mismo, es importante hacer observar unas sutiles diferencias que existen entre escribir una expresión matemática a mano o en un computador. Supongamos
que deseamos transcribir la expresión $\frac{a + b}{2c}$ de
forma entendible para el ordenador. para lograrlo, no debemos olvidar ningún signo —como * entre 2 y c en el denominador— y no magnificar la capacidad de la máquina ya que:

— Si escribimos a + b/2*c, el ordenador interpretará $\frac{a + b}{2}$ c.

— Si escribimos (a + b)/2*c, el ordenador interpretará $\frac{a + b}{2}$ c

— Sólo podrá interpretar correctamente la expresión citada si escribimos (a + b) / (2*c).

SENTENCIAS

Una sentencia es una frase, escrita conforme al vocabulario y sintaxis de lenguaje de programación, que transmite al computador una orden.

OPERADORES ARITMETICOS

Se denominan así porque actúan entre valores aritméticos, siendo el conjunto de los símbolos que los componen y su orden de prioridad establecidos por el ordenador los siguientes:

Prioridad	Operación	Operador
Máxima	exponenciación	↑
	Módulo	MOD
	multiplicación y división	*/
	división entera	\
mínima.	suma y resta	+ —

Para operadores de la misma prioridad, la máquina ejecuta las operaciones de izquierda a derecha.

Las operaciones dentro de un paréntesis las efectúa primero y siguiendo el orden de prioridad anterior.

Con MOD se obtiene el resto de una división.

EJEMPLO:

(6 * 5 ↑ 3 /(4+ 2) /(3 – 1)

1° Calcula la potencia 5 ↑ 3 (5^3).
2° Multiplica el resultado por 6.
3° Calcula 4 + 2.
4° Divide el producto (2^o) entre la suma (3^o).
5° Calcula 3 - 1.
6° Divide el producto (2^o) entre la diferencia (6^o).

EJEMPLO:

PROGRAMA:

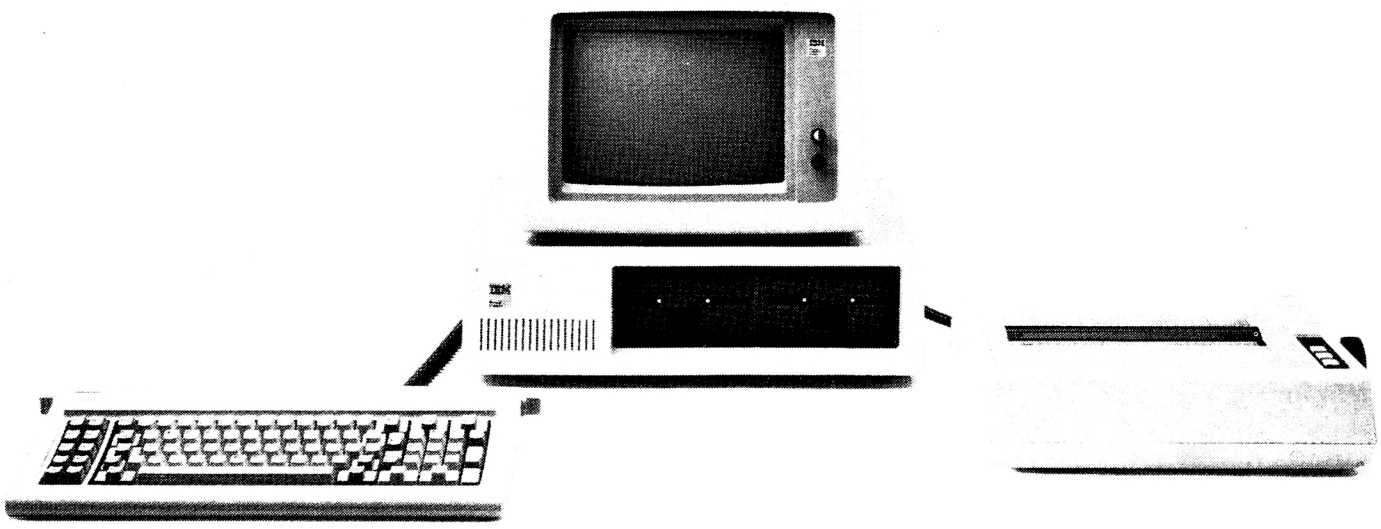
```
10 LET A$ = "ABC"
20 LET B$ = "DEF"
30 PRINT A$ + B$
```

COMENTARIO

Con las líneas 10 y 20 asignamos valores a las variables de caracteres A$ y B$.

Con la línea 30 ordenamos la impresión del contenido actual de A$ y B$, justamente uno a continuación del otro.

El resultado de ejecutar este programa será: ABCDEF.



OPERADORES DE RELACION

Estos operadores sólo actúan en proposiciones entre dos operandos, cuyo resultado únicamente puede ser CIERTO, que se representará por 1, o FALSO, que se representar por 0.

Sus símbolos son: igual que . . . =
distinto que . . . . . ><
menor que . . . . . <
mayor que . . . . . >
menor o igual que . . . . . <=
mayor o igual que . . . . . <=

Si un operador de relación compara dos expresiones en las que intervienen operadores aritméticos, antes de efectuar la comparación, actúan los operadores aritméticos.

EJEMPLO:

Determinar si 2 + 5 es menor que $(15 + 4)^3 / 20250$.

SOLUCION:

Es evidente que, sin efectuar las operaciones aritméticas que determinan el valor de cada expresión, es difícil responder a la proposición anterior.

Las cadenas de caracteres pueden ser comparadas con estos mismos operadores. Para hacerlo se cotejan, carácter a caracter, ambas cadenas, valorando cada caracter de acuerdo con su código ASCIL. Más adelante se estudiará todo lo referente a los códigos, baste por ahora saber que cada caracter tiene su propio valor.

OPERADORES LOGICOS

Responden directamente a las funciones lógicas del álgebra de BOOLE.

Los **operadores lógicos** actúan entre operandos en los que, a su vez, intervienen operadores de relación. Esto quiere decir que los operandos de los operadores lógicos tendrán siempre el valor 1 (CIERTO) o 0 (FALSO) y, por consiguiente, los **operadores lógicos** lo responden 1 ó 0 (CIERTO o FALSO).

Los **operadores lógicos** NOT, AND y OR que se estudiarán con detalle más adelante.

OPERADORES FUNCIONALES

Estos son algoritmos que residen en el propio BASIC y operan sobre datos suministrados al computador por medio del teclado o el programa, obteniéndose, como resultado un valor.

COMA FLOTANTE

El ingeniero español Leonardo Torres Quevedo fue el primero en aplicar el concepto de la **coma flotante** a los computadores.

Cualquier número, positivo o negativo, puede ser representado en forma exponencial, para ver esto, supongamos que al medir una distancia escribimos 525 metros.

Evidentemente, podríamos haber representado lo mismo con: $0.525*10^3$ y esto, en definitiva, es lo que representa el concepto de la coma flotante.

EJEMPLO: PRINT 124* 1000 000

Al ejecutar esta instrucción en modo directo el ordenador nos retorna 124000000.

El mismo resultado habríamos obtenido con PRINT 1.24 E 8.

La mantisa queda representada por 1.24 y el índice por 8.

Este sistema permite, además de un medio económico de codificación, un amplio campo de representación numérica.

FICHAS DEL AMSTRAD

CODIGO	TIPO DE ERROR	COMENTARIOS
1	Unexpected NEXT	NEXT inesperado
2	Syntax Error	Error de síntasis
3	Unexpected RETURN	RETURN inesperado
4	DATA exhausted	DATA agotados
5	Improper argument	Argumento impropio
6	Overflow	Desbordamiento en cálculo
7	Memory full	Memoria llena
8	Line does not exist	Línea inexistente
9	Subscript out of range	Subíndice inadecuado
10	Array already dimensioned	Matriz ya dimensionada
11	Division by zero	División por cero
12	Invalid direct command	Comando incorrecto
13	Type mismatch	Tipo inadecuado
14	String spacefull	Espacio para cadenas completo
15	String too long	Cadena demasiado larga
16	String expression too complex	Cadena demasiado compleja
17	Cannot CONTinue	No puede continuar
18	Unknown user fuction	Función usuario desconocida
19	RESUME missing	Se alcanza el fin del programa antes del resumen de errores.
20	Unexpected RESUME	Resumen inesperado
21	Direct command found	Encontrado comando directo
22	Operand missing	Falta operando
23	Line too long	Línea demadiado larga
24	EOF met	Fin de fichero encontrado
25	File type error	Error en el tipo de fichero
26	NEXT missing	FOR sin NEXT
27	File already open	Fichero ya abierto
28	Unknown command	Comando desconocido
29	WEND missing	WHILE sin WEND
30	Unexpected WEND	WEND inesperado
31	File not open	Fichero no abierto

El usuario puede generar su propio mensaje de eror, otorgando a este, en **expresión,** un valor entre 33 y 255.

EVERY *¡expresión 1! ¡expresión 2! GOSUB ¡número!*

Llama a la subrrutina situada en la línea **número,** cada vez que el tiempo dado por **expresión 1** se consume.

