AMSTRAD EDUCATIVO

- EL AMSTRAD Y SU CP/M.
- PROGRAMANDO EN BASIC Y LOGO. EJEMPLOS PRACTICOS.
- FICHEROS EN EL AMSTRAD

295 Ptas.

Nº1

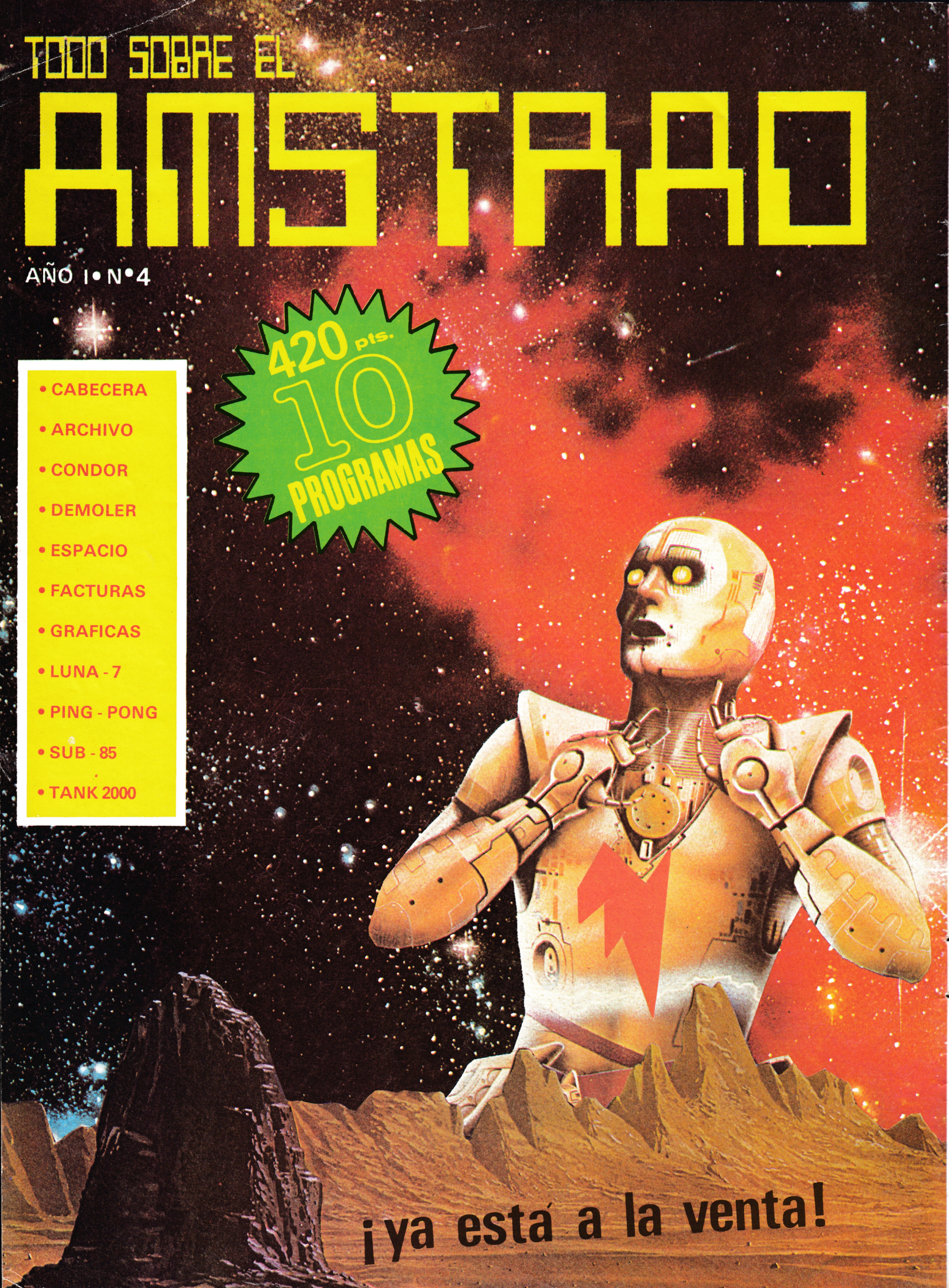
TODO SOBRE EL

AMSTRAD

AÑO I • N°4

420 pts.
10
PROGRAMAS

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



¡ya está a la venta!

EDITORIAL

En los últimos meses ha alcanzado el AMSTRAD cotas de estimable difusión, hecho que constituye por sí mismo motivo más que sificiente para que le dedicásemos toda nuestra atención. Por ello, salimos con AMSTRAD EDUCATIVO.

En el quiosco, veremos, una revista más para el usuario, una nueva manera y un enfoque de como utilizar el AMSTRAD

Con una simple mirada al sumario, en esta misma página, podéis haceros una idea de la trayectoria que pretendemos seguir. No obstante, como la revista está pensada y hecha para vosotros, sería muy importante que participáseis en ella dando vuestra opinión, haciendo todo tipo de sugerencias, indicando, en una palabra, como queréis que sea vuestra revista.

Esperamos vuestras opiniones. Que os guste.

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
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

Función del DOS en un microprocesador. Puesta en marcha

DANDO por supuesto que el lector ha utilizado un ordenador, tal vez se haya preguntado cómo es posible que al pulsar una tecla, aparezca un carácter en la pantalla.

Bien, esto es así debido a que en algún lugar de la memoria, hay un programa cuya función es examinar constantemente el teclado, informando a la CPU de todas las alteraciones que se produzcan, y, si las hay, determinar qué tecla fue apretada, fijar el modelo de puntos a que corresponde el carácter equivalente, y activar el tubo de rayos catódicos para su impresión en pantalla.

Y, admitiendo que el comando SAVE del BASIC encargado de transferir la información de la RAM a un soporte magnético se haya empleado en alguna ocasión, cabría plantearse: ¿Cómo se produce la transferencia? y ¿qué lo controla y adapta? Bajo las preguntas anteriores, y los procesos implicados en las respuestas, está subyacente el sistema operativo.

El sistema operativo es un conjunto de programas que supervisa constantemente todo el equipo, y sin el cual la máquina es inservible e incapaz de efectuar maniobra alguna.

Algunos sistemas operativos están grabados en ROM, especialmente en los microordenadores de menor porte, encargándose de los controles necesarios. Esto quiere decir que desde el momento de la conexión, el ordenador está bajo el control de su sistema operativo. En el otro extremo están aquellos computadores que deben transferir de un disco a la memoria interna su sistema operativo (DOS). Es claro pues, que en este último caso y en el momento de la conexión, el computador está imposibilitado para actuar, incluso en el sentido de manejar el disco que contiene el DOS que necesita.

Para superar esta inconsecuencia, un pequeño programa grabado en ROM es el encargado de localizar el sistema operativo en el disco situado en una unidad de disco predeterminada, e iniciar la carga del DOS, cediendo el control a este.

Este proceso es conocido como **booting** o boot. Nosotros nos referimos a él mediante la expresión "inicializar el sistema" o, también, como una entrada o arranque en frío **(cold start).**

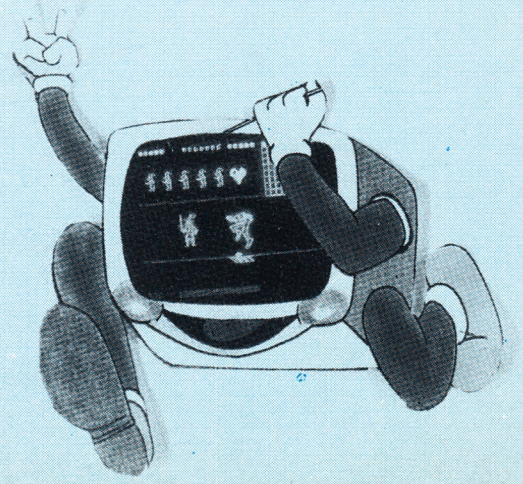
Un DOS ofrece más opciones al usuario de un ordenador, que los sistemas operativos en ROM, teniendo a su cargo, además de la supervisión general, una serie de funciones para manipular los ficheros almacenados en un disco y la unidad donde éstos están instalados.

Estas funciones se cumplen gracias a ciertos comandos u órdenes que permiten al operador instruir al DOS en el sentido adecuado.

El estudio de estos comandos se verá en la segunda parte de este curso.

Como ejemplo de las misiones encomendadas a estos comandos se pueden anticipar la orden de mostrar en pantalla un directorio completo de los ficheros que contiene un disco y la detención de un proceso ya iniciado.

Un sistema operativo elemental apenas estará capacitado, por ejemplo, para cargar en RAM un fichero determinado, buscándolo entre todos los almacenados en el soporte en que esté contenido. Por el contrario, un DOS del tipo de CP/M puede averiguar exactamente donde está situado cualquier fichero en el soporte.



POR A. BELLIDO

Por todo lo expuesto, y sea cual sea su grado de sofisticación, un sistema operativo puede considerarse como un conjunto de programas que, controlando la información dentro del ordenador, permiten la comunicación entre éste y el operador.

CP/M se hace cargo, además, de la actuación de los diversos periféricos.

Compatibilidad

Con el término inglés **portability** se da a entender la capacidad de un programa para ser ejecutado o explotado por más de un tipo de ordenador.

Esta idea la expresamos en español con la palabra compatibilidad.

Dicho esto, pasemos a considerar los motivos por los cuales un DOS es, o no, compatible en dos ordenadores diferentes, y la razón, por la cual distintos DOS no pueden actuar sobre el mismo ordenador.

Con anterioridad, en varias ocasiones, se ha hecho referencia al DOS como a un conjunto de programas.

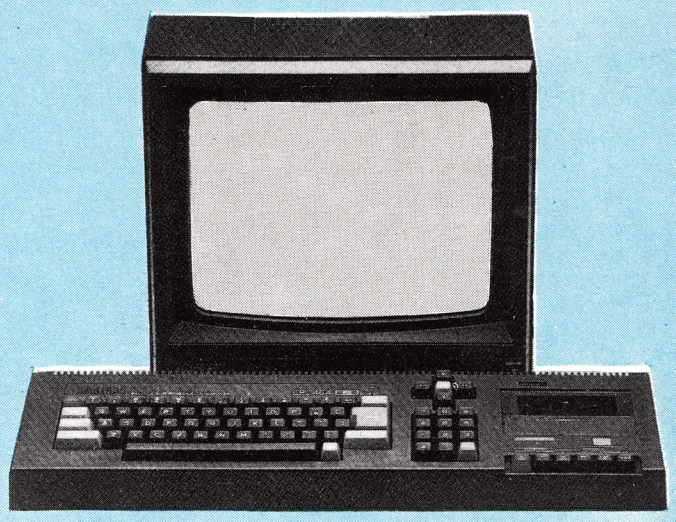
Estos programas deben estar escritos en un determinado lenguaje de programación.

El lenguaje de programación, como se comenta en otro lugar, está en función del microprocesador encargado de ejecutar sus instrucciones.

Por tanto, un DOS desarrollado para correr sobre procesadores del tipo 8080 A, Z 80, y 8085, como CP/M-80, no está en condiciones de servir en un equipo que porte un 8086 o un 8088, para los cuales sería necesario emplear CP/M-86 o MS-DOS. Y, aún entre estos últimos, su distinta concepción los hace diferentes. Antes de proseguir, tomemos contacto con las tres tareas fundamentales que tienen encomendadas los sistemas operativos en disco de los tipos CP/M y MS-DOS:

* **Recibir y ejecutar los comandos u órdenes,** a los que ya se ha dedicado algún párrafo y sobre los que volveremos detalladamente.

* **Controlar la organización de los datos** contenidos en un disco. Esta actividad es invariable para un mismo DOS.



* **Control de entradas y salidas.** Es una parte del DOS que contiene todas las rutinas necesarias para comunicar el microprocesador con los periféricos. Cada fabricante de ordenadores lo configura de acuerdo con su propio criterio, y por tanto puede variar en un mismo tipo de DOS, según haya sido, o no, modificada esta sección del sistema operativo.

La causa de incompatibilidad entre sistemas operativos en disco del mismo tipo depende por tanto, de las alteraciones introducidas en las rutinas de control de entrada/salida por los fabricantes para adaptarlas a sus respectivas configuraciones de ordenador.

Por consiguiente las causas de incompatibilidad entre un ordenador y un programa desarrollado en base a un determinado DOS pueden tener los siguientes orígenes:

1.° **El DOS bajo el cual fue desarrollado el programa es distinto del que controla las operaciones de la máquina.**
Esta causa es evidente ya que el conjunto de rutinas que conforman un DOS de una determinada marca, son distintas de las de otro cualquiera. Tal sería el caso de un programa escrito para CP/M-86, el cual sería incompatible en un ordenador con MS-DOS.

2.° **El DOS bajo el cual fue desarrollado el programa, es del mismo tipo del que controla las operaciones de la máquina,** pero el programa no es compatible por que el control de entradas y salidas no es el mismo en ambos ordenadores.

3.° **El DOS bajo el cual fue desarrollado el programa "corre" es un microprocesador distinto del que porta el ordenador.** Este es el caso de aquellos programas escritos, por ejemplo, para un CP/M-80, que trabaja correctamente en Z-80, 8080 y 8085, siendo incompatible, con otro tipo de microprocesador, como sería el caso del 8086 y 8088, que requieren un DOS del tipo CP/M-86 o MS-DOS.

Origen de CP/M. Su desarrollo. Versiones

El origen de CP/M está situado en los intrincados comienzos de los modernos ordenadores personales, los cuales, por cierto, se gestaron sin que nadie tuviera intención de darlos a la luz.

La historia, resumida, es esta.

Hacia 1971, las compañía californianas dedicadas a manufacturar circuitos integrados estaban persiguiendo la idea de llegar a producir microprocesadores económicos cuya misión fuera controlar los mecanismos de una fábrica o automatizar ciertos procesos, como por ejemplo, los arranques y paradas de un ascensor.

En otras palabras, buscaban un capataz electrónico para dirigir el trabajo de otras máquinas.

Este esfuerzo tuvo como consecuencia la aparición del primer microprocesador, denominado 4004 por su creador —INTEL—. De 4 *bits*, con muy pequeña capacidad operativa.

En 1972, INTEL desarrolló el 8008 de 8 *bits*, gracias al cual se abrieron las puertas a los entendidos para montar sus propios computadores, sin posibilidad de conectarlos con teclados o pantallas.

INTEL continuó su saga con el microprocesador 8080 también de 8 *bits*, pero manejando hasta 64 *Kbytes* de memoria interna.

