AMSTRAD EDUCATIVO

EL PROGRAMA TECNICO DEL MES:
SISTEMAS DE UNIDADES

EL AMSTRAD Y EL CPM

BASIC DEL AMSTRAD

Doctor Logo

295 Ptas.

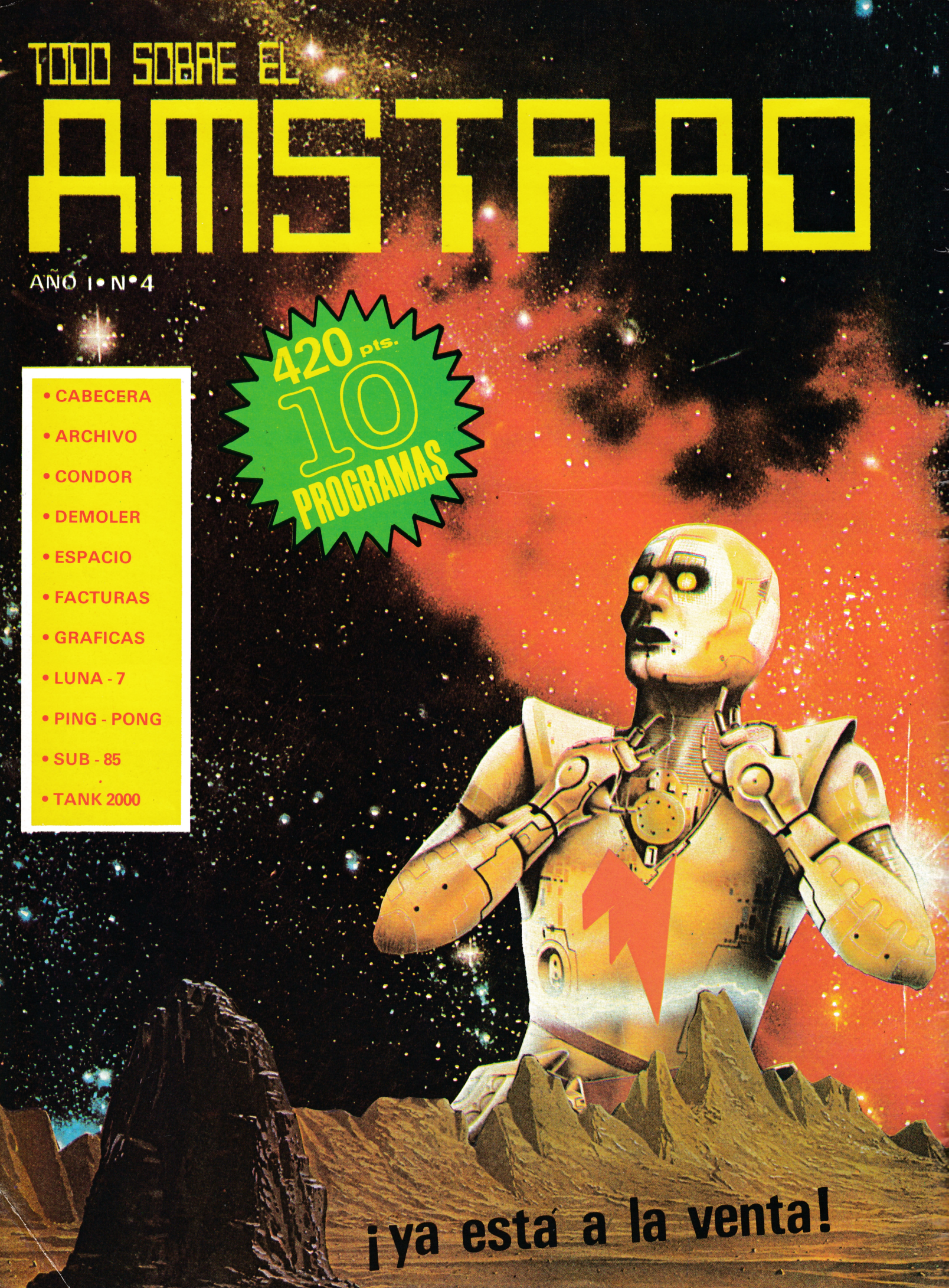
Nº2

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

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

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

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

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

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

Esperamos vuestras opiniones.