Cuatro instrucciones EVERY pueden trabajar simultáneamente con la prioridad que determina **expresión 2,** siendo la máxima 3 y la mínima 0. La unidad de tiempo es 0,02 segundos.

```
  5 INPUT x: EVERY 100/x, PI/PI GOSUB 40.
  10 EVEREY 100,2 GOSUB 50
  20 PRINT "2"; GOTO 20
30 END
40 PRINT "####": RETURN
50 PRINT "$ $ $ $"; SOUND 1,30: RETURN
```

FILL ¡expresión!

Rellena una zona delimitada de gráficos de color dado por **expresión.**
Si la zona no está perfectamente cerrada, el rellenado continúa.

```
10 MODE 1: MOVE 0,50: DRAW 637,50
20 MOVE 0,0: x = 2
30 FILL 28/x
```

FIX ¡(expresión)!

Elimina los decimales de **expresión**

```
? FIX (PI)
3.
```

FOR ¡variable = expresión 1! TO ¡expresión 2! STEP ¡expresión 3!

Establece un bucle entre la línea donde aparece esta instrucción y la instrucción NEXT, el cual se repite desde que la **variable** toma el valor dado por **expresión 1,** hasta que alcanza el dado por **expresión 2,** con la cadencia determinada por **expresión 3.**

```
10 x = 10
20 FOR y = 100 / x TO 10* x STEP 20/x
30 PRINT y;
40 NEXT
```

FRAME

Armoniza el desplazamiento de un móvil por la pantalla.

FRE (expresión)

```
FRE ("")
```

Esta función guarda la memoria libre disponible para el BASIC.

GOSUB ¡número!

Dirige la ejecución del programa a la subrrutina situada en la línea cuyo número es **¡número!**

GOTO ¡número!

Dirige la ejecución del programa a la línea cuyo número es **"número"**

GRAPHICS PAPER ¡expresión!

Fija el color de fondo de los gráficos, según el valor dado por **expresión**

```
5 INPUT x
10 MODE 1: MASK 7: GRAPHICS PAPER x
20 MASK 255: MOVE 0,50: DRAW 640,50
40 GOTO 5
```

GRAFICS PEN expresión 1 expresión 2

Fija el color de tinta de los gráficos, según el valor dado en **expresión 1.** La **expresión 2** puede ser 0 ó 1. Cualquier de las dos expresiones puede ser omitida, pero no las dos.

```
10 INPUT x,y
20 GRAPHICS PEN x,y
30 DRAW 639,0
```

HEX$ expresión 1 expresión 2

Convierte la **expresión 1** en una cadena representativa de hexadecimal equivalente con tantos dígitos como indique **expresión 2.**
Si **expresión 2** no se da, o es demasiado pequeña, la cadena cubre los dígitos hexadecimales necesarios.

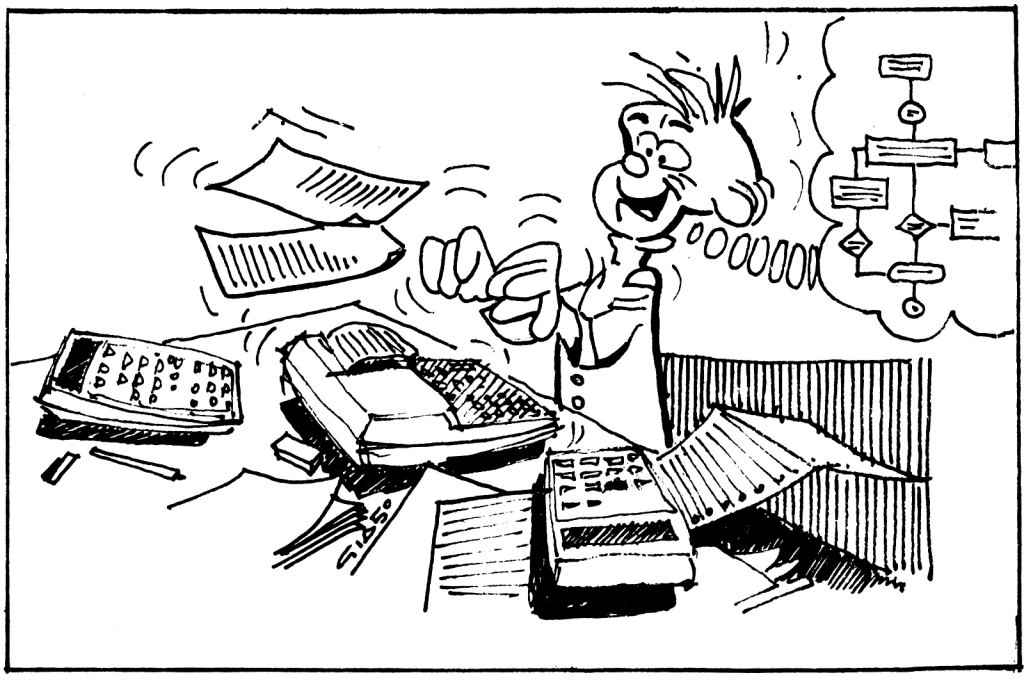
```
10 INPUT x, y
20 PRINT HEX$ (x,y)
```

HIMEN

Esta función guarda la dirección de memoria más alta disponible para el BASIC PRINT HIMEN
42619.

PROGRAMAS COMENTADOS EN BASIC

JUEGO DE BARCOS



El juego que proponemos es una versión simplificada del conocido juego de los barquitos.

La cuadrícula de juego se compone de 6 × 6 cuadritos, en la que se sitúan aleatoriamente 3 submarinos, (un solo cuadro cada uno). Pueden estar juntos o separados e incluso pegados a las pareces del cuadro.

El juego consiste en tratar de hundirlos en el mínimo número de jugadas. El ordenador nos irá indicando en cada jugada si las coordenadas del tiro (fila, columna), corresponden a "agua" o "hundido", en caso de disparar a un cuadro de forma repetida indicará "disparo ya realizado".

Al final del juego, una vez hundidos los tres submarinos, nos dará la calificación como jugador.

COMENTARIOS AL PROGRAMA:

LINEAS

10-40 Presentación y bucle de retardo (línea 40).

50 Dimensionado de una matriz de 6 × 6 (numérica), que emula la cuadrícula de juego, de manera que cada elemento de la matriz representa un cuadro del juego.

70-100 Situar barcos, las coordenadas de cada barco (fila y columna), se fian aleatoriamente mediante la línea 80, que genera dos números enteros comprendidos entre 1 y 6. Como al dimensionar la matriz, todos los elementos valen cero, a los seleccionados aleatoriamente se les da el valor 1, que representa una cuadrícula con submarino. Las que contienen un cero son agua.

La línea 90 comprueba que dos barcos no ocupen la misma casilla. El bucle termina al situar los 3 barcos (si se desean más variar el valor final del bucle).

120-150 Dibuja las líneas horizontales. Como el número de cuadros es 6, el de líneas será 7 (cuadros + 1). El paso del bucle es 16 ya que a cada cuadro en MODE 1 le corresponden 16 pixels o puntos elementales (alta resolución). El bucle se establece a partir de 16 para poder situar en la primera columna los números (1 al 6).

170-190 Este bucle es idéntico al anterior para dibujar las líneas verticales.

160 y 200-250 Imprimen los números de las columnas y de las filas, mediante la función LOCATE columna, fila, que fija el puntero de PRINT.

260-310 Entrada de las coordenadas del disparo y control de que éstas no sean mayores que 6. (Máximo número de filas y columnas). En caso de ser mayor vuelve a pedir la coordenada correspondiente (observar el cursor de INPUT en la pantalla).

320 Controla si la coordenada introducida lo fue ya anteriormente (tiro repetido). El valor 2 se fija cuando hay un disparo no repetido tanto si es agua como fuego, por la línea 380.

330 Control de "hundido", como a los barcos se les identificó en la matriz como 1, si el elemento de la matriz que corresponde a las coordenadas del disparo es uno, ahí se encuentra un barco. La variable BH contiene el número de barcos hundidos, a fin de controlar el fin de juego en la línea 370.

340 Control de "agua". Si el disparo no fue hundido ni repetido será "agua", representado en la matriz por el valor 0.

350 Contador de disparos.

360 Borrado de textos viejos, para iniciar una nueva tirada.

370-390 Control fin del juego y apunte de tirada (2 en la matriz). Vuelta al juego (continuación).

410-440 Final del juego y establecimiento de la clasificación según el número de tiradas empleado en hundir los 3 submarinos. Se pasa a esta fase del juego por la línea 370, cuando BH = 3.

MEJORAS AL PROGRAMA

El programa se ha construido pensando en su valor didáctico y no como un buen juego, por tanto el número de mejoras que pueden realizarse es grande, por ejemplo mayor cuadrícula, mayor número de barcos, barcos de más de una dimensión, mejor presentación, etc., algunas modificaciones serán fáciles de realizar, ¡anímese y hágalas! Otras serán más dificultosas, en cualquier caso inténtelo, es una buena forma de avanzar en programación.