Por estas fechas, Gary Kildall era consultor de INTEL trabajando para **Microprocessor Application Associates** en un proyecto de lenguaje de programación —nombrado PL/M— para el más reciente microprocesador de INTEL. En esta situación, Kildall pensó que era posible unir un microprocesador, de las características del que tenía entre manos, con una unidad de disco de Shugart y a un teletipo. El fin que perseguía era dotar a cada ingeniero de INTEL de su propio ordenador individual, en lugar de compartir entre ellos un ordenador de mayores prestaciones. La propueta fue rechazada.

Como respuesta, Kildall y un amigo —Torode—, construyeron su primer sistema. Aquel se encargó del sistema lógico, este del físico.

De aquí salió un programa monitor, o guía, de las operaciones encargadas a la CPU para controlar los periféricos que tenía a su cargo. Su nombre: CP/M, acrónimo de **Control Program/Monitor.**

Digital Research se creó para su comercialización.

Este fue el primer sistema operativo para el nuevo concepto de "microordenador", que, fue aceptado por aquellos fabricantes que querían completar sus equipos con unidades de disco, y cuyo computador estaba basado en un microprocesador 8080.

Posteriormente, y con mayor rapidez de proceso, CP/M correría igualmente sobre el 8085 de INTEL y el Z80 de Zilog.

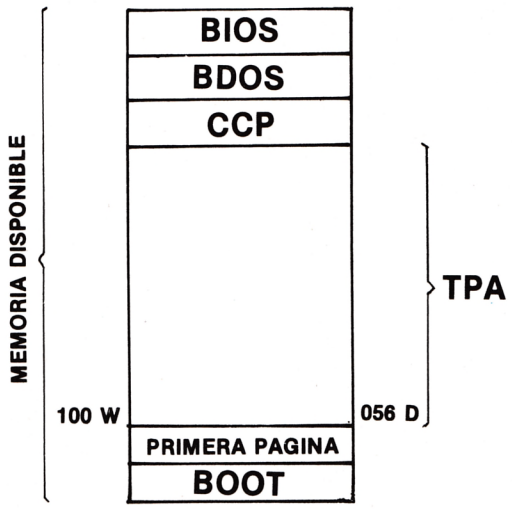
En 1981, Digital Research puso en el mercado CP/M 86, un DOS para ordenadores basados en microprocesadores 8086 y 8088.

Las versiones de CP/M-80 lanzadas al mercado hasta el momento son 1.3, 1.4, 2.0, 2.1 y 2.2.

La norma general para interpretar los dígitos que dan nombre a cada versión, es considerar que el número situado a la izquierda del punto se refiere a la propia versión y el primer dígito a la derecha del punto indica las diferentes revisiones de una versión.

Un segundo dígito a la derecha del punto, indica una adaptación específica a un ordenador de la versión y revisión manifestada según lo explicado anteriormente.

En la actualidad, el número de usuarios de CP/M debe pasar del millón, y la mayoría de los ordenadores utilizan este DOS, bien como elemento base del equipo o bien como opción.



Disco del sistema (System Disk)

Con anterioridad, nos hemos referido a un DOS como a un conjunto de programas, y CP/M no es una excepción.

El disco que contiene los programas de un DOS, recibe la denominación genérica de "disco del sistema" (*System disk*).

En un disco del sistema correspondiente a CP/M las dos pistas más externas, al menos, están reservadas para los programas que lo integran, los cuales se agrupan bajo las siguientes denominaciones, y con las funciones que se indican:

* **BIOS** *(Basic input/output system)* o "sistema básico de entradas y salidas" entre la CPU y el resto de los periféricos. Las misiones generales que tiene asignadas son:

1.° Localizar el periférico emisor o receptor de información, y comprobar que está disponible.

2.° Efectuar la transmisión de la información con eficacia.

3.° Detectar errores en la transmisión y fallos en el sistema físico.

* **BDOS** *(Basic disk operating system)* o "Sistema básico operativo del disco", destinado a controlar los ficheros —y sus datos— contenicos en un disco. En este sentido, al añadir un nuevo fichero a un determinado disco, el BDOS se encarga de:

1.° Detectar que en la pista catálogo no existe otro fichero con el mismo nombre.

2.° Detectar que hay espacio suficiente en el disco, para contener el nuevo fichero.

3.° Dar entrada en la pista catálogo al nuevo fichero y sus parámetros.

* **CCP** *(Console command processor)* o "procesador de órdenes de consola", gracias al cual, el usuario puede comunicarse con el DOS a través del teclado, escribiendo en él las diferentes órdenes que pueden ser dadas, y que se estudian posteriormente. Cuando CP/M está en condiciones de aceptar estas órdenes, decimos que el sistema está a "nivel de operador".

* **BOOT** es un pequeño programa destinado a cargar la totalidad de CP/M en la memoria interna, transfiriendo el control al BIOS, el cual lo transfiere, finalmente, al CCP.

Primeros pasos con CP/M

Una vez instalado el ordenador de acuerdo con las condiciones dadas por el fabricante, el usuario interesado en el DOS debe recurrir al manual, o sección del manual, dedicado al CP/M. Lo usual será encontrarse con una GUIA DEL USUARIO O UNA INTRODUCCION A LAS CARACTERISTICAS Y FACILIDADES DEL CP/M.

Estos manuales, además de una somera descripción del CP/M y su función, comienzan dando unas normas para la correcta lectura e interpretación del contenido de los mismos, y el proceso que debe seguirse para transferir CP/M del disco a la RAM.

La carga del CP/M puede diferir bastante de un equipo a otro, por lo cual se hace imprescindible seguir las instrucciones dadas, a este respecto, en el manual. No obstante, es usual uno de estos procedimientos:

1. Conectar todos los dispositivos y, a continuación, colocar el disco conteniendo CP/M en una determinada unidad de disco. Al cerrar la solapa de seguridad de dicha unidad se produce la carga automática del DOS.

2. Instalar el disco conteniendo CP/M en la unidad de disco oportuna y posteriormente conectar la máquina, en cuyo momento se inicia la transferencia del DOS a la RAM.

Otros computadores, como el AMSTRAD, con algún tipo de sistema operativo en ROM, exigen escribir, por ejemplo, CPM seguido del correspondiente RETURN o ENTER para transferir el DOS a la RAM, o pulsar un determinado grupo de teclas simultáneamente.

La unidad de disco —en el supuesto de haber varios disponibles— donde se debe instalar CP/M para proceder a la carga anteriormente apuntada se denomina unidad por defecto, o unidad por omisión *(default drive)* inicial y su nombre: unidad A.

Los nombres que reciben las diferentes unidades de disco que un computador tiene conectadas constantemente, o en línea *(on line)*, suelen ser A, B, C, D y así sucesivamente, en orden alfabético, una letra por cada unidad disponible.

La versión 2.0 y posteriores de CP/M pueden soportar hasta 16 unidades de disco.

El proceso de carga de CP/M se da por concluido cuando en la pantalla aparece el nombre de la unidad de disco encargada de la transferencia seguido del símbolo >. Así:

A >

Esta impronta *(pronpt)* indica al usuario que el DOS está a nivel de operador *(system-level)*. En otras palabras, CP/M espera recibir algún tipo de orden a través del teclado.

Una vez hayamos colocado el sistema a este nivel, como medida preventiva y sin mayores conocimientos, debemos sacar una copia del disco que contiene el original CP/M y que acompaña al equipo, siguiendo las instrucciones que al efecto, sin duda, están redactadas en el manual de operaciones del ordenador.

El original debe ser puesto a buen recaudo, y utilizar la copia así obtenida para los trabajos alrededor de CP/M.

Dejamos el tema en estas condiciones y pasemos al siguiente epígrafe.

Ubicación de CP/M en la memoria interna

En este punto, vamos a admitir que CP/M ha sido transferido del disco a la memoria interna del computador, con lo cual, y aparte de otros mensajes que varían de máquina a máquina, tendremos en pantalla el *prompt* indicativo de que el DOS está a nivel de operador (A>).

Ahora, CP/M está debidamente encajado en la RAM, de acuerdo con la siguiente estructura:

Los conceptos de BIOS, CCP y BOOT ya nos son conocidos del epígrafe anterior, y los programas que comprenden se sitúan en la RAM en las posiciones de memoria relativas que indica el esquema.

En la misma figura, se observa que la primera página de memoria, se la reserva CP/M para su uso, y que dentro de ella está situado BOOT. El significado de "página de memoria" se estudiará en su momento. (Ver SAVE).

La zona de memoria denominada **TPA** *(Transient program area)* o "área de programas transitorios" está comprendida entre la primera página y el comienzo de CCP. En esta zona se ubican los programas que han de ser ejecutados por el ordenador, bajo el control de CP/M.

El TPA dispone de tanta memoria como resulte de restar de la cantidad de memoria disponible, la requerida por el BIOS más el BDOS y el CCP, según se aprecia en el esquema. En otras palabras, el TPA es la memoria disponible tras cargar CP/M en el computador.

El bloque de memoria ocupado por BIOS, BDOS y CCP puede ser desplazado, dentro de lo que permita la RAM disponible, para ajustarse a las exigencias del usuario.

Como es lógico, una vez finalizada la carga, el DOS queda a la espera de recibir instrucciones sobre la acción a llevar a efecto o, dicho de otro modo, el sistema está bajo el control del CCP.

A esta situación nos referiremos en ocasiones diciendo que estamos a nivel operador, como ha sucedido en un par de ocasiones.

En este momento, ya tenemos una aproximación a lo que implica estar a nivel operador.

Los ficheros en el AMSTRAD

POR A. BELLIDO

Cómo crear ficheros en disco

A lo largo de las páginas que siguen, doy por supuesto un conocimiento no profundo pero sí elemental de la programación en BASIC.

El objetivo encomendado a este libro, es ayudar al lector interesado en los temas que aquí se tratan a dominar el dispositivo periférico que más eficacia va a añadir a su computador: la unidad de disco.

AUN cuando **Cómo crear ficheros en disco** está dedicado a la creación y mantenimiento de ficheros en disco, la primera parte trata sobre los conceptos previos generales que todo usuario de ordenador debe conocer para estar en condiciones de comprender los mecanismos que permiten a la máquina funcionar de acuerdo con los deseos del programador.

En este sentido, el principal problema con que hemos de toparnos, es concebir en su verdadera dimensión cada una de las palabras, conceptos y órdenes que el manejo de ficheros en disco hace necesario emplear.

Las personas que sepan inglés tendrán una ventaja añadida que yo no tengo en cuenta, ya que considero fundamental transmitir a nuestro idioma toda la estructura mental que soporta a cada palabra o expresión anglosajona que nos veamos obligados a utilizar.

En la segunda parte del curso se tratan específicamente los ficheros en BASIC tanto secuenciales como directos.

Todos los epígrafes contienen ejemplos que contribuyen a afianzar las ideas que en ellos se explican, y se cierran con ejercicios que permiten al lector evaluar sus conocimientos hasta el momento.

Los ejercicios de por sí tienen un gran interés didáctico en función de que ciertos detalles, no dichos explícitamente en el texto, tienen la oportunidad de aparecer reflejados en ellos.

Por otra parte, cuando sea conveniente recurrir a acrónimos, se utilizarán los derivados de las expresiones originales, por ser estos los que suelen aparecer en las publicaciones en español y, desde luego, en las anglosajonas. En todo caso, los considero como unos sustantivos nuevos aplicados a unas ideas nuevas, ya que, de utilizar otro criterio, sólo redundaría en aumentar el léxico, ya confuso, actualmente existente.

Por fin, y con respecto a la exposición de los temas, he tratado de utilizar un lenguaje sencillo, con sentencias breves y claras, apoyado, cuando ha lugar, en esquemas y gráficos que permitan una mejor comprensión.

DISK OPERATING SYSTEM

Esta expresión anglosajona y su acrónimo DOS son términos usuales en la literatura que trata los asuntos que aquí nos ocupan.

Para situarnos en el contexto, y dejando atras precisiones de momento, podemos decir

que un DOS es un conjunto de programas que tiene por misión controlar el tráfico de información en el ordenador.

Desde el punto de vista semántico, un DOS puede entenderse, bien como un sistema operativo **de** disco o bien como un sistema operativo **en** disco.

La primera acepción es válida dado que, en sus orígenes, el DOS estaba encargado fundamentalmente, de manejar el disco. Hoy en día, como veremos, el DOS tiene asignadas muchas más tareas.

El segundo significado es igualmente válido, ya que los programas que componen el DOS están contenidos en disco.

Nosotros aquí, al referirnos a este concepto, utilizaremos el término DOS por ser el más generalizado en el terreno informático.

Dicho esto, es conveniente precisar el sentido que debemos darle a palabras tales como información y ordenador, aparecidas hasta el momento, y otras que surgirán como consecuencia de sus definiciones.

A este menester, de extraordinaria importancia para la consecución de los objetivos últimos que nos hemos propuesto, se dedicarán los próximos epígrafes.

INFORMACION. CONCEPTO

En líneas generales, información es una comunicación de conocimientos o, en términos informáticos, transmisión de datos.

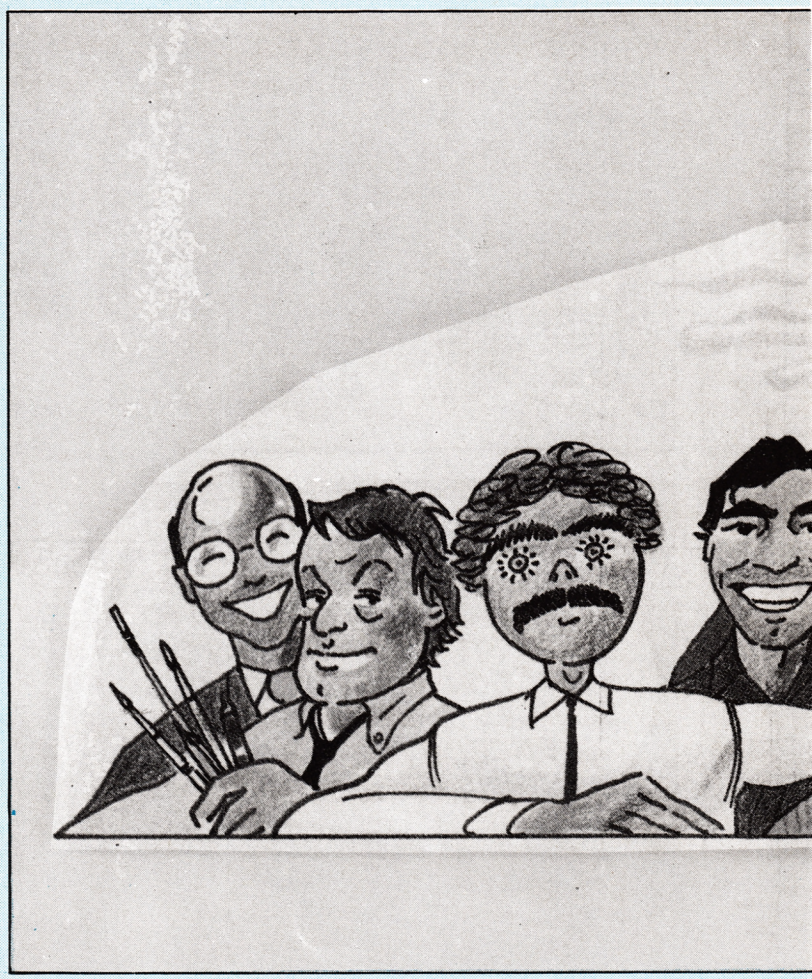
La unidad elemental de información es el **bit.**

La palabra **bit** es una contracción de *binary digit* y alude a cualquiera de los dos dígitos que componen el sistema de numeración en base dos: 0 y 1.