SUMARIO

Edita: Grupo Editorial
G.T.S., S. A.
Bailén, 20-1.º Izda.
28005 Madrid
Telf.: 266 6601-02.
Director: Antonio Bellido.
Colaboran en este número:
Juan M. Pintor, Víctor J. Campo López.
Maquetación: 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

Ejercicios con estructura CP/M

1.° —¿Qué subgrupo de programas CP/M se encarga del directorio del disco en uso?

.....

Respuesta.— BDOS.

2.° —¿Qué proceso sigue el BDOS para dar entrada a cualquier fichero? .....

Respuesta 1. Comprueba en la pista catálogo la no existencia de otro fichero con el mismo nombre.

2. Comprueba si la capacidad libre del disco es suficiente para el nuevo fichero.

3. Superadas las dos comprobaciones anteriores, da entrada en la pista catálogo a los datos correspondientes al fichero.

3.° —Para transferir información desde una unidad de disco a una impresora:

¿Qué sección del DOS se encarga de comprobar si la impresora está ocupada?

.....

Respuesta.— BIOS.

¿Qué sección del DOS se encarga de localizar la información en el disco?

Respuesta.— BDOS

¿Qué sección del DOS se encarga de que la transferencia se efectúe con eficacia? .....

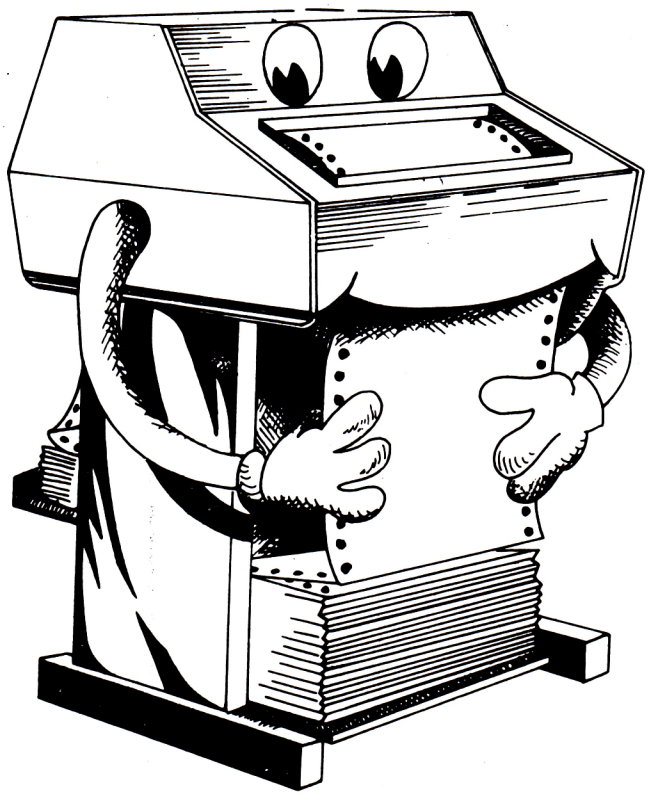
Respuesta.— BIOS.

4.° —¿Cuál es el acrónimo de la sección de CP/M encargada de procesar las órdenes del operador? ¿Qué significa en español? .....

Respuesta.— CCP. Procesador de órdenes de consola.

5.° —¿Cómo se llaman las órdenes CP/M contenidas en el CCP? .....

Respuesta.— Permanentes o residentes.



Edición de línea

Una de las acepciones del Diccionario para la palabra edición es "impresión o estampación de una obra o escrito para su publicación".

Aplicando esta idea al terreno sobre el que nos estamos moviendo, debemos interpretar el concepto de "edición de línea" como la acción que debe llevar a cabo el operador para imprimir en pantalla la orden a ejecutar por el DOS.