Personal Computer GAMES

JUEGO DE LOS BARCOS
VERSION SIMPLIFICADA

```
20 MODE 1:PRINT "JUEGO DE BARCOS. VERSIO
N SIMPLIFICADA"
30  PRINT :PRINT "SE CONSIDERA UNA CUADR
ICULA DE 6*6      Y TRES BARCOS DE UN SO
LO CUADRO"
40 FOR K=1 TO 10000: NEXT K
50 DIM C(6,6):  ' Dimensionamiento matriz
 de juego.
60  ' ..... SITUAR BARCOS
```

```
70 FOR K=1 TO 3
80 FILA=INT(RND*6)+1 : COL=INT(RND*6)+1
90 IF C(FILA,COL)=1 THEN 80 ELSE C(FILA,
COL)=1
100 NEXT K
110 ' ..........DIBUJO CUADRICULA

120 CLS
130 FOR Y=16 TO 16*7 STEP 16
140 PLOT 16,Y:DRAWR 16*6,0
150 NEXT Y
160 LOCATE 2,25: PRINT "123456"
170 FOR X=16 TO 16*7 STEP 16
180 PLOT X,16:DRAWR 0,16*6
190 NEXT X
200 LOCATE 1,19: PRINT "1"
210 LOCATE 1,20: PRINT "2"
220 LOCATE 1,21: PRINT "3"
230 LOCATE 1,22: PRINT "4"
240 LOCATE 1,23: PRINT "5"
250 LOCATE 1,24: PRINT "6"
260 ' ..................JUEGO

270  LOCATE 1,1: PRINT  "COORDENADAS DEL
 DISPARO"
280  LOCATE 1,3: INPUT "FILA ";FILA
290 IF FILA >6 THEN 260
300  LOCATE 1,4: INPUT "COL  ";COL
310 IF COL >6 THEN 260
320 IF C(FILA,COL)=2 THEN LOCATE 10,10:P
RINT "DISPARO YA REALIZADO":GOTO 260
330 IF C(FILA,COL)=1 THEN  LOCATE 15,4:
PRINT "HUNDIDO!": LOCATE COL+1,FILA+18:
PRINT CHR$(24);"*";CHR$(24):BH=BH+1
340 IF C(FILA,COL)=0 THEN LOCATE 15,4: P
RINT "AGUA    ":LOCATE COL+1,FILA+18: PR
INT CHR$(24);" ";CHR$(24)
350 disparos=disparos+1
360 LOCATE 10,10:PRINT "
    "
370 IF BH=3 THEN LOCATE 10,10 : PRINT "F
IN DEL JUEGO":GOTO 410
380 C(FILA,COL)=2
390  GOTO 260
400 ' ..............  FIN DEL JUEGO

410 IF DISPAROS<25 THEN PRINT "ERES UN F
ENOMENO":GOTO 440
420 IF DISPAROS<30 THEN PRINT "NO ESTA N
ADA MAL":GOTO 440
430 PRINT "DEDICATE A OTRA COSA"
440 END
```

II.3 CONCEPTO DE PROCEDIMIENTO

Podemos pensar que cuando el ordenador está en "toplevel", está esperando órdendes. Estas órdenes pueden ser instrucciones directas, según se vio antes, o bien puede ser la orden para la ejecución de un programa. Al comenzar la ejecución de un programa, el ordenador abandona el nivel superior, y entra a niveles inferiores. Este es otro de los posibles modos de operación. Se llama **MODO DE EJECUCION.**

El concepto de procedimiento es uno de los más ricos de todo el lenguaje LOGO y, desde luego, da origen a toda su estructura. Dicho brevemente, un procedimiento es un programa escrito en Logo.

En la definición de un procedimiento se pueden utilizar todas las instrucciones primitivas del lenguaje, así como los procedimientos que ya se hayan definido. Esto permite confeccionar los programas desglosando acciones largas y complejas en grupos de instrucciones que realicen una parte específica: los procedimientos constituyentes del anterior.

Por ello, un modo usual de trabajo en Logo, es escribir procedimientos útiles o utilidades que se pueden almacenar en memoria externa, como los discos. Cada vez que necesitemos alguno de ellos basta cargarlo en memoria y utilizarlo como un primitivo más.

Si el ordenador está ejecutando un procedimiento-1 que ha comenzado desde el nivel superior, decimos que está en nivel 1. A su vez este , podría "llamar" a otro procedimiento-2 que, de este modo, comenzaría a ejecutarse cuando el anterior, en realidad, todavía no había concluido. Decimos entonces que el Logo está en nivel 2. Y así sucesivamente. Este tipo de situaciones pueden representarse esquemáticamente como vemos en la figura II.1. En ella, cada línea horizontal representa un nivel; la más alta, el nivel superior, y cada paso hacia abajo representa la llamada a un nuevo procedimiento. Sólo cuando el procedimiento-2 termina por completo su ejecución, puede continuar el procedimiento-1 que le había llamado.

II.4 DEFINICION DE PROCEDIMIENTO CON "TO" Y "END"

Hemos estudiado hasta aquí dos de los modos en que el Logo puede estar: "modo de órdenes directas" o "nivel superior", y "modo de ejecución". Otro modo en que puede estar es **MODO DEFINICION.** Se entra en este modo para definir un nuevo procedimiento.

Un procedimiento consta de tres partes: **NOMBRE, CUERPO** y **FINAL.** El nombre es una palabra que le identifica a todos los efectos. Para que una palabra pueda ser el nombre de un procedimiento, debe cumplir las siguientes condiciones:

— No puede ser el nombre de un primitivo.

— No puede coincidir con el nombre de un procedimiento ya definido.

— Su longitud ha de ser a lo sumo de 74 caracteres.

— Debe respetar todas las reglas de formación de las palabras.

El nombre de un procedimiento, en el AMSTRAD, puede estar formado con letras mayúsculas o minúsculas o una mezcla de ambas. Si dos procedimientos tienen el mismo nombre, pero uno en mayúsculas y otro en minúsculas, el ordenador los diferencia. La tecla CAPS LOCK, se utiliza como un interruptor para pasar de un tipo de letras a otro.

EJEMPLO II.3

— Nombres correctos de procedimientos:

a) dibujo
b) DIBUJO 2
c) Palabra.1

— Nombres incorrectos:

a) cs (porque es un primitivo)
b) pa/2 (porque el carácter "/" es un separador de palabras.
c) A B (porque incluye un espacio en blanco).

El cuerpo del procedimiento consite en todas las instrucciones que lo forman. Se escriben ordenadamente de arriba abajo y de izquierda a derecha. Entre cada dos instrucciones basta dejar un espacio en blanco, como entre cada dos palabras. Sin pulsar la teclar enter, se pueden escribir a lo más dos líneas.

Por último, el final del procedimiento indica sencillamente donde termina este.

Para situar a Logo en modo defincón, hemos de comenzar por decidir el nombre del procedimiento que vamos a definir. Supongamos que el nombre elegido es PALABRA. Entonces basta escribir desde el toplevel, to PALABRA
y pulsar enter después. Cuando el AMSTRAD entra en modo definición cambia el promopt para indicarlo: pasa "?" a ">". Todo lo que se escriba en modo definición lo tama el ordenador como constituyente del procedmiento que se está definiendo. Por consiguente no detecta ningún error; estos sólo se detectan en la ejecución. Tras cada instrucción pulsamos la tecla enter o bien dejamos un espacio en blanco antes de escribir la próxima.

Para que el ordenador abandone el modo definición, basta indicar que se ha llegado al final del procedimiento. Esto se hace escribiendo al comienzo de una nueva línea la palabra,

END

seguida de enter. Entonces Logo escribe en pantalla el mensaje,

PALABRA defined

que indica que el procedimiento llamado PALABRA forma parate ya de la memoria. El ordenador queda entonces en toplevel.

EJEMPLO II.4

— Si suponemos que el procedimiento llamado PALABRA está formado por las instrucciones INS1, INS2 e INS3, en la pantalla del ordenador veremos:

```
?to PALABRA
>INS1, INS2
>INS3
>end
?PALABRA defined
```

Obsérvese que se podía haber escrito INS2 en una línea distinta a la ocupada por INS1.

Otro modo de interrumpir la definición de un procedimiento es pulsando la tecla **ESC.** En este caso el procedimiento queda sin definir, cualquiera que sea el momento en que se haga. Esto obedece a un hecho general ya que:

"Al pulsar la tecla "esc", el Logo abandona lo que estuviera haciendo y regresa a toplevel, apareciendo en pantalla el mensaje **Stopped!**".

II.5 EDICION DE PROCEDIMIENTOS

Finalmente, el Logo puede estar en **MODO EDICION.** Este modo puede ser usado sistemáticamente para definir procedimientos. Permite definir más de un procedimiento a la vez, y permite modificar los procedimientos ya definidos, y presentes en memoria. Se estudia por completo en el capítulo V.

III.1 PRESENTACION DE LA TORTUGA

UNA de las cosas que cautivan desde el principio, cuando uno se aproxima al lenguaje Logo, es su capacidad para dar órdenes a una "supuesta" tortuga que le obedece ciegamente... si se le dan las órdenes correctamente. La tortuga, en efecto puede ser nuestra primera aproximación a lo que podría ser un pequeño **robot.**

Este robot "vive" en la pantalla de nuestro ordenador en la actualidad. Nada se opone a que de hecho sea el precursor de otras "tortugas" que sean capaces de moverse fuera de la pantalla: en nuestro mundo real. En realidad las tortugas actuales son sucesoras de otras que nacieron con el lenguaje, y que estaban conectadas por cables al ordenador. Estaban provistas de unas plumas de diversos colores y de goma de borrar, y hacían dibujos sobre el suelo obedeciendo las órdenes que recibían de su amo: claro está, le hablaba en lenguaje Logo.