Otra unidad de información, de orden superior, es el **byte.**

Un *byte* es un grupo de ocho bits o, lo que es igual, cualquier combinación de ocho ceros y unos, del tipo 10011101.

Con un *bit* tenemos dos posibilidades de recibir o emitir información: 0 ó 1; y si



admitimos un código que asocie el **cero** a NO (o FALSO) y el **uno** a SI (o VERDADERO), habremos establecido una forma elemental de comunicación.

Así, si preguntamos a algo o a alguien: ¿Aprobó Juan Pérez?", y esto o éste nos responde: 1, sabremos que Juan Pérez SI aprobó.

Es claro, que con un *bit* sólo podemos informar de un acontecimiento entre dos posibles.

Pero, ¿qué sucede cuando la información debe abarcar a cualquier evento?

Para los seres humanos, a estas alturas de su evolución, es relativamente sencillo comunicarse a través del lenguaje hablado y, utilizando el código que representa el conjunto de los símbolos que forman el alfabeto, mediante la escritura.

Pero, ¿qué sucede cuando la información debe establecerse entre hombre y máquina o entre dos máquinas?

La respuesta a esta interesante cuestión pasa por saber que, por exigencias de su electrónica, un ordenador maneja exclusivamente números en base dos.

Cualquier número en base dos tiene su correspondiente en otra base y, más concretamente, en base 10.

El sistema decimal, —o en base 10— es el utilizado normalmente en la vida diaria y, por tanto, a él nos referiremos aun cuando los ordenadores sólo trabajen con números binarios.

Por otra parte, y como ya dijimos, un **byte** es una combinación de ocho **bits** y consiguientemente, representa un número en base dos compuesto por ocho dígitos binarios.

Así, por ejemplo, el binario 00000000 equivale al decimal 0, o el binario 01111110 al decimal 126.

El número más alto que se puede representar mediante un *byte* es 11111111 o, en decimal, 255.

Teniendo esto por cierto, sólo resta preparar un código que dé una correspondencia entre todos los caracteres que son necesarios para la escritura y otros tantos números.

La **American Standard Code for Information Interchange** ha establecido un código que lleva por nombre su acrónimo ASCII, y del cual hay amplia referencia en el manual del AMSTRAD.

Como se puede observar al estudiarlo, los primeros 31 números cumplen las funciones que indican los comentarios, y sirven para completar todos los requisitos que pueden ser exigidos en una comunicación escrita.

A los efecto que aquí se persiguen, la correspondencia entre números y caracteres de uso en la escritura queda establecida entre el 32 y el 127.

Dicho esto, codificar una cadena de caracteres es sencillo. Hagamos la prueba con la palabra COMPUTADOR:

LETRA	DECIMAL	BINARIO
C	67	01000011
O	79	01001111
M	77	01001101
P	80	01010000
U	85	01010101
T	84	01010100
A	65	01000001
D	68	01000100
O	79	01001111
R	82	01010010

La tercera columna representa la secuencia de *bytes* que el computador transmitiría para enviar el mensaje que, una vez decodificado, diera por resultado la palabra COMPUTADOR.

Concluyendo. Dentro del ordenador sólo se manejan *bytes*, o lo que es igual, secuencias de ocho bits.

Y es mediante *bytes* como se comunica el computador con sus diferentes periféricos.

EL BASIC DEL AMSTRAD

Introducción al curso

POR A. BELLIDO

LA palabra BASIC es un acrónimo formado por las iniciales Beginners All-purpose Symbolic Instruccion Code.

El BASIC es un lenguaje de programación de alto nivel que fue desarrollado en el Dartmouth College en los Estados Unidos de América del Norte, y de esa idea inicial surgieron diferentes dialectos, o distintos BASIC, adaptados por cada fabricante a sus máquinas.

El computador AMSTRAD, incorpora un tipo de BASIC muy amplio, completo y generalizado.

Este curso, cuyo INDICE está en el epígrafe siguiente, está dirigido a mostrar al lector las posibilidades que ofrece el BASIC del AMSTRAD, comenzando con los más elementales conceptos y llegando hasta la creación y mantenimiento de ficheros en disco.

Para conseguir este objetivo, cada capítulo tiene una parte teórica con ejemplos relativos a los temas tratados, una ficha-recuadro donde se enfatizan ciertos puntos o se estudian aspectos marginales de interés específico y, finalmente, se proponen ejercicios, cuya solución, a algunos de ellos. Se ofrece en el siguiente número.

Concluiré esta breve introducción con un comentario respecto a los lenguajes de programación en general y a la postura que el programador debe adoptar frente a ellos.

Todo lenguaje requiere de unas normas de escritura (sintaxis) y de un vocabulario (palabras). Una persona que desee expresarse en ese idioma, ha de conocer esa sintaxis y estar al corriente de ese vocabulario.

Con respecto a la organización del curso, anticipo que los temas a estudiar están agrupados en grandes unidades.

En el primero, CONCEPTOS PREVIOS, se informa al lector sobre el significado y aplicación de las palabras e ideas que utilizará en lo sucesivo.

En el segundo, INSTRUCCIONES FUNDAMENTALES, se desarrolla un estudio de programación en BASIC, con el apoyo que dan la propia instrucción que se está analizando, y algunas de las anteriores.

En el tercero, FUNCIONES INCORPORADAS, se amplía el vocabulario BASIC mediante fichas de todas las funciones incorporadas al AMSTRAD. Algunas de las instrucciones que figuran en este apartado no son propiamente funciones, pero figuran en él por razones didácticas y así se hace constar en su lugar; en el INDICE figuran en negritas.

En el cuarto, INSTRUCCIONES RELACIONADAS CON EL TIEMPO, se analizan las posibilidades de control temporal que tiene a su alcance el programador.

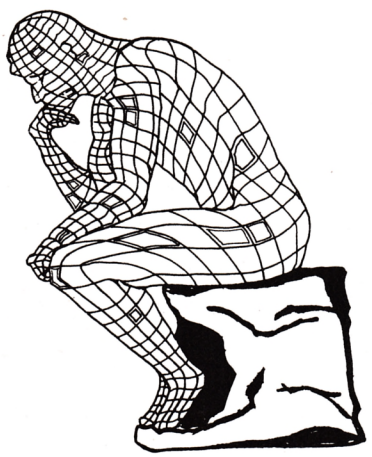
En el quinto, INSTRUCCIONES RELACIONADAS CON LA PRESENTACION, se ve en detalle la tremenda oferta del AMSTRAD para mostrar en pantalla gráficos y mensajes. También la salida a impresora.

En el sexto, MATRICES, se profundiza en el manejo de estos cómodos artificios, conocidos, también, por *arrays*.

En el séptimo, SONIDO, los amantes de la música tienen la oportunidad de conocer, en la práctica, la capacidad del AMSTRAD.

En el octavo, FICHEROS, se introduce al lector en el manejo de discos para el proceso de datos, con una introducción a todos los conceptos que exige la comprensión de la técnica implicada.

INDICE DEL CURSO



Fichas del

Uno de los principales problemas con que se enfrenta todo nuevo usuario del AMSTRAD, se deriva de la voluminosidad de su manual.

Como una ayuda, y con el fin de sustituir al manual en determinado tipo de consultas, se ofrece a continuación una guía práctica y resumida de los comandos y funciones, que el AMSTRAD pone a disposición del programador en BASIC.

Guía de palabras BASIC

La notación aplicar, responde a los siguientes criterios:

1. Las palabras encuadradas corresponden a las del BASIC.

2. La información contenida entre dos signos de interjeción (¡...!) **debe ser** suministrada por el programador imprescindiblemente.

3. La información contenida entre dos barras verticales (|...|) **puede ser** suministrada por el programador opcionalmente.

4. Si la palabra **expresión** va entrecomillas, se refiere a una expresión alfanumérica. En caso contrario es una expresión numérica.

AMSTRAD: GUIA DE PALABRAS BASIC. SU SINTAXIS Y APLICACION

ABS ¡(expresion)!

Convierte la **expresión**, sea cual sea su signo, en la misma pero positiva.

```
PRINT ABS (-1,2)
 1,2
```

AFTER ¡expresion 1!|, espresión 2| gosub

Llama a una subrutina cuando el tiempo que indica **expresión 1** ha transcurrido. Con

AMSTRAD

POR A. BELLIDO

expresión 2 se determina, opcionalmente, el número del retardador: 0, 1, 2, 3. Por defecto es el 0. Cada retardador puede dirigirse a su propia subrutina.

```
5x = 2 : y = 5
10 AFTER x*y*10 GOSUB 30 : AFTER
  x*100, x GOSUB 40
20 FOR N = 0 TO 800 : PRINT "P"; : NEXT
30 PRINT "RO RO RO RO" : RETURN
40 PRINT "R2 R2 R2 R2" : RETURN
```

Después de dos segundos, se interrumpe la impresión del carácter P para dar lugar a las cuatro RO y, tras 4 segundos, a las cuatro R2.

ASC ¡("expresion")!

Se obtiene el código numérico del primer carácter de "expresión

```
PRINT (Ab)
65
```

ATN ¡(expresion)!

Determina el arcotangente de la expresión

```
PRINT ATN (-.41614684)
-0,394348109
```

AUTO | número 1 | |, número 2|

Hace aparecer en pantalla automáticamente las líneas del listador a partir del **número 1** y con la cadencia dada por **número 2**. Para pasar a la siguiente, basta pulsar ENTER.

Por defecto, **número 1** y **número 2** son 10.

Para cancelar AUTO, apretar ESC.

BIN$ ¡(EXPRESION 1) |, EXPRESION 2|)!

Convierte la **expresión 1** en una cadena representativa del binario equivalente, con tantos dígitos como indique **expresión 2.**

Si **expresión 2** no se da, o es demasiado pequeña, la cadena cubre los dígitos binarios mínimos necesarios.

```
10 × = 8 : y = 2040
20 PRINT BIN$ (y/x,x)
11111111
```

BORDER ¡expresion 1! |, expresion 2|

Fia el color dado por **expresión 1** en la zona de pantalla que bordea el área de impresión. Si **expresión 2** se da, el borde cambia de color entre los colores determinados por ambas expresiones, las cuales deben resultar en números comprendidos.
meros comprendidos entre 0 y 26.

```
10 FOR x=0 TO 26
20 BORDER x,26-x
30 FOR y=0 TO 1000 : NEXT
40 NEXT
50 BORDER 1
```

El borde parpadeará —excepto para x=13- entre dos colores.

Pruebe el efecto quitando el segundo parámetro en la línea 20.

CAT

Muestra el catálogo de ficheros del disco instalado en el **drive** cuya identificación aparece en pantalla, junto con la capacidad disponible aún. Este comando no afecta al contenido que, a la sazón, tenga el computador.

CHAIN ¡"expresion"|, expresion|

Carga en memoria, procedente del disco, el programa denominado "expresión" ejecutándose desde la línea dada por **expresión.**

Si **expresión** no se da, el programa se autoejecuta desde la primera línea del listado.

Carga en memoria, procedente del disco, el programa denominado "expresión" ejecutándose desde la línea dada por **expresión.**
? Si expresión no se da, el programa s Si **expresión** no se da, el programa denominado "expresión" ejecutándose desde la línea dada por **expresión.**
Si **expresión** no se da, el programa se autoejecuta desde la primera línea del listado.

```
x=25 : A$="Laser" : CHAIN A$, x*2
```

Esta secuencia de instrucciones, cargará el programa titulado **Laser** y se autoejecutará a partir de la línea 50.

CHAIN MERGE ¡"expresion"!, expresion | |, delete número 1 - número 2 |

Actúa exactamente igual que CHAIN, pero mezclando el programa titulado "expresión" con el que, a la sazón, esté en memoria, sustituyendo en éste aquellas líneas de programa que, procedentes de aquel, coincidan.

Si la opción DELETE se usa, todas las líneas comprendidas entre **número 1** y **número 2** quedarán borradas.

CHAIN MERGE A$, 50, DELETE 9000 - 9500

Esta instrucción, cargará en memoria el programa cuyo título está representado por A$, y lo ejecutará a partir de la línea 50, borrando las líneas comprendidas entre 9000 y 9500.

Esta instrucción, cargará en memoria el programa cuyo título está representado por A$, y lo ejecutará a partir de la línea 50, borrando las líneas comprendidas entre 900 y 9500.

CHR$ ¡(expresion)!

Convierte el número resultante de **expresión**, en su código ASCII equivalente, el cual debe estar comprendido entre 32 (código del espacio) y 255 (código de la doble flecha).

```
x=2 : y=34 : PRINT CHR$ (x*y)
```

CINT ¡(expresion)!

Convierte la **expresión** en el número entero más próximo

```
PRINT CINT (-1,9), CINT (-1,5), CINT (-1,1)
da:     -2      -2      -1
```

CLEAR

Borra todas las variables, matrices y funciones definidas por el usuario. Cancela cualquier situación de OPEN en un fichero.

CLEAR INPUT

Asegura que el **buffer** de entradas por teclado está vacío.

CLG |expresion|

Borra los gráficos de la pantalla dejando el color de fondo de éstos al determinado por **expresión**, o en su defecto, al actual.

```
10 PRINT "222"
20 DRAW 50,50
30 FOR x=1 TO 1000
40 CLG 15
50 PRINT "333"
60 DRAW 100,100
```

CLOSEIN

Cierra la entrada de información procedente de un fichero en disco.

CLOSEOUT

Cierra la salida de información hacia un fichero en disco.