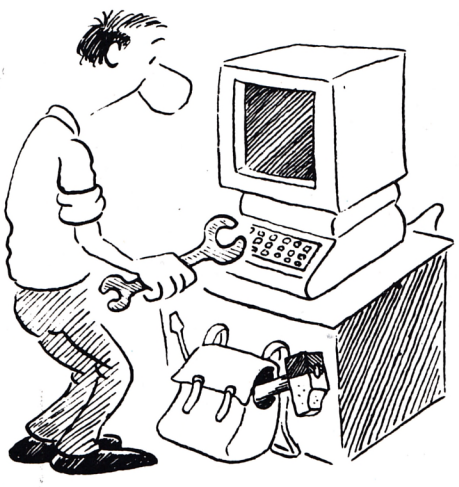
Esta acción pasa por escribir la orden correctamente en el teclado, y finaliza en el momento de pulsar la tecla de retorno de carro normalmente conocida por ENTER o RETURN. A esta tecla, y el efecto subsecuente que se produce al pulsarla, nos referiremos mediante <cr>.

Las órdenes en CP/M, dadas a través del teclado, pueden ocupar una línea de pantalla o más, pero, en todo caso, no pueden exceder de un número predeterminado de caracteres. Al conjunto de caracteres que configuran la orden se le da el nombre de "línea de orden", las líneas de orden dependiendo del ordenador, suelen permitir hasta un máximo de caracteres, comprendidos entre 127 y 255. Por tanto, debe consultarse el manual en este sentido.

Con toda seguridad, el operador, en algún momento, cometerá errores durante el tecleado o escritura de las líneas de orden y, por tanto, deberá conocer los mecanismos que CP/M pone a su disposición para corregirlos.

A estos mecanismos se les conoce con el nombre de **"órdenes de edición de línea".** Resumen de estas órdenes:

E.— Cambia cursor al comienzo de la línea siguiente.
H.— Borra último carácter.
J.— Da por concluida la línea actual.
M.— Retorno de carro.
R.— Repite línea en edición.
S.— "Congela" pantalla.
U.— Cancela línea de orden actual.
C.— Reinicializa el sistema.
P.— Conmuta impresora.
Z.— Da por terminado el proceso de entradas actual.



Ordenes de edición de línea

El usuario puede modificar una línea de orden en cualquier momento antes de pulsar <cr>.

CP/M permite estas correcciones mediante ciertas órdenes de edición de línea, los cuales suelen representarse con una letra y con el signo asu izquierda. PC: E.

La acción que lleva a efecto cada orden de edición de línea se obtiene pulsando simultáneamente la tecla de control y la de la letra del comando correspondiente. En algunas máquinas la tecla ALT sustituye a la de control.

La función que lleva a efecto cada orden está controlada por el CCP.

Veamos, a continuación, un resumen:

E

Forma de obtenerlo: Pulsando simultáneamente CONTROL (o ALT según el ordenador) y la tecla de la letra E.

Función asignada: Producir un final físico en la línea que actualmente se está escribiendo. Al pulsar E, el cursor salta al comienzo de la línea inmediatamente inferior, pero la orden no pasa a la CPU para su ejecución. Util para línea de orden mayores que una línea de caracteres en pantalla. En todo caso, la línea de orden no se da por concluida hasta que no se pulsa <cr>.

Ejemplo: Supongamos que se desea dar la orden DIR, lo cual exigiría teclear D, I y R, y pulsar <c r>, apareciendo en pantalla

A > DIR

y a continuación, como veremos al estudiar esta orden, una relación de los ficheros que contiene el disco que esté instalado en la correspondiente unidad. Si con el fin de comprender las implicaciones de E escribiéramos, tras A>, cada una de las letras de DIR seguidas de E, obtendríamos la siguiente imagen

A > D
I
R

En estas condiciones, al pulsar <c r>, el resultado sería el mismo que en el supuesto primero.

De todo esto podemos inferir que E no afecta a la orden que se esté escribiendo, y sí, exclusivamente, a la forma en que se configura la misma en pantalla.

H

Función asignada: Mover el cursor atrás una posición de carácter borrando. Esta función se puede usar en tanto no se pulse <c r>.

Formas de obtenerlo: Pulsando simultáneamente CONTROL y la tecla de la letra H. También apretando BACKSPACE, ROBUOT o DELETE si el ordenador dispone en su teclado de alguna de estas teclas.