Nosotros podremos hacer muchas cosas con la tortuga del Logo. Disponemos de muchas órdenes directas (los primitivos) y podemos inventar cuantas órdenes queramos. La forma que tiene la tortuga en el AMSTRAD recuerda a la de una pequeña flecha. Pero lo mejor es que la conozcamos ya:

Carguemos el Dr. LOGO en el AMSTRAD y escribamos nuestra primera orden a la tortuga:

(en minúsculas y seguido de "enter").

Esta es la orden para decir a la tortuga que aparezca en pantalla, en la posición en que esté situada. Entonces aparece la pantalla de gráficos y la tortuga situada en el centro... esperando nuestras órdenes.

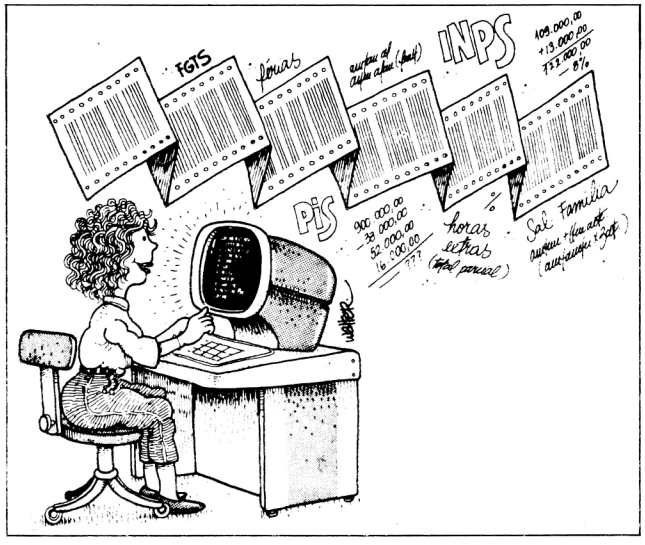
Podemos hacer invisible a la tortuga, sin que por ello abandone su actual posición en la pantalla ni ninguna de sus otras características. Simplemente dejaremos de verla pero ella estará ahí. Esta orden es:

ht

Pongamos atención sobre un hecho ya mencionado: cada orden que se refiera a una característica de la tortuga, cambia precisamente esa característica y respeta el estado de las demás.

III.2 SOBRE LOS MENSAJES DE ERROR

SI hubiéramos escrito la orden anterior incorrectamente, la tortuga no nos hubiera entendido, como es natural. Probemos, por ejemplo a darle la orden anterior pero escribiendo cada letra en mayúsculas. Inmediatamente después de que pulsemos la tecla enter aparece el siguiente mensaje en la pantalla: 'I don't know how to



ST', ('No sé cómo se hace ST'). Es un mensaje de error.

Siempre que demos una orden (una instrucción) incorrectamente, aparecerá algún mensaje de error. El ordenador no se estropeará por eso, ni la tortuga... ni nadie. Lo que sí debemos hacer cuando veamos un mensaje de error es pararnos a analizar qué es lo que ha pasado, qué es lo que hemos hecho equivocadamente para que podamos poner remedio y sobre todo, para que no nos vuelva a suceder. Esta es la forma más razonable de aprender.

Si el error se produce cuando estamos trabajando en modo directo bastará con que escribamos de nuevo la orden pero ahora correctamente. Por el contrario si un programa se para cuando todavía no había terminado y emite un mensaje de error entonces deberemos reescribir la instrucción que lo ha provocado, pero ahora dentro del programa. Ya aprenderemos cómo se hace esto.

Hay muchos mensajes de error que podemos recibir de parte del Logo. Intentan dar una indicación de la razón del error. A lo largo de estos capítulos procuraremos ir indicando los tipos de errores más frecuentes asociados con lo que estemos viendo en cada momento. Tomemos nota entonces de que si una orden no la reconoce la tortuga (el Logo en general), recibiremos el mensaje anterior ('No sé cómo se hace...').

III.3 ESTADO DE LA PANTALLA

NO sólo se puede trabajar con la tortuga en Logo. También se pueden escribir programas en los que apenas aparezca o en los que no aparezca en absoluto. Incluso se pueden hacer algunos gráficos sin usar la tortuga.

Para los programas que no usen gráficos el lenguaje Logo dispone de una pantalla dedicada por completo a texto. En ella no se puede hacer ningún gráfico. La llamaremos pantalla de texto. Para situarla se usa la orden

GRAFICOS TORTUGA

ts

Es una abreviatura de la palabra textscreen que en inglés significa pantalla de texto. Hagamos la prueba en el ordenador y veremos cómo perdemos de vista a la tortuga. Vemos que aparece el prompt **?** y a su derecha el cursor (que es un cuadrado).

Cada vez que demos una orden que afecte a la tortuga se instalará la pantalla de los gráficos, para que la tortuga nos pueda obedecer. Pero existen dos pantallas de gráficos: una mixta gráficos-texto en la que se reservan algunas líneas para texto en la parte inferior y otra de sólo gráficos. Más adelante veremos que, incluso en el terreno reservado a la tortuga, es posible escribir texto.

La orden que instala la pantalla de sólo gráficos es:

fs

Es abreviatura de fullscreen. Cualquier orden que afecte a la tortuga instala la pantalla mixta gráficos-texto. Pero además nosotros podemos instalarla cuando lo creamos más conveniente sin más que escribir la orden:

ss

que es abreviatura de splitscreen.

Cuando trabajamos en Logo, hemos de tener en cuenta siempre que las cosas son relativas; que cada momento sucede a otro y que recoge siempre las condiciones que el precedente le transmite. Así por ejemplo supongamos que ejecutamos un procedimiento que, al acabar, deja la pantalla en modo mixto. Si después ejecutamos otro en el que se escriban muchas cosas en pantalla, y queremos que se vean, es necesario pasar a la pantalla de sólo texto: de otro modo sólo se verían las líneas inferiores de la pantalla. Esto no hubiera pasado si el primero que ejecutamos hubiera sido el segundo de los programas. En realidad basta con poner la orden que sitúa la pantalla de texto al principio de cada programa que la necesite. Aunque ahora no nos demos cuenta del todo, este hecho es de la mayor importancia.

III.4 MOVIMIENTO RELATIVO Y MOVIMIENTO ABSOLUTO

Enseguida estaremos dando órdenes a la tortuga para que haga dibujos en la pantalla. Pero antes es preciso que hablemos un poco de la pantalla de los gráficos que es el mundo de la tortuga.

La tortuga se mueve por la pantalla y podemos pensar que para moverse va dando pasos. En realidad si hacemos que camine un número de pasos suficientemente grande entonces sobrepasará los límites de la pantalla y... dejaremos de verla. Pero ella seguirá obedeciendo nuestras órdenes aunque no la veamos. Puede hacer dibujos muy grandes de los que sólo veamos una pequeña parte.

¿Cuántos pasos puede dar la tortuga horizontal y verticalmente, de modo que permanezca dentro de los límites de la pantalla? Horizontalmente la pantalla mide 640 pasos y verticalmente 400. De modo que si la tortuga está situada en el centro de la pantalla puede dar 320 hacia la derecha o la izquierda y 200 hacia arriba o abajo sin salirse de la pantalla. Atención porque cuando esté situada la pantalla mixta no podremos ver a la tortuga si la situamos en la zona de texto. Dentro de un momento haremos las primeras pruebas.

Existen dos clases de órdenes que hacen moverse a la tortuga. Unas, las más usadas, indican un movimiento relativo: el resultado (la nueva posición de la tortuga) depende de la situación que tenía cuando se le dio la orden. De esta manera, la misma orden dada en momentos diferentes dejará a la tortuga situada en lugares distintos. Por ejemplo es el caso de que le ordenemos algo así como: 've para adelante 10 pasos'. Es lo que se llama 'movimiento relativo' de la tortuga.

Además existe el 'movimiento absoluto'. Podemos imaginar que tenemos asociado con cada punto de la pantalla un par de números: el número de pasos que debe dar la tortuga horizontalmente y verticalmente para situarse en él, a partir del centro de la pantalla. Son las coordenadas de los puntos. Así, las coordenadas del centro son [0 0] porque este punto es el origen. Un punto que esté situado en la misma vertical que el origen pero por ejemplo 100 pasos de tortuga hacia arriba será el [0 100]: horizontalmente no ha de avanzar la tortuga para ir a él, pero verticalmente ha de dar 100 pasos.

También podemos dar a la tortuga órdenes para que se sitúe en el punto que nosotros queramos, sabiendo las coordenadas. Estas órdenes corresponden al movimiento absoluto, porque el resultado es inde-

pendiente de la situación que tenía la tortuga cuando la recibió. Tomemos nota de que hemos representado los puntos en coordenadas por dos números encerrados entre **corchetes** y separados por un espacio en blanco. En el primer capítulo decíamos lo que es una palabra y lo que es una lista. Como vemos, cada uno de los números es una palabra y hemos representado el punto por una lista [N M] en la que el primer elemento es la coordenada horizontal y el segundo la vertical.

III.5 PRIMEROS PASOS DE LA TORTUGA

Ahora estamos en condiciones de dar órdenes a la tortuga para comenzar a hacer dibujos. Comenzaremos estudiando el movimiento relativo y más adelante veremos el absoluto. Sería deseable poder ordenar a la tortuga que vaya hacia adelante o hacia atras, o que gire a derecha o izquierda, etc. Todas estas órdenes y muchas otras se le pueden dar.