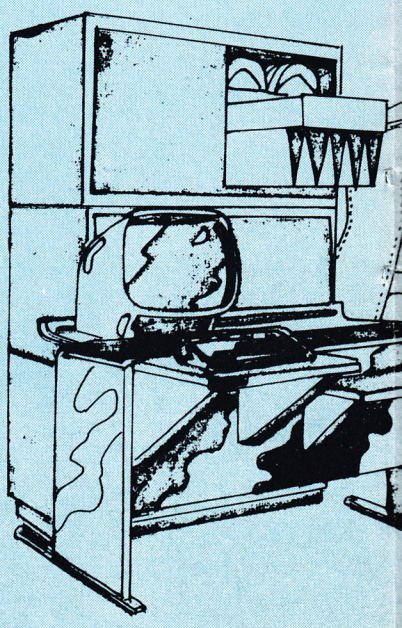
CLS # expresion |

Borra la ventana indicada por # **expresión** a su color actual de fondo. La **expresión** debe dar de un número entre 0 y 7. Si # **expresión** se omite, se asume # 0, borrándose lo que en estos momentos se ve en pantalla.

```
10 PRINT "333"
20 WINDOW # 7,1,80,1,10 : PAPER # 7,15 :
   CLS # 7
30 PRINT # 7, "222"
40 FOR x=1 TO 3000 : NEXT
50 CLS
```

CONT

Permite continuar la ejecución de un programa tras un BREAK producido por dos pul-



saciones sucesivas de ESC, o por un STOP en modo programa.

COPYCHR$ ¡(#expresion)!

Permite copiar el carácter que está situado en la actual posición del cursor en la ventana dada por **expresión.**

```
10 CLS : PRINT "HOLA" : LOCATE 4,1
20 WINDOW # 7,1,80,15,25
30 a$=COPYCHR$ (#0)
40 LOCATE #7,10,2
50 PRINT #7 a$
```

Con este programa se transfiere la A de HOLA, de la ventana #0, a la posición 10,2 de la ventana # 7.

COS ¡(expresion)!

Determina el coseno de la **expresión** en radianes (RAD) o grados (DEG), por defecto en radianes.

```
PRINT COS (PI/2)
0
```

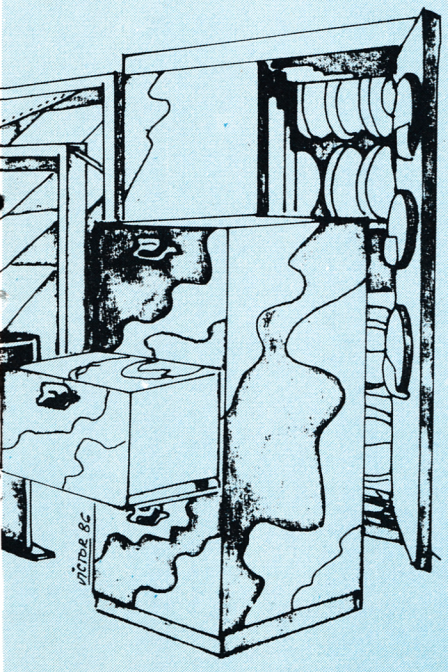
CREAL ¡(expresion)!

Se obtiene el número real correspondiente a la **expresión**

```
PRINT CINT (10/3) ,, CREAL (10/3)
3              3,33333333
```

CURSOR ¡expresion!

La **expresión** debe resultar un 1 o un 0. Con 1, el CURSOR aparece, con 0 desaparece.



```
10 x=1 : CURSOR x
20 PRINT INKEY$ ;
30 IF INKEY$="q" THEN CURSOR 0 :
STOP
40 GOTO 20
```

Con este programa, un cursor irá indicando la posición del próximo carácter a imprimir, hasta que se pulse la q.

DATA ¡constante 1, constante 2,..., constante n!

Cada **constante** puede ser numérica o alfanumérica. Almacena valores y los deja a disposición del comando READ.

Las constantes son leidas una tras otra por los READ de un programa.

```
DATA 255, JUAN, 1, AMSTRAD, 3, 127, 2
```

DEC$ ¡expresion, "formato"!

Convierte la **expresión** en una cadena representativa del número dado por ella y con el **formato** indicado.

Para los "especificadores" del formato, ver APENDICE A

```
PRINT DEC$ (2*PI*30, "###.####")
188.4956
```

DEF FN ¡variable!(parámetros) ¡=expresion!

Permite al usuario definir sus propias funciones, expresión, con el nombre dado por variable, y pudiendo definir los **parámetros** implicados cuando sea necesario.

```
10 DEF FN A(x,y) = x+y*2
20 FOR x=1 TO 10
30 LET Z=FN A(x,10.1)
40 PRINT Z;
50 NEXT
21.2 22.2 23.2 24.2 25.2 26.2 27.2 28.2 29.2 30.2
```

DEFINT ¡letra 1, letra 2,..., letra n! o defint ¡letra 1-letra n!

Convierte el valor de cualquier variable que comience con **letra 1, letra 2,** etc. o esté incluida en el campo determinado por **letra 1-letra n**, en el número entero más próximo.

Si al programa del ejemplo anterior se le añade:

```
5 DEFINT Z
```

La serie se transforma en

```
21 22 23 24 25 26 27 28 29 30
```

Programas comentados

CUENTOS FANTASTICOS

He aquí un programa, relatador incansable de cuentos fantásticos e irrepetibles.

Como todos los cuentos, comienza con el "Erase una vez..." pero la historia es cada vez distinta y su final impredecible.

Todo el programa gira en base a cinco fases básicas, que constituyen la estructura del cuento y que son complementadas con otras fases o palabras, elegidas aleatoriamente por el ordenador. A cada frase básica, le corresponden cinco o diez complementarias.

Estructura general del cuento:

1. ERASE UNA VEZ... (personaje del cuento)
2. QUE (acción que realizó)
3. SU (objeto sobre el que realizó la acción) (lugar en que la realizó).
4. Y COMO CONSECUENCIA (acción resultante) (lugar en que se produjo).
5. Y NUNCA MAS SE SUPO (frase final).

Como puede verse las frases 1, 2 y 5 tienen una frase complementaria, mientras que las 3 y 4, tienen dos.

Los cuentos o relatos resultantes son cortos, pero será muy fácil añadir nuevas frases básicas y complementarias, modificando el pgm. en el dimensionamiento de las matrices y añadiendo más frases y palabras en las líneas DATA.

Como sugerencia, te aconsejamos que tomes un libro o poema de todos conocidos y para cada frase básica construyas una serie de palabras complementarias, algunas de ellas disparatadas, de esta forma te aseguramos que ni el propio Cervantes sería capaz de reconocer su "Quijote", escrito por tu Amstrad.

La zona izquierda del cuadro, constituye una matriz (array) de caracteres, unidimensional, formada por 5 elementos, que son las frases básicas.

En la tabla siguiente se establecen las frases y correspondencia entre las frases y palabras usadas por el programa desarrollado.

La zona izquierda del cuadro, constituye una matriz (array) de caracteres, unidimensional, formada por 5 elementos, que son las frases básicas.

La zona derecha, forma una matriz bidimensional de caracteres, con 7 filas y 5 columnas, en la que cada elemento, representa una palabra o frase complementaria.

La estructura del programa es sencilla, una rutina de carga nos servirá, para introducir los textos, contenidos en líneas DATA, en las matrices correspondientes, debidamente dimensionadas.

A la primera matriz, que contiene las frases básicas, le daremos de dimensión cinco y por nombre F, como consecuencia la dimensionaremos cmo F$ (5). A la segunda, que nombraremos C, y que tiene 7 filas por 5 columnas, las dimensionaremos C$ (7,5).

A continuación presentamos el conjunto del pgm. comentado línea a línea, de manera que sea fácilmente comprensible y nos facilite el hacer modificaciones, como indicamos anteriormente.

70 Direccionamiento a subrutina de carga de frases.
80-100 Borrado de pantalla. Inicialización de A y B y actualización de la var. A (a, contiene el número de frase básica).
110 Control de fin de pgm., cuando el número de frases básicas se ha agotado, A 6).
120 Escribe frase básica.

	FRASES COMPLEMENTARIAS					
FRASES BASICAS	1	2	3	4	5	
ERASE UNA VEZ UN...	VIEJO	LAGARTO	NENE	VAMPIRO	BORRICO	1
QUE	ENVOLVIO	COMIO	DURMIO	PINTO	ROMPIO	2
SU	CABEZA	TAZA	PIE	NARIZ	CARRO	3
	EN EL WATER	EN LA CAMA	POR VOLVERSE VERDE	CON CHOCOLATE	CON ARENA	4
Y COMO CONSECUENCIA	SE PERDIO	APRENDIO BASIC	EXPLOTO	CRECIO MUCHO	ENGORDO	5
	EN UN RIO	COMO UN RAYO	EN LA COCINA	EN UN VASO	CON PIENSO	6
Y NUNCA MAS SE SUPO	COMO MURIO	DONDE SE ENCUENTRA	QUE SE LO LLEVO	COMO VOLO	CON QUIEN SE CASO	7

en BASIC

130 Actualiza var. B. (B contiene la fila de las complementarias).
140 Establece un valor aleatorio para la var. N, comprendido entre 1, 5, que corresponde a la columna de frases complementarias (INT RND X 5) genera números entre 0 y 4, por eso se le suma una unidad).
150 Escribe la frase complementaria correspondiente a la fila B, columna N (ver frases en el cuadro).
180 Direccionamiento control nuevo juego (380-440)
190-200 Borra líneas anterior de la pantalla y va al comienzo del juego (90), después de la rutina de carga.
210 Subrutina de carga de frases.
220 Dimensiona matriz de frases básicas.
230 Carga frases en la matriz.
240 DATA conteniendo frases básicas.
250 Dimensiona matriz frases complementarias.
260-290 Carga frases.
300-360 Datas frases complementarias.
370 Fin de subrutina y retorno al pgm. principal.
380-440 Subrutina de control para un nuevo relato, las líneas 400 a 410 constituyen un bucle indefinido, hasta tanto no se presione una tecla. Si la tecla pulsada es "C" se efectua el retorno al pgm., si es "P" fin de pgm. cualquier otra tecla, provoca la entrada al bucle de lectura del teclado (400-410).

```
10 REM  .................
20 REM
30 REM     RELATOS FANTASTICOS
40 REM
50 REM
50 REM     .................
60 REM
70 GOSUB 210
80 CLS
90 A=0: B=0
100 A=A+1
110 IF A=6 THEN GOTO 180
120 PRINT : PRINT L$(A);" ";
130 B=B+1
140 N=INT(RND*5)+1
150 PRINT P$(B,N);
160 IF B=3 OR B=5 THEN GOTO 130
170 GOTO 100
180 GOSUB 380
190 PRINT CHR$(11)+"
190 PRINT CHR$(11)+"                ":PRINT
200 GOTO 90
210 REM CARGA DE LAS MATRICES DE DATOS ..............................
220 DIM L$(5) :REM FRASES BASICAS
230 FOR K=1 TO 5:READ L$(K): NEXT K
240 DATA " ERASE UNA VEZ UN...", "QUE", "SU", "COMO CONSECUENCIA," "Y NUNCA MAS SE SUPO"
250 DIM P$(7,5)
260 FOR F=1 TO 7
270 FOR C=1 TO 5
280 READ P$(F,C)
290 NEXT C,F
300 DATA"VIEJO","LAGARTO","VAMPIRO","BORRICO","JOVEN"
310 DATA"ENVOLVIO","COMIO","DURMIO","PINTO","ROMPIO"
320 DATA"CABEZA","TAZA","NARIZ", "MORRO","CARRO"
330 DATA " EN EL WATER"," EN LA CAMA"," AL VOLVERSE VERDE", "CON CHOCOLATE", " EN UNA NUBE"
340 DATA "SE PERDIO","APRENDIO BASIC", "EXPLOTO", "CRECIO MUCHO", "ENGORDO"
350 DATA " EN UN RIO"," COMO UN RAYO"," EN LA COCINA"," EN UN VASO"," CON PIENSO"
360 DATA "COMO MURIO","DONDE SE ENCUENTRA","QUE SE LO LLEVO", "PORQUE SALIO VOLANDO","CON QUIEN SE CASO"
370 RETURN
380 PRINT "==========================================="
390 PRINT : PRINT "PRESIONA: <C> CONTINUAR, <P> PARAR"
400 Z$ = INKEY$
410 IF Z$="" THEN GOTO 400
420 IF Z$="C" OR z$="c" THEN RETURN
430 IF Z$="P" OR Z$="p" THEN END
440 GOTO 400
```

INTRODUCCION A LOS

CAPITULO I

POR JUAN MARTINEZ PINTOR

INTRODUCCION

*LA palabra Logo deriva de la palabra latina "logos", que significa "pensamiento", "idea".

* Características del lenguaje:

— No requiere conocimientos previos de ninguna clase, ni tiene cota en cuanto a su potencial.

— Es un lenguaje estructurado. Los programas se crean combinando procedimientos.

— Es un lenguaje interactivo. La comuni cación hombre/máquina, puede hacerse de forma inmediata a través de la consola.

— La estructura del lenguaje está muy relacionada con los objetos que maneja: números, palabras y listas.

— El Logo cuenta con los famosos gráficos tortuga. Esto facilita su aproximación con éxito a los niños.

— Es un lenguaje diseñado con objetivos educativos.

* Los mensajes de error los emite el ordenador cuando hacemos algo incorrectamente.

* Al trabajar en Logo, hay muchas palabras que el ordenador conoce: son los primitivos del sistema.

* Los conceptos de "palabra" y "lista", son muy importantes desde el punto de vista estructural.

* Una palabra es una serie de caracteres consecutivos, ninguno de los cuales puede ser un espacio en blanco, ni ningún otro separador de palabras.

* Una serie de palabras, perfectamente delimitada es una lista.

* Un "objeto-Logo", es una palabra o una lista.

* En general una lista es una serie de objetos-Logo perfectamente delimitada en su ordenación, comienzo y fin.

* Una manera de indicar el comienzo y el final de una lista, es mediante el uso de caracteres "[" y "]", respectivamente. Entre dos objetos consecutivos de una lista, solo media uno o varios espacios en blanco.

* Los programas Logo se llaman "proce dimientos". Tienen un nombre que los identifica.

* Del mismo modo que una lista puede formar parte de otra, un procedimiento puede formar parte de otro.

I.1. EL LOGO Y LOS LENGUAJES DE PROGRAMACION

Los ordenadores han inundado nuestros hogares y nuestras aulas. Del buen uso que se les dé en los ambientes educativos, dependerá en gran medida la formación de los hombres de las próximas generaciones. El lenguaje de programación Logo ha sido diseñado para acercar la informática a los niños del modo más natural y más rentable posible desde el punto de vista educativo.

A diferencia de las palabras FORTRAN, COBOL o BASIC, la palabra Logo no está

PROGRAMAS EN LOGO

formada por siglas. Al contrario, deriva de la palabra latina logos, que a su vez proviene del vocablo griego que designa "idea", "pensamiento".

El lenguaje Logo ha sido desarrollado por el profesor Seymour Papert y su equipo, en el laboratorio de inteligencia artificial del Instituto Tecnológico de Massachussets (MIT), hace poco menos de 20 años. Papert es un matemático que ha dedicado mucho tiempo a investigar los procesos de aprendizaje y, por tanto, a pensar sobre el propio pensamiento, sobre las ideas. Por ello ha diseñado un lenguaje de programación de ordenadores con el que aprender a pensar. A diferencia de lo que se hace en otros ambientes, en el que el alumno sigue los pasos que le marca un programa (por ejemplo de ejercicios), en ambiente Logo, es el alumno el que dice al ordenador lo que debe hacer: el niño programando al ordenador en vez de lo contrario, según afirma el propio Papert.