Ejemplo: Si al teclear la orden DIR escribimos:

A > D I T < ■

y antes de dar por concluida la orden con <c r>, se utiliza H, por alguno de los procedimientos explicados, la T desaparece bajo el cursor (■) pudiendo pulsar, ahora, la R para conseguir la orden correcta.

J

Formas de obtenerlo: Pulsando simultáneamente CONTROL y la tecla de la letra J o apretando LINE FEED (LF) si el teclado tiene esta función.

Función asignada: Dar por concluida la línea de orden y pasa a la siguiente: Equivale a <cr>. Su denominación es *Line feed* o alimentación de línea.

Ejemplo: La orden dada en los dos ejemplos de los comandos anteriores, se han concluido con un <c r>. Si cambiamos el <c r> por J el efecto sería el mismo.

M

Forma de obtenerlo: Pulsando simultáneamente CONTROL y la tecla de la letra M o apretando <c r>.

Función asignada: Su denominación es *carriage return* o retorno de carro: obliga al cursor a volver al comienzo de líneas (margen izquierdo).

Ejemplo: Vale lo dicho en el anterior comando.

R

Forma de obtenerlo: Pulsando simultáneamente CONTROL y la tecla de la letra R.

Función asignada: Volver a imprimir en pantalla la actual línea de orden, siempre, claro está, que no se haya concluido con <c r>. La misión encomendada a R no tiene mayor validez en CP/M 2.2 y 86, y se justifica en versiones anteriores ya que al borrar caracteres producían una doble impresión. Cuando se aplica R, la línea en pantalla se cancela con el símbolo #.

Ejemplo: Supongamos que, a título de curiosidad, tecleamos

A > DIR

y, sin pulsar <cr>, damos un R, la pantalla nos mostraría

A > DIR #
DIR

Con lo cual se consigue repetir lo escrito. Nada más.

S

Forma de obtenerlo: Pulsando simultáneamente CONTROL y la tecla de la letra S.

Función asignada: Detener la impresión en pantalla de la secuencia de salida que, a la sazón, se esté produciendo.

Pulsando cualquier tecla el proceso continúa a partir del carácter donde se produjo la interrupción. Si una impresora está activada, con P como veremos, ésta responde de la misma forma.

Ejemplo: Tras una orden del tipo DIR, ya utilizada en otros ejemplos, la pantalla comenzará a mostrar los ficheros que actualmente tiene un determinado disco. En estas condiciones, si ordenamos S el listado se detendrá indefinidamente hasta que se pulse una tecla.

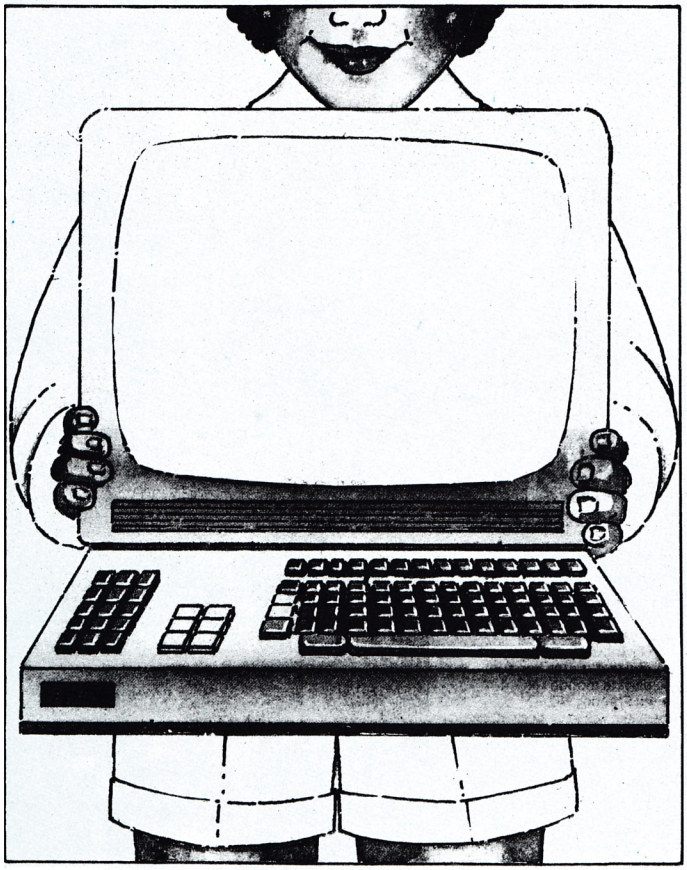
U (X)

Forma de obtenerlo: Pulsando simultáneamente CONTROL y la tecla de la letra U (o de la letra X).