En cada momento la tortuga "mira" hacia alguna dirección. Cuando le ordenemos que avance hacia delante se moverá en esa dirección. La orden que hace que avance N pasos es:

fd <N>

en la que N es un número que **tenemos** que escribir dejando un espacio en blanco después de la palabra fd (por eso lo hemos escrito encerrado entre <y>. Entonces la tortuga avanza N pasos y deja una línea dibujada por donde va pasando. Si se nos olvida escribir el número, entonces aparecerá en pantalla el mensaje,

Not enough inputs to fd

que significa 'no suficientes datos para fd'. Es un mensaje de error que nos aparecerá siempre que se nos olviden datos. La palabra fd es abreviatura de forward que, en inglés, significa adelante.

Para que la tortuga se mueva hacia atrás la orden es:

bk <>

donde todos los detalles son como antes (incluso el posible mensaje de error). La palabra bk es abreviatura de back (atrás).

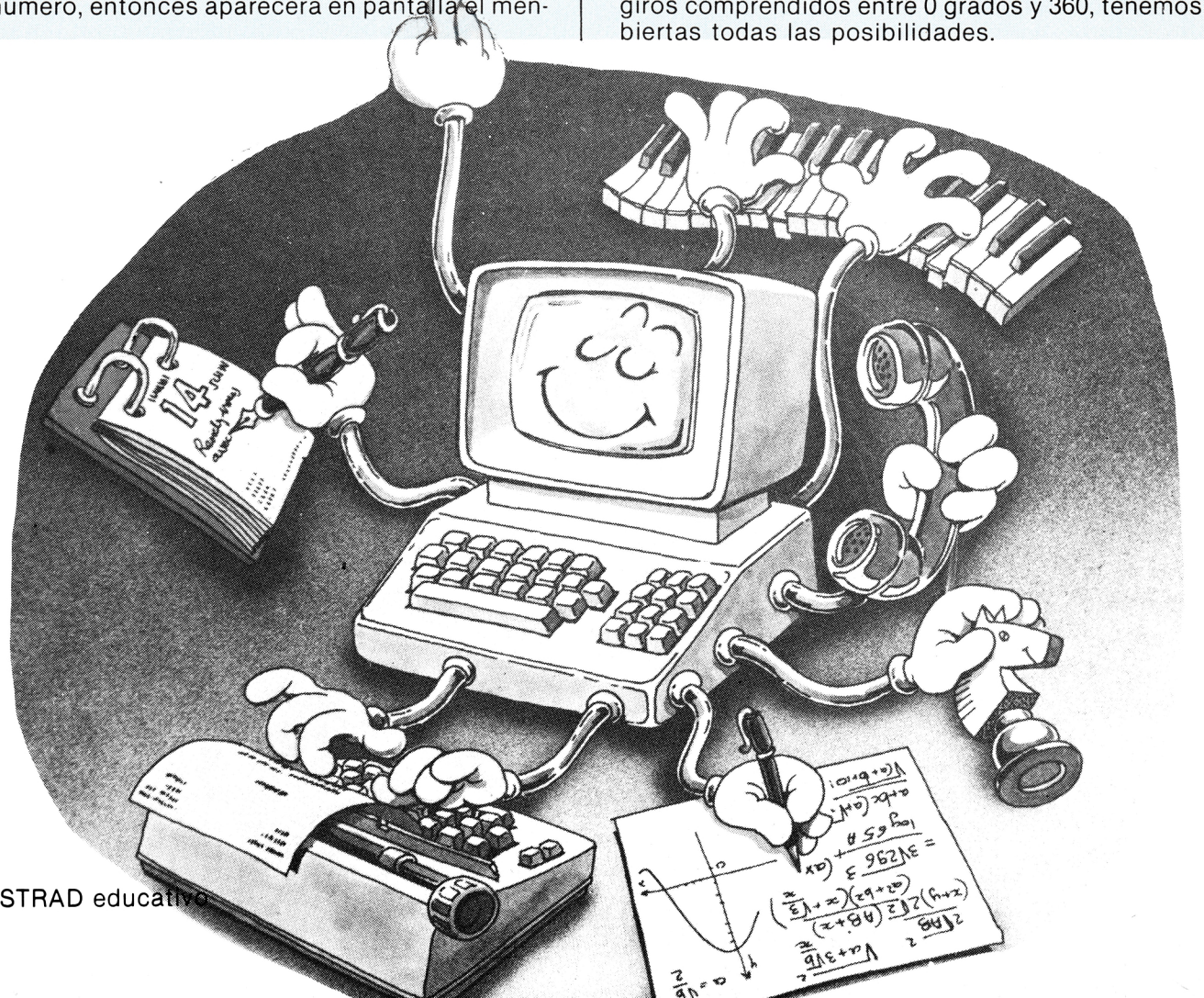
Claro que sin poder hacer que la tortuga gire, apenas podemos hacer dibujos de interés. Podemos hacer que la tortuga gire hacia la derecha o la izquierda con las siguientes órdenes:

rt <a>

para girar hacia la derecha (de la tortuga, ¡no la nuestra!), y

lt <a>

para girarla hacia su izquierda. En ambos casos la letra a representa el número de grados que queremos que gire. Es capaz de girar incluso un grado. Para ángulos pequeños la posición de la tortuga parecerá parecerá no cambiar, pero el giro lo habrá hecho. Con giros comprendidos entre 0 grados y 360, tenemos cubiertas todas las posibilidades.



Con estas sencillas instrucciones ya podemos hacer muchos y variados gráficos tortuga. Tenemos que tener siempre en cuenta un hecho importante:

Como el ángulo llano mide 180 grados, cuando queramos que la tortuga dibuje un ángulo de medida a grados, debemos girarla 180-a.

EJEMPLO III.1

Vamos a hacer que la tortuga dibuje un ángulo de 45 grados, formado por dos lados de longitudes 50 pasos. Las órdenes serán:

```
ft 50
rt 135
fd 50
```

Esto es así porque hemos calculado que 180 - 45 = 135. ¿Podría haberlo calculado la tortuga? Sí. Las órdenes que hemos estudiado permiten que en vez de un número escribamos una operación entre números.

EJEMPLO III.2

1) La siguiente secuencia de instrucciones produce el mismo resultado:

```
fd 50
rt 180-45
fd 50
```

2) Es muy sencillo dibujar un cuadrado. Basta hacer avanzar a la tortuga la cantidad que queramos que mida cada uno de los lados, y girar cada vez 90 grados. Hagámoslo:

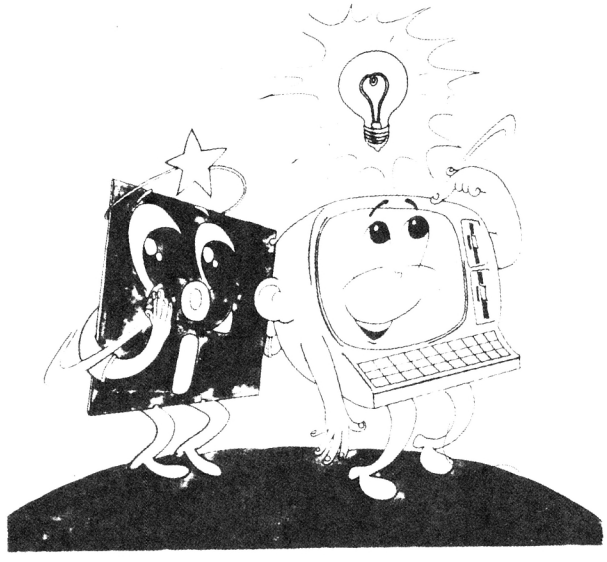
```
fd 100 lt 90
fd 100 lt 90
fd 100 lt 90
fd 100 lt 90
```

En este último ejemplo hemos escrito más de una orden en la misma línea. Ya sabemos que esto se puede hacer, siempre que incluyamos un espacio en blanco entre cada instrucción y la siguiente. Además hemos hecho el cuadrado girando hacia la izquierda. Por último vale la pena observar que el último giro que hemos hecho no es necesario para dibujar el cuadrado. Pero haciéndolo hemos conseguido que la tortuga acabe de dibujar el cuadrado en la misma posición que tenía cuando lo empezó. Ya veremos que esto es muy importante para algunas cosas.

EJERCICIO III.1

1) Escribir las órdenes necesarias para que la tortuga dibuje un cuadrado de lado 200 girando hacia la derecha.

2) Dibujar un rectángulo cuyos lados midan 100 y 200 pasos de tortuga.



3) Dibujar los dos lados de un ángulo de 30 grados (cada lado mide lo que se quiera). (Atención: ¡no ha de girar 30 grados!).

4) Dibujar los dos lados de un ángulo de 60 grados.

III.6 BORRADO DE LOS GRAFICOS

A estas alturas, si se han seguido en el ordenador los ejemplos y los ejercicios, la pantalla estará un poco 'sucia'. Es necesario aprender cuanto antes a borrar los gráficos.

Una primera forma es saliendo de la pantalla de gráficos. Así si escribimos ts pasaremos a la pantalla de texto. Si después regresamos a los gráficos con ss o con fs, la tortuga seguirá donde estaba, pero se havemos a la tortuga:

Podemos además borrar los gráficos sin salir de la pantalla del siguiente modo: la instrucción

cs

borra los gráficos que haya, sitúa a la tortuga en el centro de la pantalla (de coordenadas [0 0]) y mirando hacia arriba, e instala la pantalla mixta si se ejecuta desde la pantalla de texto o desde la mixta. Es pues una instrucción muy conveniente cuando queremos recomenzar a trabajar en un gráfico.