En su libro **MINDSTORMS: Children, Computers and Powerful Ideas** (publicado en castellano con el título "Desafío a la mente: computadoras y educación", por Ediciones Galápago), Papert mantiene la tesis de que el Logo además de ser una modalidad para el uso de los ordenadores, es también un instrumento para explorar el conocimiento.

Para Papert son de la mayor importancia desde el punto de vista educativo, las ideas que el individuo ha hecho suyas, comprendiéndolas en profundidad y asimilándolas. Son lo que denomina los propios modelos. Para él, el hecho fundamental del aprendizaje (expuesto en el libro anterior), es éste: cualquier cosa es fácil si uno puede asimilarlo a la propia colección de modelos. Si eso no es posible, cualquier cosa puede resultar angustiosamente difícil. De modo que lo que un indi viduo puede aprender y como lo aprende depende de los modelos con que cuenta.

De todo lo anterior resulta la importancia de situar lo antes posible a los niños en un ambiente en el que el propio pensamiento sea de la mayor importancia.

Al avanzar en el estudio del lenguaje, veremos como la estructura de sus programas posee un estilo diferente al de otros. Una estructuración que nos cautiva por su compleja sencillez.

Algunas características del lenguaje, son:

— Para comenzar a estudiar LOGO es suficiente un conocimiento básico de matemáticas. Incluso una persona que no posea conocimiento alguno podría comenzar a estudiar LOGO con éxito. Por otra parte, tampoco tiene una cota en cuanto a su potencial. De modo que las personas que quieran trabajar con LOGO, tienen abierto un gran campo.

— LOGO es un lenguaje **estructurado**. Los programas en este lenguaje se crean combinando estructuras de "rutinas", llamados "procedimientos", cada uno de los cuales podría estar construido del mismo modo. La importancia de este hecho, se irá viendo al ir progresando en el estudio del lenguaje.

— Es un lenguaje **interactivo**. Esto significa, que la comunicación hombre-máquina, puede hacerse de forma inmediata a través del teclado y la pantalla. Así, para ejecutar muchas de las órdenes del lenguaje y sus programas, basta escribirlas en el teclado.

— La forma en que LOGO maneja la información incluye "números", "palabras" y "listas". Además, la propia estructura del lenguaje está muy relacionada con estos objetos, lo que le hace particularmente capacitado para algunos tipos de problemas en los que haya que manejar símbolos.

— Una característica muy importante del LOGO, es el hecho de que posee los llamados "gráficos-tortuga". Los gráficos tortuga pre-

sentan muchas características interesantes. Por ejemplo, son un medio fabuloso para iniciar el estudio de la programación. Además, permiten al iniciado realizar estudios de matemáticas basadas en ordenador.

— De lo anterior deducimos ya que LOGO es un lenguaje diseñado también con objetivos educativos. En efecto, la facilidad de los gráficos tortuga permiten el acercamiento con éxito a los ordenadores incluso a los niños muy pequeños. Esto, ya de por sí muy interesante, unido a la estructura de sus programas, puede permitir que las nuevas generaciones de estudiantes, posean una organización mental de gran riqueza y, desde luego, completamente nueva. Con lenguaje de programación como LOGO (y sus sucesores), podremos conseguir que una mayoría de los niños se interesen de modo natural (e incluso amén), el propio razonamiento que les permite progresar en sus juegos y en sus hallazgos con el ordenador. Este amor por el pensar si será una revolución educativa. En este sentido, hemos de tomar al LOGO como un lenguaje más para aprender que para enseñar.

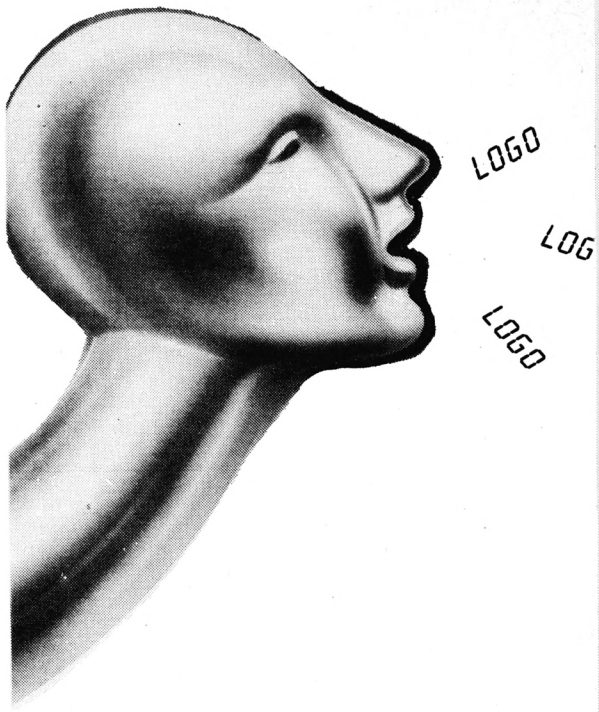
I.2. ESTRUCTURA DE LOS PROGRAMAS LOGO: PRIMITIVOS Y PROCEDIMIENTOS

Una de las características del lenguaje Logo por las que más nos gustará trabajar con él, es porque es un lenguaje estructurado. Y por la estructura que tienen sus programas. A continuación veremos la primera aproximación a estas ideas.

Vamos a suponer que encendemos el ordenador, y que instalamos en su memoria el lenguaje Logo. En la ficha número 1, vemos como se hace esto. A modo de experimento escribamos la palabra "hola" (sin las comillas) y pulsemos después la tecla "enter". Entonces el AMSTRAD responde escribiendo en pantalla el mensaje.

I don't know how to hola

(no sé cómo se hace hola). Los mensajes de error los emite el ordenador como respuesta a



algo que hemos hecho incorrectamente. En otro capítulo volveremos sobre ellos.

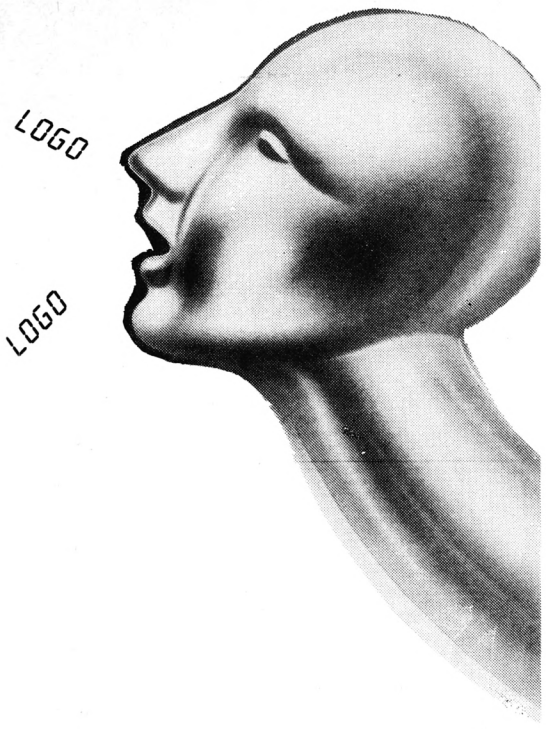
De modo que la palabra "hola" no la interpreta bien el ordenador; no la conoce. Escribamos ahora "ct" y pulsemos "enter". Vemos que la pantalla se borra, y no aparece ningún mensaje de error. De modo que la palabra **ct** sí la conoce el Logo: es una de sus instrucciones.

Todos los lenguajes manejan palabras como herramienta importante: es natural que los lenguajes estén formados así. Pero los conceptos de **PALABRA** y **LISTA**, son fundamentales desde el punto de vista estructural en Logo. Una palabra en Logo es una serie de caracteres consecutivos, ninguno de los cuales puede ser un espacio en blanco. El espacio en blanco es el separador de palabras. Hay algunos otros caracteres que tampoco pueden formar parte de las palabras, en general, porque son de algún modo también separadores de ellas. Tal es por ejemplo el caso del carácter +.

EJEMPLO I.1
— Son palabras **pa1,** pa.2, pa.3, **349.**
— No son palabras np+1, np 5.

Conforme vayamos avanzando en el estudio del Logo, iremos viendo qué caracteres no pueden formar parte de una palabra.

Como vimos antes, al cargar el Logo hay una serie de palabras que conoce, y por lo



tanto con ellas podemos ordenarle determinadas cosas. Estas palabras son las que en Logo se llaman **PRIMITIVOS**.

Naturalmente, los programas llevarán a efecto acciones mucho más complicadas que las desarrolladas por los primitivos. Antes de referirnos a la estructura de los programas, veremos el concepto de lista, al que ya hemos hecho referencia, y que necesitaremos más adelante.

El concepto de lista es muy potente y rico, entre otras muchas cosas porque subyace a toda la estructura del lenguaje. Una serie de palabras, con indicación de donde empieza y donde acaba, es una lista en Logo. En general un **objeto-Logo** es una palabra o una lista. Pues bien, resulta que una lista es una serie de elementos cada uno de los cuales es un objeto Logo, es decir, una palabra o una lista. De este modo, podemos tener listas de varios niveles formando parte, como objetos-Logo de una lista; y palabras al mismo nivel que listas... Una manera (no la única), de indicar a Logo que una serie de objetos ha de ser tomada como una lista, es encerrando sus elementos entre los caracteres "[" y "]".

EJEMPLO I.2

Como pa1, pa2 y pa3 son palabras, una lista formada por ellas en ese orden será: [pa1 pa2 pa3].

Como pa4 es otra palabra, otra lista más compleja que incluya como elementos palabras y listas será: [[pa1 pa2 pa3] pa4]. Esta lista está formada por dos objetos-Logo: la lista [pa1 pa2 pa3] y la palabra pa4.

Observemos que los distintos objetos-Logo de una lista, sólo se separan entre sí por espacios en blanco.

¿Da lo mismo el orden en que se sitúen los diferentes objetos de una lista? **NO**. La lista anterior y la lista [pa4 [pa1 pa2 pa3]], son distintas.

Quizás estemos ya intuyendo por donde van las cosas desde el punto de vista estructural, en la confección de programas en Logo. Un pequeño programa podría muy bien estar formado solamente por una serie de órdenes elementales (primitivos), del mismo modo que una lista puede estar formada sólo por palabras. Pero tal vez queramos más adelante hacer un programa un poco más complejo, en el cual el anterior sólo representa una parte. Podemos imaginar que los programas tengan un nombre, y que nos referimos a ellos precisamente por su nombre: una palabra. Entonces esa palabra (que no es ahora un primitivo), está formada por otras, por ejemplo primitivos... ¡sí, del mismo modo que una lista debajo nivel está formada por palabras! Pero el programa 'un poco mayor' que utiliza a varios del tipo de nuestro programita se parece entonces a una lista que estuviera formada por varias listas y, tal vez algunas palabras.

Vemos pues, que aunque el ordenador al empezar a trabajar no sabía cómo hacer "hola", es posible definir un programa con ese nombre y asi enseñar al ordenador cómo debe hacer esa orden.

Esta es en efecto la estructura del lenguaje. Los programas complejos se dividen en partes, que realizan acciones un poco más simples, cada una de las cuales a su vez se dividen posiblemente en nuevas partes... en niveles arbitrarios de complejidad. Son lo que se llaman **PROCEDIMIENTOS** en Logo. Del mismo modo que una lista puede formar parte de otra, un procedimiento puede formar parte de otro. En realidad, más adelante, cuando estemos ya trabajando con el LOGO del ordenador AMSTRAD, realizaremos el proceso de incluir unos procedimientos en otros, casi sin darnos cuenta.

DOCTOR LOGO

Programas comentados en LOGO

ES indispensable en programación que el estudio teórico sistemático de todas las instrucciones y detalles del lenguaje, vaya complementado con la práctica continuada casi desde el primer momento.

En estos "ejercicios" el lector interesado podrá profundizar en la práctica de la programación. En efecto, presentaremos los listados de muchos y variados programas escritos en el lenguaje Logo y a su derecha se comentarán las instrucciones y procedimientos. Cuando sea necesario, se incluirán recuadros en los que se aclaren ideas más complejas o largas y en los que se desarrollen y aclaren conceptos de otras materias que sea necesario utilizar. El lector no interesado en esos detalles podrá pasarlos por alto sin más.

Cada programa irá acompañado también por unas instrucciones de uso. Esto permitirá al usuario interesado plantearse el propio programa como un ejercicio e intentar desarrollarlo el mismo. Posteriormente se puede consultar el listado aquí propuesto y comparar los detalles. Además, nada impide tampoco que se hagan nuevas versiones mejoradas de los programas propuestos.

En el extremo opuesto, las personas cuyo interés esté sólo en utilizar los programas pueden ignorar la parte de listado y comentarios, concentrándose directamente en la máquina. No es nuestro consejo. Aparecerán, aquí programas de todos los tipos: desde sencillas utilidades que nos ayudarán posteriormente a trabajar en los nuestros, hasta sofisticados programas (no necesariamente de juegos). Para ayudar a elegir la manera en que debemos afrontar cada programa, estos llevarán al lado de su nombre un número de 'D': una sola D para indicar que el programa es fácil, dos para indicar que es medio y tres para los considerados difíciles. El criterio adoptado se refiere, naturalmente, a personas que no dominan el lenguaje.

Para teclear los programas escritos en Logo hemos de situar el ordenador en este lenguaje. Encendamos pues la máquina, introduzcamos el disco que contiene el Dr. Logo y escribamos |CPM. Después de algunos segundos la máquina está dispuesta para trabajar en Logo. En un primer paso, y mientras avanzamos en nuestro estudio del Logo, lo mejor es teclear los programas tal y como

aparezcan en el listado. Cuando queramos guardar en un disco todos los programas que tenemos en la memoria, buscamos un nombre (por ejemplo, **nombre**) y entonces escribimos: save "nombre". Para cargar este trabajo posteriormente en la memoria, desde el disco que lo contiene, se escribe: load "nombre".

MINIMAX (LOGO) (Procedimientos con resultado) (D)

Instrucciones de manejo:

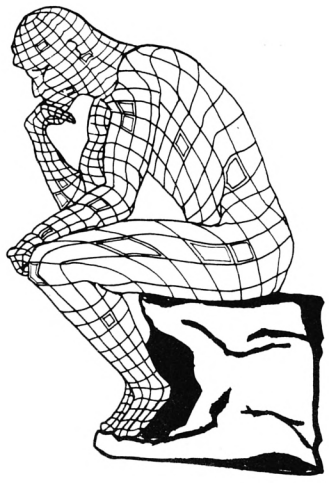
El programa "MINIMAX" contiene dos utilidades. Una para obtener el mínimo de dos números y otra para obtener el máximo.