Otras veces sin embargo estaremos interesados en borrar los gráficos sin afectar a la situación de la tortuga. Esto se hace con la orden,

clean

Cuando se ejecuta, se borran los gráficos y se instala la pantalla mixta si se estaba en la de texto (todas las órdenes que afectan a la tortuga hacen esto último).

EJERCICIO III.2

1) Dibujar el perfil de los peldaños de una escalera.
2) Escribir las órdenes necesarias para borrar la pantalla y dibujar un triángulo equilátero de lado 100, desde el centro de la pantalla y girando la tortuga hacia la derecha. Repítase luego, girando hacia la izquierda y con el lado igual a 230. (Nota: recuérdese que los tres ángulos de un triángulo equilátero miden 60 grados).
3) Dibujar dos triángulos equiláteros que tengan un lado común.

III.6 DIBUJO DE POLIGONOS REGULARES

Un polígono regular está formado por el mismo número de lados que de ángulos. De modo que si la tortuga recorre uno cualquiera de ellos, cuando llegue al punto de partida habrá girado precisamente 360 grados (una vuelta completa). Como todos los ángulos son iguales resulta que si el polígono tiene N lados y N ángulos cada vez la tortuga habrá tenido que girar 360/N.

Esto nos permite dibujar todos los polígonos regulares.

EJEMPLO III.3

1) Pentágono regular de lado 120, de modo que no vemos a la tortuga):

```
cs
ht
fd 120 rt 360/5
fd 120 rt 360/5
fd 120 rt 360/5
fd 120 rt 360/5
fd 120 rt 360/5
```

2) Hexágono regular de lado 90:

```
clean
st
fd 90 lt 360/6
fd 90 lt 360/6
fd 90 lt 360/6
fd 90 lt 360/6
fd 90 lt 360/6
fd 90 lt 360/6
```

3) Dos cuadrados iguales que tienen un lado común:

```
cs
ht
bk 50 rt 90
bk 50 rt 90
bk 50 rt 90
bk 50 rt 90
fd 50 rt 90
fd 50 rt 90
fd 50 rt 90
fd 50 rt 90
```

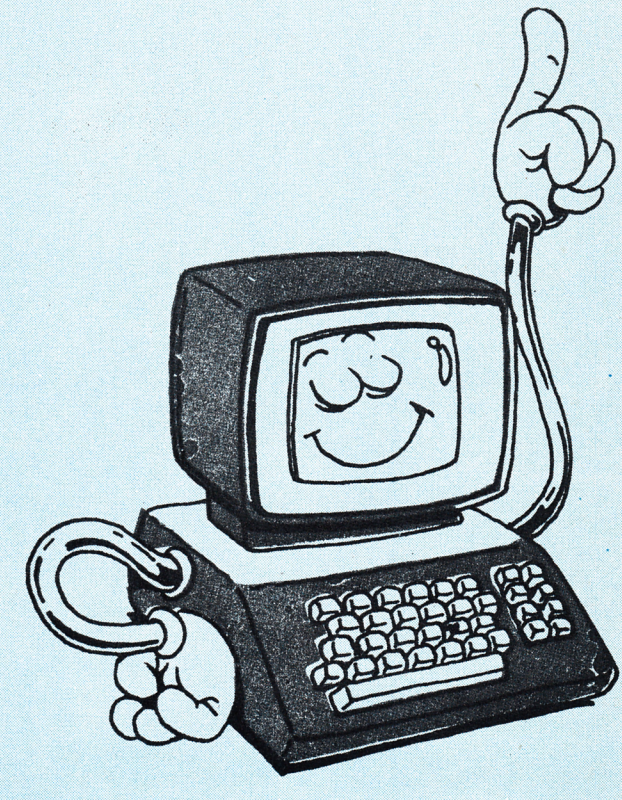
EJERCICIO III.3

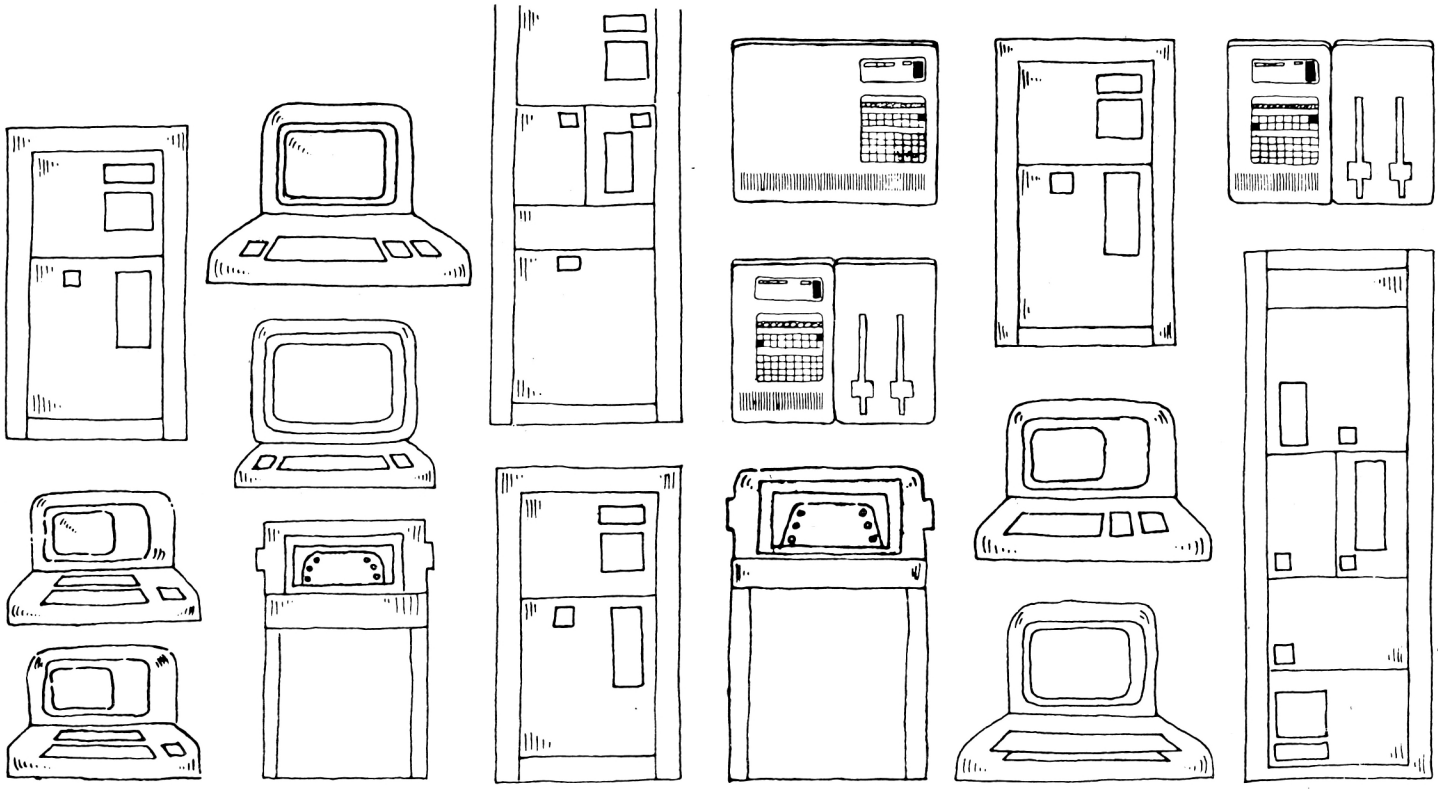
1) Dibujar un octógono (8 lados) regular de lado 70 y, sin borrarlo, otro desde donde esté la tortuga al final del primero pero de lado 50.
2) Dibujar 3 eneágonos (9 lados) encajados (como los octógonos anteriores).

III.8 EDICION DE UNA LINEA DIRECTA

Por mucho cuidado que hayamos puesto al hacer en la máquina los ejemplos y ejercicios, es inevitable que nos hayamos equivocado más de una vez mientras estamos escribiendo las órdenes en el teclado. Si pulsamos la tecla "enter" y la orden contiene un error, no se puede ya corregir. Recibiremos, en tal caso, el correspondiente mensaje de error y a continuación podemos proceder a escribirla de nuevo.

Si advertimos el error antes de haber pulsado enter, entonces podemos rectificarlo sin tener que reescribir toda la línea. Las acciones que permiten corregir una línea se llaman acciones de edición de esta. Con las teclas de movimiento del cursor a la izquierda y derecha (→ y ←) podemos situar el cursor en el lugar necesario. Si escribimos encima de otros caracteres, estos no serán sustituidos; por el contrario lo que escribamos se insertará entre los caracteres que queden a la izquierda del cursor y el caracter sobre el que está situado este. Para borrar el caracter situado





a la izquierda del cursor se utiliza la tecla rotulada 'DEL'. Para borrar el caracter situado bajo el cursor se utiliza la tecla rotulada 'CLR'.

Como máximo, una línea de instrucciones (línea lógica) puede ocupar 2 renglones (2 líneas físicas) antes de pulsar enter. Si tenemos una línea de instrucciones que ocupe dos renglones, podemos pasar de uno a otro con las teclas de movimiento vertical del cursor.

APENDICE: SOBRE LA INSTRUCCION "REPEAT"

Aunque desde el punto de vista puramente estructural tal vez no sea este el lugar más adecuado, vamos a hablar en este apéndice al capítulo III de una instrucción que nos puede ahorrar muchos minutos de escritura en el teclado. Esto independientemente de que más adelante volvamos sobre ella.

En programación es muy importante la creación de rutinas (algoritmos) que repetidos un número de veces por el ordenador, produzcan el resultado deseado.

Así en nuestro EJEMPLO III.3.1 y III.3.2., hemos escrito repetidamente la misma secuencia de dos instrucciones. El EJERCICIO III.3.1 y III.3.2 ha puesto al lector en la necesidad de escribir muchas veces la misma cosa.

Existe en Logo una instrucción que viene a resolver el problema, cuando el número de veces que ha de repetirse un grupo de instrucciones depende de un valor numérico. La forma de la instrucción es:

repeat <N> <[lista de instrucciones]>.

La palabra 'repeat' ha de escribirse en minúsculas. El número N (es obligatorio poner un número o una operación entre números) indica cuántas veces ha de repetirse la lista de instrucciones. La lista de instrucciones es la secuencia de instrucciones que queremos que se ejecute N veces. Cada una de las instrucciones ha de separarse de la que le precede y de la que la sigue por uno o más espacios en blanco. No es necesario poner ninguna instrucción dentro de los corchetes, pero estos son indispensables. Se pueden incluir tantas órdenes como quepan en las dos líneas.

Usando esta instrucción algunos de los ejemplos ya vistos son mucho más cómodos y manejables.

EJEMPLO III.4

```
    1) (Como el ejemplo III.3.1)
cs ht
repeat 5 [fd 120 rt 360/5]
    2) (Como el ejemplo III.3.2)
clean st
repeat 6 [fd 90 lt 360/6]
    3) (Como el ejemplo III.3.3)
cs ht
repeat 4[bk 50 rt 90]
repeat 4[fd 50 lt 90]
```

(Nota: no ha de dejarse obligatoriamente espacio entre el número N y [o entre [y el caracter siguiente porque los caracteres '[' y ']' son separadores de palabras. Pero pueden dejarse).

EL NUMERO OCULTO

El presente programa tiene como objeto el jugar al número oculto con la máquina. El ordenador piensa un número de 0 a 99 y se dispone de 6 intentos para averiguarlo. Para ejecutarlo basta escribir "numoc" después de escribirlo en el teclado.

LISTADO

```
to numocu
ct                                          BORRA PANTALLA.
make "oculto random 100                     GENERA EL NUMERO
pr [EL JUEGO DEL NUMERO OCULTO.]            TITULOS
pr [- - - - - - - - - - - - - - - -]
pr [] pr[] pr[]
pr [TIENES QUE AVERIGUAR EL NUMERO OCULTO.]
pr [TIENES SOLO 6 OPORTUNIDADES.]
pr [CUANDO ESTES PREPARADO PULSA ENTER.]
make "listo rq                              ESPERA A QUE SE PULSE ENTER
ct
make "numin 0                               NUM. INTENTOS=0.
repeat 6[intento if :numero=:oculto [stop]] REPITE 6 VECES EL PROCEDI-
                                            MIENTO intento Y CADA VEZ
                                            ANALIZA SI SE AVERIGUO EL
                                            NUMERO.
end

to intento
type [CUAL ES EL NUMERO?]                   NO HACE RETORNO DE CARRO.
```

```
make "numero rq                                    PIDE EL NUMERO AL TECLADO
                                                   Y LO ALMACENA EN LA VARIA-
                                                   BLE LLAMADA numero.
if :numero=:oculto [pr[LO HAS CONSEGUIDO           SI SE HA ACERTADO EMITE
     EN  ] type :numin+1 type [ ] pr" INTENTOS]]   MENSAJE.
if :numero>:oculto [pr[EL OCULTO ES MENOR]         SI EL OCULTO ES MENOR INFOR-
     make "numin :numin+1]                         MA Y AUMENTA EL CONTADOR.
if :numero<:oculto [pr[EL OCULTO ES MAYOR]         COMO LA ANTERIOR.
     make "numin :numin+1]
end
```

2) ** MUCHOS CUADRADOS ** (D)

Este programa realiza una figura que consiste en 10 cuadrados, cada uno de los cuales se obtiene girando el anterior 36 grados, para conseguir una figura regular muy interesante

```
to figura
cs fs                                              BORRA LA PANTALLA E INSTALA
                                                   LA PANTALLA DE SOLO GRAFICOS
repeat 10 [cuadrado rt 36]                         REPITE 10 VECES EL PROCEDI-
                                                   MIENTO CUADRADO Y GIRAR 36
                                                   GRADOS A LA DERECHA.
end

to cuadrado
repeat 4[fd 100 rt 90]                             POR 4 VECES AVANZA Y GIRA.
end
```

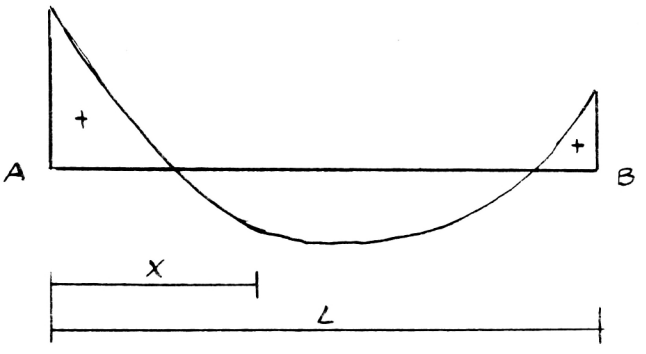
LEYES DE MOMENTOS EN UNA VIGA

Víctor J. Campo López

UNA vez se ha resuelto por Cross u otro medio un pórtico o viga continua, queda la labor de calcular, viga por viga, las leyes de momentos obtenidos por el cálculo anterior, mas los debidos a las cargas sobre la viga, generalmente uniformes (incluido el peso propio).

Ya que la ley debida a una carga uniforme en una viga biapoyada es una parábola de segundo grado, construir la ley resultante, supone "colgar" dicha parábola desde los momentos debidos al Cross.

El momento resultante en un punto cualesquiera de la viga, será la suma algebraica de los obtenidos por tres efectos, el momento en A, el de B y el debido a la carga uniforme. Es decir,



$$Mx = MA \frac{L - X}{L} + MB \frac{X}{L} + Q \frac{X\,(L-X)}{L}$$

Siendo: MA = Momento en A.
MB = Momento en B.
Q = Carga uniforme.
L = Longitud de la viga.
X = Punto considerado.

Existen tres puntos singulares A y B, que son conocidos (extremos) y el punto en el que el momento negativo es máximo.

El programa calcula el momento máximo central (no en el centro necesariamente), y los momentos a lo largo de la barra, en varios puntos, lo que permite ver la evolución de los mismos y los cambios de signo (puntos de corte con la barra, de momento = 0).

Además el programa dibuja gráficamente la ley de momentos resultantes.

Por último y para el caso de hormigón armado, calcula las capacidades mecánicas de la armadura, para los momentos máximos. En el caso en que el canto elegido sea inferior al mínimo, el programa avisa de este hecho.

El dibujo utiliza en longitud, dos escalas distintas a fin de que no se salga de la pantalla. Las escalas verticales y horizontales son distintas, para dar mayor claridad, pueden ser modificadas en la línea 330.

Las unidades utilizadas vienen indicadas en los propios INPUTS. Todas las longitudes se expresan en metros (m.), los momentos en metros por tonelada (m.T) y las cargas en Toneladas/metro (T/m), consecuentemente la capacidad mecánica se obtiene en Toneladas (T).

El criterio de signos puede modificarse, suprimiendo la línea 140.

Introduciendo los datos en el programa, en la forma:

— Longitud de la barra (m) = 6,0
— Carga uniforme (T/m) = 2,3
— Momento en A (m.T) = 0,875
— Momento en B (m.T) = 3,16
— Canto de la viga (m) = 0,4
— Ancho de la viga (m) = 0,3

Se obtienen en pantalla los siguientes resultados:

• Momento central máximo = 8,36 en X = 2,80 (el momento central es el máximo negativo según la figura, no tiene por qué coincidir con X = L/2, salvo en el caso que MA = MB).

• Tabla de valores de X, con los momentos correspondientes.

• Gráfico de la ley de momentos.

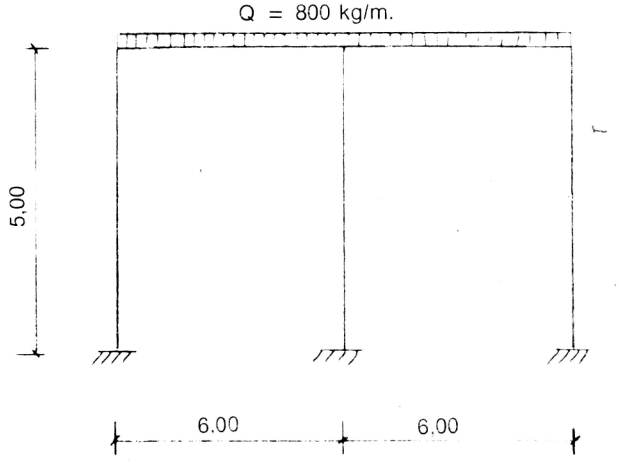
• Capacidades mecánicas a absorber por la armadura (hormigón armado).