El procedimiento para hallar el mínimo se llama "min"; requiere dos números como datos de entrada y da como resultado el menor de ellos.

De igual modo el procedimiento "max" da como resultado el mayor de sus dos datos de entrada.

Como es usual en Logo, estos datos de entrada no han de ser números necesariamente, sino que pueden ser procedimientos que den números como resultado. Asimismo, es posible concatenar cuantos procedimientos deseemos, sin más que tener en cuenta que las prioridades son las usuales: primero se ejecutan los procedimientos más internos y progresivamente los más externos.

POR JUAN MARTINEZ PINTOR



LOGO

EJEMPLOS:

```
?pr min 2 7
2
?pr max 3+4 23
23
?pr max 3 min 7 1
3
?pr 2*max .5 1.5
3.
```

LISTADO

```
to min :a :b

if :a<:b [op :a]


op :b

end

to max :a :b
if :a< :b [op :b]
op :a
end
```

COMENTARIOS

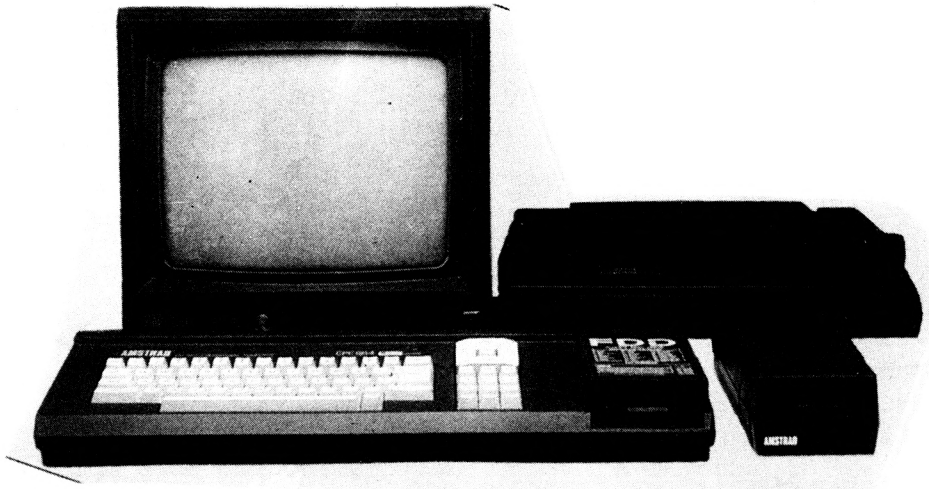
La primera palabra es el nombre del procedimiento. Las otras son para los datos.

La primera parte es el condicional "si :a es menor que :b". Si la condición se cumple se ejecuta la lista encerrada entre corchetes.

La orden "op :a" hace que el procedimiento de como resultado el valor de :a y termine ahí su ejecución.

Como la anterior. Pero sólo una de ellas se ejecuta.

Fin del procedimiento.



COMENTARIOS
POLIG (LOGO
(La estructura repetitiva) (D)

Instrucciones de manejo:

El programa "POLIG" consiste en dos utilidades. Permite dibujar, con la tortuga del Dr. Logo, polígonos regulares. Estos polígonos pueden tener el número de lados y la longitud de lados que se decida al ejecutar los procedimientos.

El programa consiste en dos procedimientos, para dibujar los polígonos hacia la izquierda o hacia la derecha, de la posición de la tortuga. Los nombres respectivamente son "polii" y "polid".

Por ejemplo, para dibujar un polígono regular de tres lados (triángulo) de modo que la tortuga gire hacia la izquierda y de modo que cada lado mida 100 pasos de tortuga, basta ejecutar "polii 100 3". Del mismo modo, "polid 140 8" dibuja un octogono de lado 140 y de forma que la tortuga gira hacia la derecha.

Este tipo de utilidades es bueno almacenarlas en los discos para que podamos utilizarlas en cualquier momento, sin más que cargarlas en la memoria.

LISTADO

```
to polii :lado :número

repeat :número [fd :lado

lt 360/número]
end

to polid :lado :número
repeat :número [fd :lado
  lt 360/ :número]
end
```

COMENTARIOS

El nombre del procedimiento es la primera palabra. Las otras dos palabras son los nombres de las variables locales que utiliza.

Al ejecutar el procedimiento las variables locales adquieren un valor. Se repetirá la lista de instrucciones encerrada entre los corchetes, el número indicado de veces.

LOGO

El programa técnico del mes

Las crecidas en los ríos y arroyos. Cálculo de caudales de aven

INTRODUCCION

EL paso de caudales extraordinariamente grandes, por un río, arroyo o barranco, se denomina avenida, riada o crecida.

Sus efectos negativos son tan conocidos como indeseados, por ello el estudio de los fenómenos que producen las riadas, es de gran interés, así como la evaluación de las mismas, que nos permitirá, el tratar de protegernos de sus efectos.

¿Qué causas provocan las riadas? Generalmente, salvo casos bastante infrecuentes, como la rotura de una presa de embalse o similares, hay dos fenómenos naturales que pueden provocar riadas: las lluvias intensas y la fusión rápida de la nieve. Una combinación de ambos fenómenos también es posible, y de hecho sus efectos se potencian al aparecer simultáneamente.

En nuestra geografía, sin ser despreciables, los efectos de la nieve, son menos importantes que los productos por las lluvias copiosas, dándose con relativa frecuencia lluvias muy intensas en corto espacio de tiempo y en zonas donde el de precipitaciones anuales es muy reducido.

Este artículo girará en torno a la lluvia como origen de las avenidas.

GENERALIDADES SOBRE EL FENOMENO

Las gotas de agua desprendidas por un aguacero, siguen al caer al suelo, trayectorias muy distintas: Una parte se deposita en la parte superior del suelo, dándole humedad, que será absorbida por la vegetación y el resto se perderá por evaporación. Otra parte es retenida directamente por las plantas y otra se depositará formando charcos, que posteriormente se evaporarán.

Todos estos fenómenos y caminos que toma el agua, corresponden a precipitaciones de poca cuantía, pero en aguaceros importantes, a medida que el suelo se va saturando de agua, va dejando escurrir, cada vez, una mayor cantidad de agua caída y simultáneamente se va produciendo una infiltración, en el subsuelo, pasando a engrosar las aguas subterráneas.

Estas aguas subterráneas, acaban desaguando en los cauces, pero en este proceso invierten generalmente un largo tiempo.

A diferencia de las anteriores las aguas superficiales, desaguan con gran rapidez en los cauces, y son evacuadas en un corto espacio de tiempo. Como consecuencia, puede afirmarse que es la escorrentía superficial, el determinante principal de la formación de avenidas, por lo que al menos en una primera aproximación, puede prescindirse de los demás factores.

De la totalidad del agua caída, como hemos visto, sólo una parte da lugar a escorrentía. Este volumen de agua se conoce como "lluvia neta".

Ahora bien, existe un retraso entre la entrada del agua en la cuenca, y su salida por el punto de desagüe, debido al tiempo invertido por las aguas en su trayecto, la medida de este retraso se denomina, "Tiempo de concentración", que no depende de la cantidad de agua, que constituye la escorrentía, sino únicamente de las características físicas de la cuenca, longitud y pendiente del terreno.

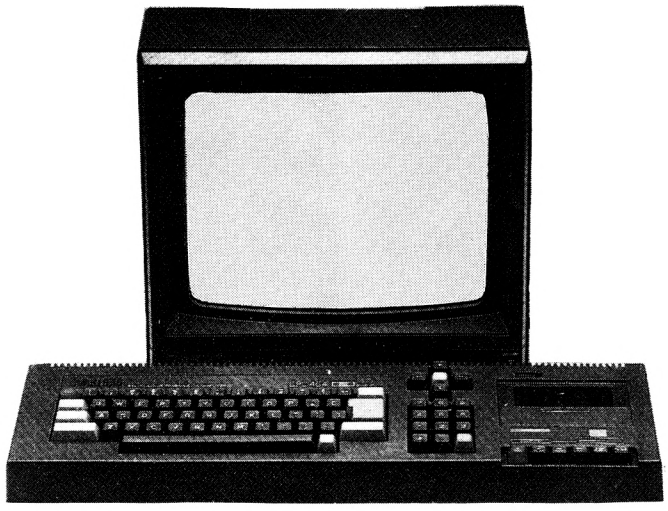
Este y otros conceptos hidrológicos, son necesarios para el análisis de la problemática de la escorrentía, pero no es nuestro interés, el realizar un estudio teórico completo, sino el facilitar un método rápido para obtener un valor, suficientemente aproximado, de la magnitud del caudal máximo previsible de avenida, con una cierta garantía.

METODOLOGIA UTILIZADA EN EL PROGRAMA

El programa que presentamos, responde al criterio expuesto anteriormente, de sencillez, rapidez y suficiente aproximación.

De entre los numerosos estudios existentes, sobre el cálculo de caudales de avenida, el programa está basado en el estudio expuesto en la

da en pequeñas cuencas naturales



VICTOR J. CAMPO LOPEZ

obra "Cálculo Hidrometeorológico de Caudales Máximos en Pequeñas Cuencas Naturales" de don José Ramón Temez, doctor Ingeniero de C. C. y P. (Dirección General de Carreteras M.O.P.U.). Este estudio está basado estructuralmente en el "Método Racional", que reune los requisitos de sencillez y fiabilidad. Los parámetros que intervienen en esta ley general, han sido adaptados al caso de España Peninsular y contrastados experimentalmente.

UTILIDAD DEL PROGRAMA

El programa que presentamos a continuación, permite obtener el caudal máximo previsible en una cuenca, para un período de retorno dado, en función de los parámetros de la cuenca y de la intensidad de lluvia máxima previsible.

El conocimiento de este caudal nos permitirá dimensionar con suficiente garantía, aquellas obras que hayan de realizarse sobre el cauce o proximidades, alcantarillas, pontones, desvios, azudes, tomas de aguas, etc.

El programa, se aplicará únicamente a cuencas, cuya extensión no sea superior a 75 Km^2 y que estén ubicadas, dentro de la Península.

FORMULACION EMPLEADA. Para aquellos que desean profundizar en el programa, exponemos la formulación básica utilizada.

— Tiempo de concentración (Tc)

$$Tc = 0.3 \times \frac{L^{0,76}}{J^{1/4}}$$

L = Longitud del curso principal
J = Pendiente media

— Intensidad-Duración.

$$\frac{I}{Id} = \frac{Il}{Id} \frac{28^{0.1} - D^{0.1}}{0.4} \frac{Il}{Id}$$

= Relación intensidad horaria/diaria

D = Tiempo

— Precipitación máxima diaria (Pd).

Obtenida a partir de los mapas de Isoyetas (ver ejemplo).

— Coeficiente de escorrentía.

$$C = \frac{(Pd-Po)(Pd+23\ Po)}{(Pd+11\ Po)^2} \quad Po = \text{Parámetro propio de la cuenca, ver su determinación en "Uso del pgm"}$$

— Caudal Máximo

C.I.A.

$$Q = \frac{C.I.A.}{3}$$

Fórmula Racional

A = Area de la cuenca I = Intensidad

GLOSARIO DE TERMINOS UTILIZADOS

A continuación exponemos una relación de palabras, utilizadas en este artículo, no con el fin de ser tenidas en cuenta como definición, sino aclaratorias, para aquellas personas con pocos conocimientos de Hidrología.

— Caudal: Cantidad de agua que fluye por una sección, en la unidad de tiempo.

— Lluvia neta: Parte de la lluvia caída, que produce la escorrentía.

— Escorrentía: Agua que fluye sobre la superficie del terreno o bajo él. (Esc. superficial o subterránea).

— Cuenca: Area receptora de aguas y conductora de las mismas, a un cauce común.

— Cauce: Parte más profunda de la cuenca, por donde fluyen las aguas.

— Tiempo de concentración: Medida del efecto de retraso en la escorrentía, debido al trayecto que debe recorrer el agua, desde los puntos de caída hasta el punto de concentración considerado.

— Parámetro Po: Se produce escorrentía, cuando la suma de la lluvia acumulada desde el comienzo del aguacero es igual a Po.

— Avenida, riada, crecida: Paso por un cauce, de caudales extraordinariamente grandes.

— Divisoria: Línea límite que separa cuencas adyacentes. Se obtiene uniendo los puntos más altos, que delimitan la cuenca.

USO DEL PROGRAMA

* ENTRADA DE DATOS

Introduzca los valores correspondientes a los INPUTS, que se relacionan a continuación, en orden de aparición en pantalla.

— Nombre de la cuenca: Nombre del arroyo, río o barranco. Si no se conoce pulsar ENTER.

— Superficie de la cuenca en Km^2: Si no se conoce a priori, puede obtenerse planimetrando la cuenca, en la cartografía disponible. Tenga en cuenta que el pgm. sólo admite superficies menores de 75 Km^2, por tanto si rebasa esta cifra, se producirá un mensaje de aviso y el pgm. volverá al principio.

— Longitud máxima del cauce Km: Esta longitud corresponde al cauce principal (más largo). Si se desconoce puede obtenerse midiendo sobre la misma cartografía en que se determinó el área.

— Pendiente media: en m/m: Es la pendiente media del cauce principal. Puede obtenerse dividiendo la diferencia entre la cota más alta del cauce y la más baja, por la longitud. Si el cauce es muy irregular, puede dividirse por tramos y calcular la pendiente media ponderada de dichos tramos.

— Il/Id: Esta relación puede obtenerla del gráfico que el ordenador le presenta en pantalla, interpolando entre las curvas del mismo según la situación geográfica de la cuenca.

— Máxima precipitación en 24 horas: Puede obtenerse (si se desconoce) con suficiente aproximación de los planos de Isoyetas, publicados por la Dirección General de Carreteras (M.O.P.U.), para el período de retorno considerado. (Puede consultarse La Instrucción de Carreteras Precipitaciones max. anuales en 24 horas).

— Período de retorno: Introduzca el valor en años, de acuerdo con la max. precipitación considerada.

— Valor medio de Po: La estimación de Po (parámetro de umbral de escorrentía), se realiza de acuerdo con las características del terreno y tipo de cultivo, mediante las tablas que exponemos a continuación.