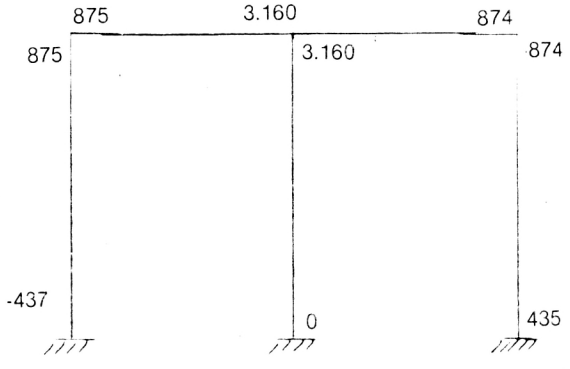
UA = 2,15 en el extremo A (empotramiento).
UB = 7,97 en el extremo B (empotramiento).
UC = 22,4 en el vano (corresponde al Mmax).

(El símbolo Pt equivale a #)

EJEMPLO

En el pórtico de la figura, se han obtenido por el método de Cross los siguientes momentos de empotramiento.





```
10 ' ..................................
15 '
20 '        LEY DE MOMENTOS
30 ' ..................................
35 '
41 MODE 1:  PRINT "   ESTE PGM   CALCULA
 LA LEY DE MOMENTOS EN UNA VIGA A PARTIR
 DE LOS MOMENTOS DE EMPOTRAMIENTO Y LA S
OBRECARGA UNIFORME (No se incluyen carga
s puntuales)."
42 PRINT : PRINT "  IMPRIME LA TABLA DE
MOMENTOS, GRAFICO Y CAPACIDADES MECANICA
S DE LAS ARMADURAS PARA LOS MTOS. MAXIMO
S (izquierdo, derecho y central)."
50 LOCATE 1,25: PRINT "PRESIONAR UNA TEC
LA PARA CONTINUAR"
60 Z$=INKEY$:IF Z$="" THEN 60
70 MODE 2
80  INPUT "Longitud de la barra (m)=",l:
 LOCATE 40,1
90  INPUT "Carga uniforme     (T/m)=",q
100  INPUT "Momento en A        (m.T)=",m
a: LOCATE 40,2
110  INPUT "Momento en B        (m.T)=",m
b
120  INPUT "Canto de la viga     (m) =",h
: LOCATE 40,3
130  INPUT "Ancho de la viga     (m) =",b
140 MA=-MA:MB=-MB
150 DEF FN A(X)=MA/L*(L-X)+MB*X/L+Q*X/2*
(L-X)
160 MMAX=-1000:XMAX=0
170 FOR x=0 TO 1 STEP 0.1
180 F=FN A(X)
190 IF F>MMAX THEN MMAX=F:XMAX=X
200 NEXT X
210 PRINT CHR$(24);"    Xmax="; USING "##
#.##    ";XMAX: LOCATE 40,4
220 PRINT "    Mmax="; USING "###.##    ";
FN A(XMAX): PRINT CHR$(24)
230 PRINT "  X       Mto.     X      Mto."
240 PRINT "-------------------------------
-"
245 IF L<8 THEN S=0.5 ELSE S=0.75
```

```
250 FOR x=0 TO L STEP S
260 MX=FN A(X): PRINT  USING "##.##  ";X
;MX;
270 IF (X+S/2)>L THEN 300
280 MX=FN A(X+S/2): PRINT  USING "##.##
 ";X+S/2;MX
290 NEXT X
300 IF X<>L THEN PRINT  USING "##.##  ";
1;FN A(L)
310 GOSUB 450
320 ' DIBUJO
330 IF L<8 THEN EH=40:EV=10 ELSE EH=30:E
V=10
340 PLOT 250,200
350 DRAWR L*EH,0
360 PLOT 250,200:DRAWR 0,MA*-EV
370 PLOT 250+L*EH,200:DRAWR 0,MB*-EV
380  FOR X=0 TO L STEP 0.05
390 PLOT 250+X*EH,200-FN A(X)*EV
400 NEXT X
410 LOCATE 40,6: PRINT CHR$(24);" Ua=";
USING "##.##";UA
420 LOCATE 50,6: PRINT "  Uc="; USING "#
##.##";Uc
430 LOCATE 60,6: PRINT "  Ub="; USING "#
##.##":Ub: PRINT CHR$(24)
440 Z$=INKEY$:IF Z$="" THEN 440 ELSE RUN
 50
450 ' canto minimo y capacidades mecanic
as
460 m=ABS(ma):GOSUB 500:GOSUB 530:UA=U
470 m=ABS(mb):GOSUB 500:GOSUB 530:UB=U

480 m=ABS(mmax):GOSUB 500:GOSUB 530:UC=U

490 RETURN
500 hmin=1.69*SQR(m/b/1666)
510 IF h>hmin THEN RETURN
520 PRINT CHR$(24);"CANTO INFERIOR AL MI
NIMO";CHR$(24):RETURN
530 ' CAPACIDAD MECANICA
540 U=0.97*M/H*(1+M/B/H/H/1666)
550 RETURN
```

NUESTRO PRÓXIMO NÚMERO

AMSTRAD EDUCATIVO

N.º 4

EL AMSTRAD y el CPM

Ficheros en el AMSTRAD

Basic del AMSTRAD

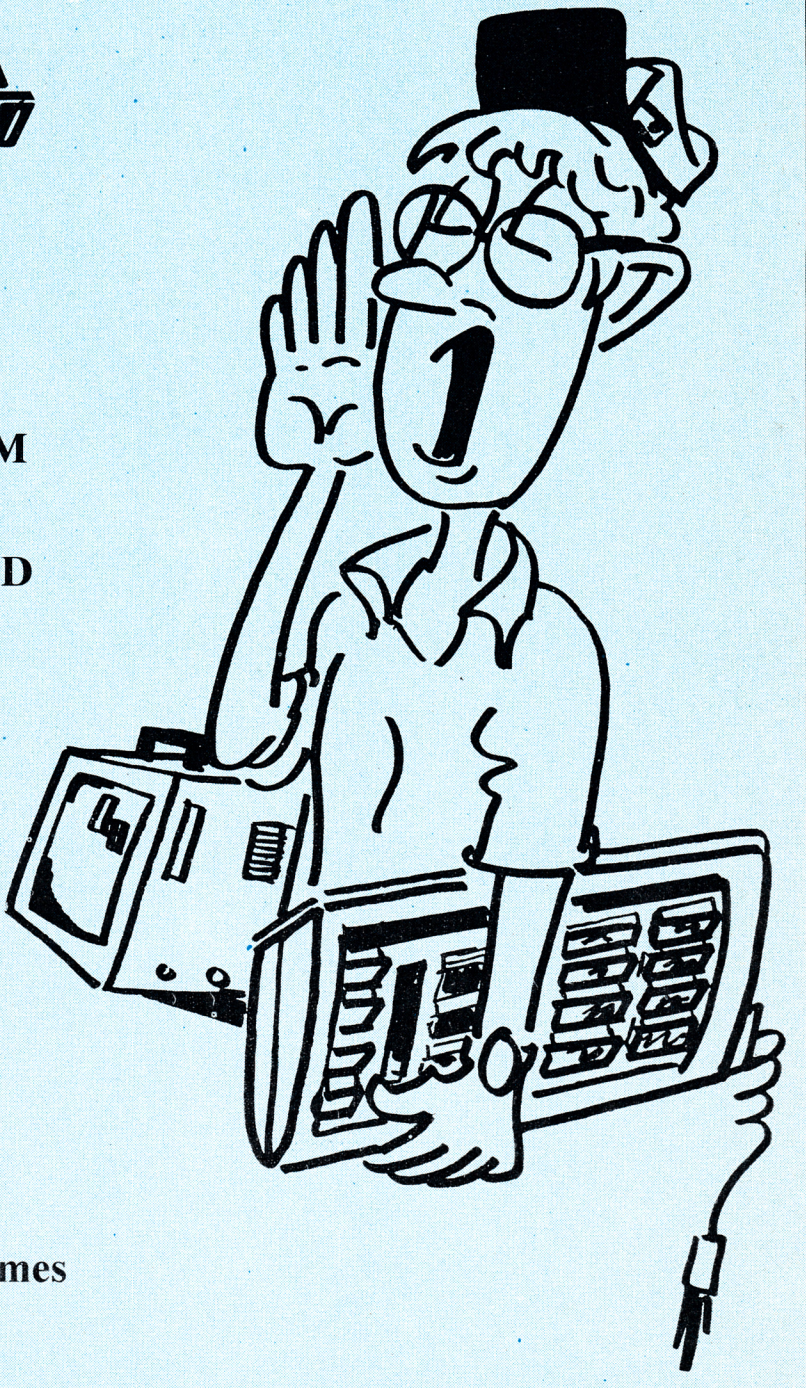
Fichas del AMSTRAD

Programando en Basic

Logo del AMSTRAD

Doctor Logo

El programa técnico del mes



LA "BIBLIA" DEL

¡ya está a la venta!

AMSTRAD

FASCICULO 4

295 PTAS.

DOMINARLO ES UN JUEGO

¡ya está a la venta!