TABLA PARA LA ESTIMACION INICIAL DEL PARAMETRO Po

Uso de la tierra	Pendiente %	Características hidrológicos	Grupo de suelo A	B	C	D
Barbecho	⩾ 3	R	15	9	6	4
		N	17	11	9	6
	< 3	R/N	20	14	11	8
Cultivos en hilera	⩾ 3	R	23	13	8	6
		N	25	16	11	8
	< 3	R/N	28	19	14	11
Cereales de Invierno	⩾ 3	R	29	17	10	8
		N	32	19	12	10
	< 3	R/N	34	21	14	12
Relación de cultivos pobres	⩾ 3	R	26	15	9	6
		N	28	17	11	8
	< 3	R/N	30	19	13	10
Relación de cultivos densos	⩾ 3	R	37	20	12	9
		N	42	23	14	11
	< 3	R/N	47	25	16	13
Praderas	⩾ 3	Pobre	24	14	8	6
		Medio	53	23	14	9
		Buena	—	33	10	13
		Muy buena	—	41	22	15
	< 3	Pobre	58	23	12	7
		Media	—	35	17	10
		Buena	—	—	22	14
		Muy buena	—	—	22	14
		Muy buena	—	—	25	13
Plantaciones regulares de aprovechamiento forestal	⩾ 3	Pobre	62	26	15	10
		Media	—	34	19	14
		Buena	—	42	22	15
	< 3	Pobre	—	34	19	14
		Media	—	42	22	15
		Buena	—	30	25	16
Mares forestales (bosques, monte bajo, etc.)		Muy claro	40	17	8	5
		Clara	60	24	14	10
		Media	—	34	22	16
		Espesa	—	47	31	23
		Muy espesa	—	65	43	33
Rocas permeables	⩾ 3			3		
	< 3			5		
Rocas impermeables	⩾ 3			2		
	< 3					

LABORES DE CULTIVO

En línea recta (símbolo R).

Cuando el laboreo del suelo, la siembra y las labores de cultivo se realizan en la dirección de la máxima pendiente o a media ladera.

En líneas de nivel (símbolo N).

Cuando el laboreo del suelo, la siembra y las labores de cultivo se realizan siguiendo las curvas de nivel del terreno. Evidentemente en terrenos llanos no resulta fácil, ni tiene mucho sentido, matizar las líneas de nivel, por lo que no se hace diferencia entre el laboreo en línea recta (R) y el laboreo en línea de nivel (N).

SUELOS

Grupo A

En ellos el agua se infiltra rápidamente aún cuando estén muy húmedos. Profundos y de texturas gruesas (arenosas o areno-limosas), están excesivamente drenados.

Grupo B

Cuando están muy húmedos tienen una capacidad de infiltración moderada. La profundidad de suelo es de media a profunda, y su textura franco-arenosa, franca, franco-arcillo-arenosa o franco-limosa según terminología del U. S. Departament of Agriculture. Están bien o moderadamente drenados.

Grupo C

Cuando están muy húmedos la infiltración es lenta. La profundidad de suelo es inferior a la media y su textura es franco-arcillosa, franco-arcillo-limosa, limosa o arcillo-arenosa. Son suelos imperfectamente drenados.

Grupo D

Cuando están muy húmedos la infiltración es muy lenta. Tienen horizontes de arcilla en la superficie o próximos a ella y están pobremente o muy pobremente drenados. También se incluyen aquí los terrenos con nivel freático permanentemente alto y suelos de poco espesor (litosuelos).

Si la cuenca es bastante uniforme, introduzca el valor obtenido de la tabla. Si no lo es (diversidad de cultivos, suelos etc.) divida la cuenca en zonas homogéneas, halle la superficie de cada zona, calcule con la tabla el Po correspondiente y tome para el pgm. el valor medio ponderado de Po.

— Multiplicador regional: Este parámetro trta de corregir el valor de Po, de acuerdo con el grado de humedad existente, antes de comenzar el aguacero y se obtiene directamente interpolando en el gráfico que se ofrece por pantalla, de acuerdo con la situación geográfica de la cuenca dentro de la Península.

* SALIDA DE RESULTADOS

El pgm. presenta un cuadro con todos los valores físicos de la cuenca y parámetros introducidos anteriormente y el valor del caudal máximo previsible de avenida, para el período de retorno considerdo.

EJEMPLO DE APLICACION: Sobre el barranco de los Regueros, en Fresno de Torote, Madrid, se ha de ubicar una obra de paso, para un camino vecinal, en el punto que se indica en el plano.

Se ha delimitado la cuenca trazando la línea divisoria, hasta el punto de ubicación de la obra.

Se desea conocer el caudal máximo previsible, para un período de retorno de 10 años, con el fin de dimensionar la alcantarilla de paso.

Planimetrando la cuenca, se obtuvo una superficie de 2.18 Km^2. La longitud del cauce principal es de 2.35 Km.

La pendiente media será:

cota más alta 792
cota más baja 730
desnivel 62 m.

$$J = \frac{\text{desnivel}}{\text{longitud}} = \frac{62}{2350} = 0.026$$

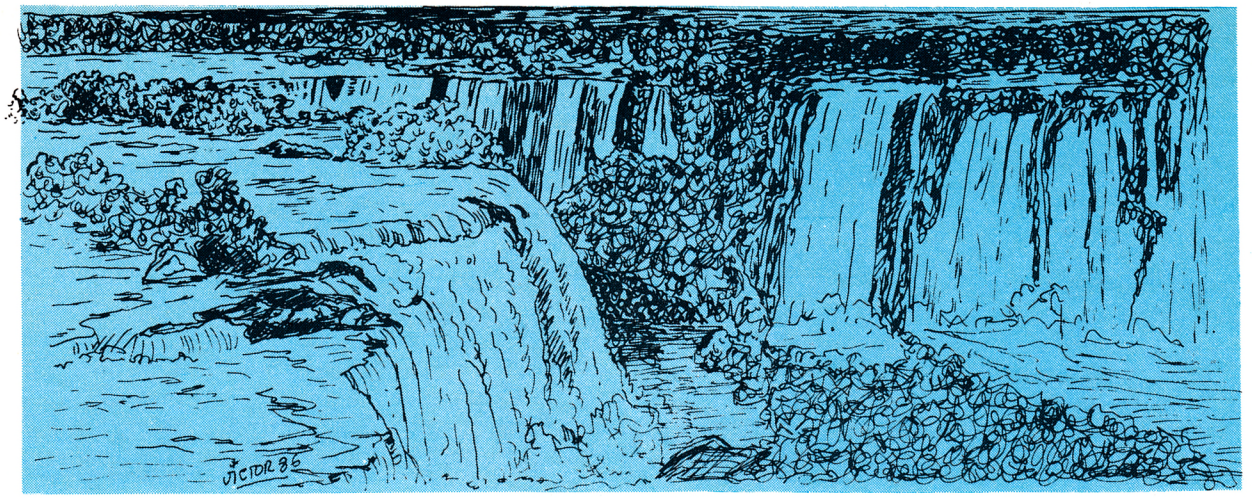
Del gráfico de pantalla, obtenemos un valor de Il/Id = 9, de acuerdo con la zona de ubicación de la cuenca.

La precipitación máxima en 24 horas. Se ha obtenido del plano de "Precipitaciones máximas anuales en 24 horas", para PR = 10 años, y es 90 mm. El período de retorno considerado es 10 años.

El valor medio de la Po se ha obtenido dividiendo la cuenca en dos zonas, una de monte bajo que representa el 70% de la superficie y la otra con rocas impermeables.

	Po
Zona 1) Monte bajo, muy claro, con suelo tipo C	5
Zona 2) Suelo formado por rocas impermeables	8

$$\text{Po medio} = \frac{5 \times 30 + 8 \times 70}{100} = 7.1$$



El multiplicador regional obtenido interpolando en el gráfico de pantalla, es 2.8.

Una vez introducido todos los datos en el programa, se obtiene como caudal máximo previsible 8.92 $m^3/s.$, para el período de retorno de 10 años considerado.

```
10 REM ..............
20 REM
30 REM. CALCULO DE CAUDALES DE .
40 REM. AVENIDA EN PEQUENNAS .
50 REM. CUENCAS NATURALES .
60 REM.
70 REM ..............
71 MODE 0: PRINT CHR$(24): PRINT "     ";
   "CAUDALES DE AVENIDA";
72 PRINT "     "; "EN CUENCAS NATU-
   RALES";
73 PRINT "_______" : PRINT CHR$(24)
74 LOCATE 7,15:PRINT "AMSTRAD":LOCA-
   TE 15,16: PRINT "Club"
75 FOR K=1 TO 7000:NEXT K
80 MODE 1: LOCATE 10,5: PRINT "DATOS
   DE LA CUENCA":PRINT TAB(9);"_____.":
   PRINT
90 INPUT "NOMBRE DE LA CUENCA";C$
100 INPUT "Superficie cuenca (Km2)";A
110 GOSUB 520
120 INPUT "Longitud max. cauce (Km) ";]
130 INPUT "Pendiente media (m/m)     ";j
140 CLS : GOSUB 570
150 GOSUB 820
160 LOCATE 1,23
170 INPUT "I1/ID ";I1
180 TC=0.3*(L/J 0.25) 0.76
190 IID=I1 ((28 0.1-TC 0.1)/0.4)
200 INPUT "MAX. PRECIPITACION (mm/
    24h.) ";PD
205 INPUT "PERIODO DE RETORNO";PR
210 ID=PD/24: I=IID*ID
220 INPUT "VALOR MEDIO DE PO";PO
230 CLS: GOSUB 570
240 GOSUB 2040
249 LOCATE 1,21
250 INPUT "MULTIPLICADOR REGIONAL"
    ;MR
260 PO=PO*MR
270 C=(PD-PO)*(PD+23*PO)/(PD+11*PO) 2
280 Q=C*I*A/3 : REM fórmula racional
290 f2$= "PtPtPt.PtPt":F5$="PtPtPt.PtPtPtPtPt"
300 MODE 1
310 WINDOW Pt1,1,17,5,15
320 WINDOW Pt2,21,40,5,15
330 PRINT Pt1," S="; USING F2$;A;: PRINTPt1,
    "Km2"
340 PRINT Pt2,"J ="; USING F5$;J;: PRINTPt2,
    "m/m"
350 PRINT Pt1, " L="; USING F2$;L;: PRINT
    Pt1, "Km"
360 PRINT Pt2, "I/="; USING F2$;I1
370 PRINT Pt1, " TC="; USING F2$;TC;: PRINT
    Pt1, "A."
380 PRINT Pt2,"MR="; USING F2$;MR
390 PRINT Pt1, "Po="; USING F2$;PO/MR;:
    PRINTPt1," mm."
400 PRINT Pt2,"PD="; USING F2$;PD;: PRINT
    Pt2, "m/m"
410 PRINT TAB(10); "CUENCA - ";C$
420 PRINT TAB(5); "PERIODO DE RETORNO
    ";PR;" A."
425 PRINT CHR$(24)
430 LOCATE 9,12: PRINT "CAUDAL =";USING
    "PtPtPtPtPt.PtPt";Q;
431 PRINT "M3/s."
440 LOCATE 1,22
450 PLOT 0,350:DRAW 600,350
460 PLOT 0,250:DRAW 600,250
470 PLOT 0,185:DRAW 600,185
480 PLOT 0,185:DRAW 0,350
490 PLOT 295,250:DRAW 295,350
500 PLOT 600,185:DRAW 600,350
505 PRINT CHR$(24)
510 END: REM FIN DE PROGRAMA ======
    ==========================
    ===
520 IF A <=75 THEN RETURN
530 LOCATE 1,16
540 PRINT "EL METODO SOLO ES VALIDO
    PARA CUENCAS MENORES DE 75 Km2
550 FOR K=1 TO 6000:NEXT K
560 GOTO 80
570 REM ..... DIBUJO MAPA PE-
    NINSULA .....
580 MODE 2
610 RESTORE 730
```

Próximo programa del mes: "SISTEMAS DE UNIDADES".

AMSTRAD
CPC 464, 664 y 6128
PROGRAMACION ESTRUCTURADA

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

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

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

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

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

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

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

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

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

P.V.P. 1.700 ptas.

NUESTRO PRÓXIMO NÚMERO

AMSTRAD EDUCATIVO

N.º 2

El AMSTRAD y el CPM (2.ª parte)

Ficheros en el AMSTRAD

Basic del AMSTRAD

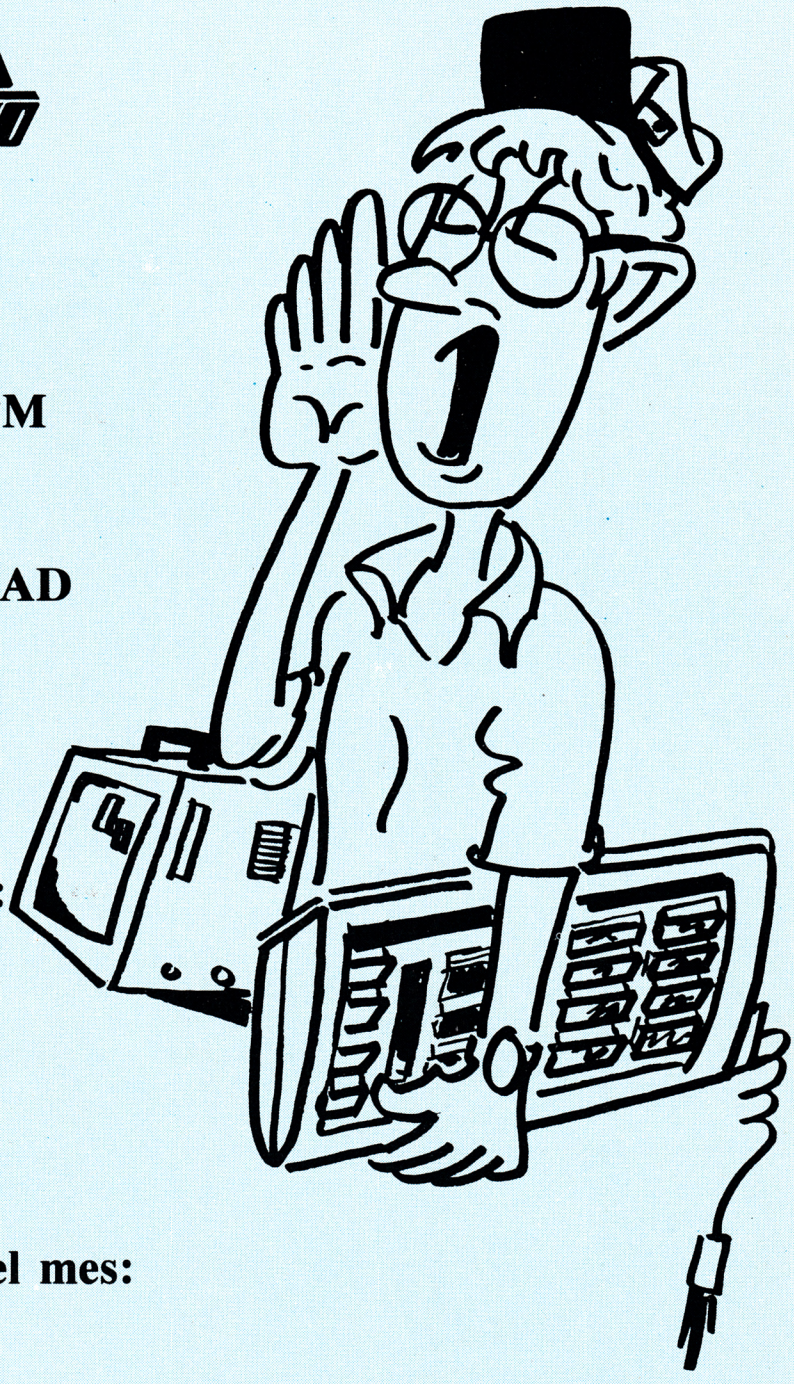
Fichas del AMSTRAD

Programando en Basic: Las 21 cerillas

Logo del AMSTRAD

Doctor Logo

El programa técnico del mes: Sistemas de unidades.



LA "BIBLIA" DEL

¡ya está a la venta!

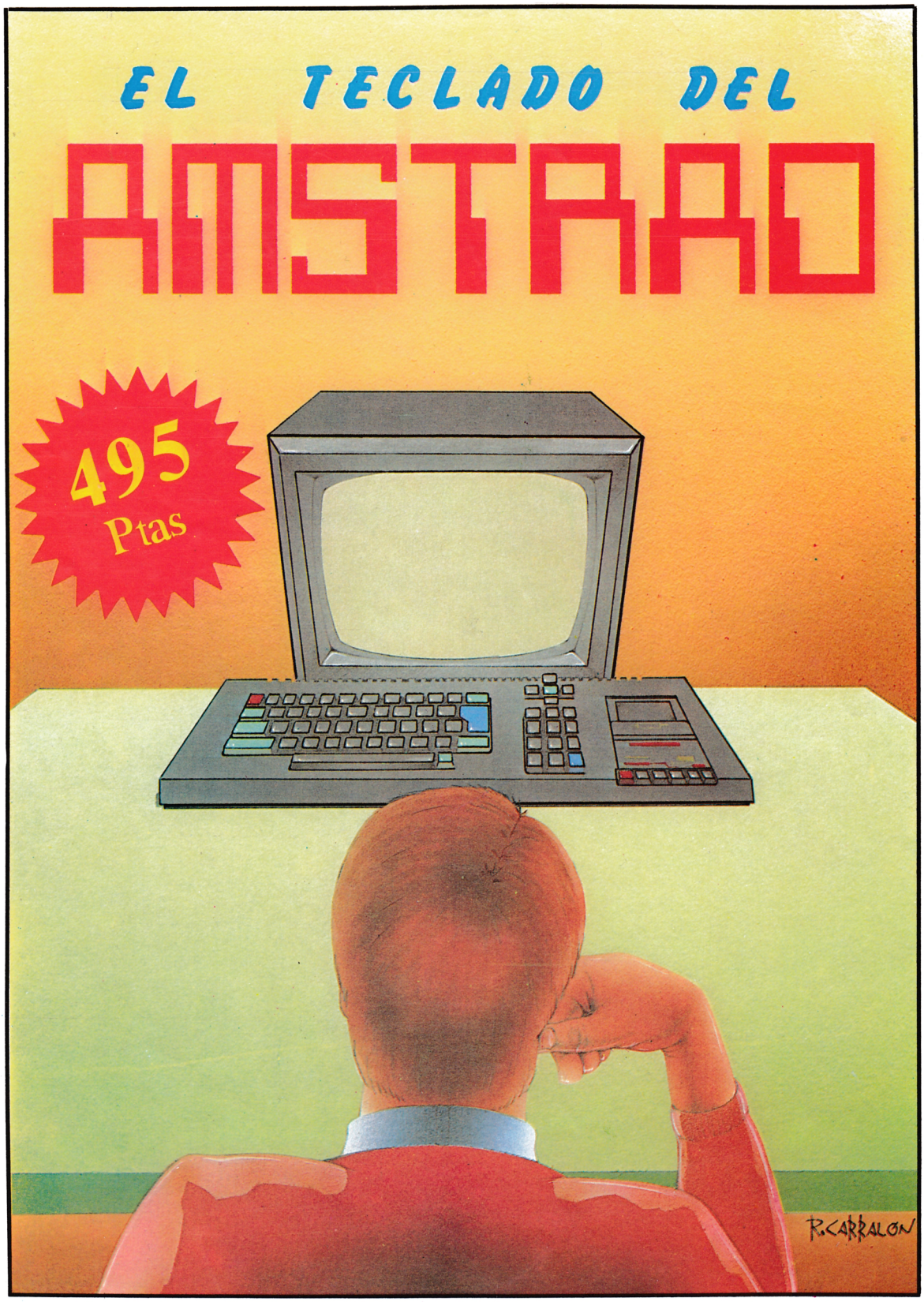
AMSTRAD

FASCICULO 2

295 PTAS.

DOMINARLO ES UN JUEGO

¡ya está a la venta!



AMSTRAD EDUCATIVO N.º 1

Fe de Erratas

Pág. 27:
— Donde pone

LISTADO

```
to polii :lado : número

repeat: número (fd :lado

lt 360/ número)
end

to polid: lado :número
repeat :número (fd: lado
lt 360/ :número)
end
```

— Debe poner

LISTADO

```
to polii : lado :número

repeat :número (fd : lado

Lt 360/:número)
end

to polid :lado :número
repeat : número (fd: lado
rt 360/ :número)
end
```

Página 29:
— Donde pone:
— Tiempo de concentración (Tc)

$$Tc = 0.3 \times \frac{L^{0,76}}{J^{1/4}}$$

L = Longitud del curso principal
J = Pendiente media

— Intensidad - Duración

— Debe poner:
— Tiempo de concentración (Tc)

$$Tc = 0.3 \left(\frac{L}{J^{1/4}}\right)^{0.76}$$

L = Longitud del curso principal
J = Pendiente media

— Intensidad - Duración

LISTADO DEL PROGRAMA TECNICO DEL MES

Las crecidas en los ríos y arroyos. Cálculo de caudales de avenida en pequeñas cuencas naturales.

Nota.—El símbolo Pt se debe sustituir por #

```
                10 REM ..............................
20 REM .                                            .
30 REM .   CALCULO DE CAUDALES DE                   .
40 REM .   AVENIDA EN PEQUENNAS                     .
50 REM .   CUENCAS NATURALES                        .
60 REM .                                            .
70 REM ..............................
71 MODE 0: PRINT CHR$(24); PRINT "                                    " ;"CAUDALES DE AVENIDA ";
72  PRINT  "                                  "; "EN CUENCAS NATURALES";
73  PRINT "____________________" : PRINT  CHR$(24)
```

```
74 LOCATE 7,15: PRINT  "AMSTRAD":LOCATE 15,16:  PRINT  "Club"
75  FOR K=1 TO 7000:NEXT K
80 MODE 1: LOCATE 10,5: PRINT "DATOS DE LA CUENCA":PRINT TAB(9);"--------------------": PRINT
90 INPUT  "NOMBRE DE LA CUENCA ";C$
100  INPUT  "Superficie  cuenca (Km2) ";A
110  GOSUB 520
120  INPUT  "Longitud max. cauce (Km) ";L
130  INPUT  "Pendiente media (m/m)    ";J
140 CLS :  GOSUB 570
150  GOSUB 820
160 LOCATE 1,23
170  INPUT  "I1/ID ";I1
180 TC=0.3*(L/J^0.25)^0.76
190 ID=I1*((28^0.1-TC^0.1)/0.4)
200 INPUT "MAX.PRECIPITACION (mm/24h.) ";PD
205 INPUT  "PERIODO DE RETORNO";PR
210 ID=PD/24: I=ID*ID
220 INPUT  "VALOR MEDIO DE P0 ";P0
230 CLS:  GOSUB 570
240  GOSUB 2040
249 LOCATE 1,21
250 INPUT  "MULTIPLICADOR REGIONAL ";MR
260 P0=P0*MR
270 C=(PD-P0)*(PD+23*P0)/(PD+11*P0)^2
280 Q=C*I*A/3 ; REM formula racional
290 F2$= "###.##";F5$="###.#####"
300 MODE 1
310 WINDOW #1,1,17,5,15
320 WINDOW #2,21,40,5,15
330 PRINT #1," S ="; USING F2$;A;: PRINT#1," Km2"
340 PRINT #2,"J ="; USING F5$;J;: PRINT#2, " m/m"
350 PRINT #1," L ="; USING F2$;L;: PRINT#1," Km"
360 PRINT #2,"I/="; USING F2$;I1
370 PRINT #1," TC="; USING F2$;TC;: PRINT#1," H."
380 PRINT #2,"MR="; USING F2$;MR
390 PRINT #1," Po="; USING F2$;P0/MR;: PRINT#1," mm."
400 PRINT #2,"PD="; USING F2$;PD;: PRINT#2, " m/m"
410  PRINT TAB(10); "CUENCA -  ";C$
420  PRINT TAB(5); "PERIODO DE RETORNO ";PR;" A."
425 PRINT  CHR$(24)
430 LOCATE 9,12: PRINT "CAUDAL =";USING "#####.##";Q;
431 PRINT  " M3/s."
440 LOCATE 1,22
450 PLOT 0,350:DRAW 600,350
460 PLOT 0,250:DRAW 600,250
470 PLOT 0,185:DRAW 600,185
480 PLOT 0,185:DRAW 0,350
490 PLOT 295,250:DRAW 295,350
500 PLOT 600,185:DRAW 600,350
505 PRINT  CHR$(24)
510 END: REM F I N  D E  P R O G R A M A=========================================
520 IF A<=75 THEN RETURN
530  LOCATE 1,16
540 PRINT  "EL METODO SOLO ES VALIDO PARA CUENCAS   MENORES DE 75 Km2
550 FOR K=1 TO 6000:NEXT K
560 GOTO 80
```

```
570 REM ..... DIBUJO MAPA PENINSULA .....
580 MODE 2
610 RESTORE 730
620  READ X,Y:X=X*2.3: Y=Y*2.3: PLOT X,Y
630 FOR K=1 TO 48: READ X,Y:X=X*2.3: Y=Y*2.3: DRAW X,Y
640 NEXT K
650 RESTORE 780
660  READ X,Y:X=X*2.3: Y=Y*2.3: PLOT X,Y
670 FOR K=1 TO 7: READ X,Y:X=X*2.3: Y=Y*2.3: DRAW X,Y
680 NEXT K
690 RESTORE 790
700  READ X,Y:X=X*2.3: Y=Y*2.3: PLOT X,Y
710 FOR K=1 TO 4: READ X,Y:X=X*2.3: Y=Y*2.3: DRAW X,Y
720 NEXT K
730 DATA 190,168,186,160,163,161,152,159,136,163,134,165,130,163,108,169,102,165,106,161
740 DATA 93,161,89,157,90,152,94,153,92,147,97,145,94,139,94,119,89,99,82,87
750 DATA 82,80,88,84,86,77,90,77,90,65,86,51,99,51,103,49,110,53,119,52
760 DATA 133,32,139,32,141,38,150,40,152,44,184,44,196,59,206,61,204,64,208,74
770 DATA 217,80,212,82,208,92,222,113,222,118,252,137,250,147,248,151,248,156
780 DATA 94,139,104,139,109,135,124,136,129,128,121,121,112,74,110,53
790 DATA 186,160,216,149,221,153,238,146,250,147
800 DATA 108,142,127,159,159,159,171,157,178,142,182,158,187,150,218,130,238,130,150,130,150,112,158,98,109,54,130,55,166,
      50,198,66, 206,92
810 RETURN
820 REM ..... ISOLINEAS 11/ID .....
825 LOCATE 1,7:  PRINT CHR$(24)+ "----------------------"
826 LOCATE 1,8:  PRINT  "ñ MAPA DE ISOLINEAS ñ"
827 LOCATE 1,9:  PRINT "----------------------"
828  PRINT CHR$(24)
830 RESTORE 850: z=3:  GOSUB 2000
840 LOCATE  44,1: PRINT  8
850 DATA 152,165,160,157,188,159,192,166
860 z=2:  GOSUB 2000
870 DATA 110,170,119,153,208,155
880 LOCATE  32,1: PRINT  9
890 z=3:  GOSUB 2000
900 DATA 136,145,148,148,163,135,164,129,153,117,136,110,128,114,125,121,130,138,136,145
910 LOCATE  36,5: PRINT  10
920 z=7:  GOSUB 2000
930 DATA 142,142,157,135,154,124,138,113,132,116,131,127,136,139,142,142
940 LOCATE  38,7: PRINT  11
950 z=10:  GOSUB 2000
960 DATA 163,115,171,115,172,111,168,100,169,88,165,80,160,77,155,83,158,93,151,105,163,115
970 LOCATE  47,14: PRINT  "9"
971 LOCATE  48,10: PRINT  "H"
980 z=15:  GOSUB 2000
995 DATA 142,36,145,49,151,50,162,47,180,58,178,105,187,115,176,123,182,132,169,142,168,146,172,150,187,151,212,145,235,
      145,251,151
996 LOCATE 40,22: PRINT  "10"
1000 z=9:  GOSUB 2000
1010 DATA 190,48,198,78,195,96,202,115,191,125,187,134,172,145,177,148,216,134,254,145
1020 LOCATE 55,20: PRINT  "11"
1030 z=6:  GOSUB 2000
1040 DATA 212,72,202,100,206,115,204,128,219,130,234,135,252,133
1050 LOCATE 62,16: PRINT  "12"
1500 RETURN: REM Fin rutina Isolineas
```

```
2000 REM ....  RUTINA DIBUJO  .....
2010  READ X,Y:X=X*2.3: Y=Y*2.3: PLOT X,Y
2020 FOR K=1 TO z: READ X,Y:X=X*2.3: Y=Y*2.3: DRAW X,Y: NEXT K
2030 RETURN :REM Fin rutina dibujo
2040 REM ..................................... MULTIPLICADOR REGIONAL ......
2045 RESTORE 2060
2050 z=5:  GOSUB 2000
2060 DATA 106,135,120,141,132,156,148,158,186,156,196,160
2070 z=6:  GOSUB 2000
2080 DATA 112,88,128,88,168,98,190,110,201,135,232,137,254,133
2085 Z=5: GOSUB 2000
2090 DATA 140,36,144,47,180,52,184,72,192,80,218,84
2100 LOCATE 34,4 : PRINT  "2"
2110 LOCATE  46,11: PRINT  "3"
2120 LOCATE  48,10: PRINT  "M"
2130 LOCATE 53,14 : PRINT  "4"
2140 LOCATE 9,8: PRINT  CHR$(24)+" MAPA "
2150 LOCATE 1,9: PRINT  " MULTIPLICADOR REGIONAL ": PRINT  CHR$(24)
2170 RETURN
```