Función asignada: Anular la presente línea de orden, permitiendo comenzar a escribir una nueva sin que la anulada sea ejecutada. La línea en pantalla se cancela con el símbolo #.

Ejemplo: Al intentar dar la orden DIR, se ha escrito por equivocación:

A > DOM



Gracias a U, podemos anular la orden y comenzar de nuevo, siendo la situación resultante en pantalla:

A > DOM #

El cursor está ahora en condiciones de comenzar una nueva línea de orden.

Nota: U y X cumplen la misma función.

Hasta aquí los comandos de edición de línea que permiten manipular lo escrito en una orden antes de ser ejecutadas.

Veamos ahora otra serie de comandos que, aunque no actúan sobre una línea de orden en el sentido hasta ahora expuesto, sí permiten al operador imponerse al sistema.

C

Forma de obtenerlo: Pulsando simultáneamente CONTROL y la tecla de la letra C.

Función asignada: Reinicializar el sistema y:

1. Catalogar el disco insertado en la unidad de disco que esté en uso, guardando en la RAM un mapa de asignaciones para el directorio (o catálogo) de dicho disco. Este proceso es conocido como **log in** o **logs.**

2. Interrumpir el proceso que se esté llevando a efecto para devolver el control al operador.

En CP/M 86, lo anterior es válido teniendo en cuenta que antes, y también mediante C, se deben interrumpir, uno tras otro, todos los procesos que se estén llevando a efecto.

En CP/M, y en términos generales, se debe dar C siempre que se cambien discos cuando el sistema está a nivel de comandos. Si se intenta escribir en el disco recién cambiado sin utilizar C, se obtendrá un mensaje de error.

Ejemplo: Habrá ocasión, más adelante, de probar este comando, no obstante, podemos tomar contacto mediante estos supuestos:

1. Si con la impronta A > en pantalla y un disco insertado en la unidad de disco, ordenamos un C, observaremos como el disco se pone en movimiento para ser recatalogado.

2. Si en el ejemplo propuesto en S, pulsamos C en lugar de S, se notaría que además de interrumpirse el listado, se devuelve el control al operador, mostrando la pantalla la impronta A >.

P

Forma de obtenerlo: Pulsando simultáneamente CONTROL y la tecla de la letra P sin activar la salida por impresora. Repitiendo esta operación se desactiva.

Función asignada: Enviar todas las salida enviadas a la pantalla, también a la impresora. Ordenando nuevamente P se cancela la salida por impresora. Se debe tener precaución con P ya que, si la impresora no está debidamente conectada, el sistema se "colgará" y deberemos inicializar el DOS, perdiendo el contenido actual de la RAM.

Ejemplo: Activar la impresora con P y poner en práctica el ejemplo dado en S, con lo cual veríamos el listado salir por la impresora.

Z

Forma de obtenerlo: Pulsando simultáneamente CONTROL y la tecla de la letra Z.

Función asignada: Dar por terminada la entrada de información a través del teclado.

Ejemplo: Este comando se verá en acción en toda su plenitud, al estudiar las órdenes PIP y ED.

Sabiendo ya como escribir, y corregir si ha lugar, las líneas de orden, sigamos el camino que nos conduce a comprenderlas.

Caracteres de control

1.° —¿Qué caracter

1.° —¿Qué carácter de control reinicializa el sistema? .....

Respuesta.— C

2.° —¿Qué carácter de control congela la salida de información? .....

Respuesta.— S

3.° —¿Qué carácter lo descongela? .....

Respuesta.— Cualquiera, excepto C que reinicializa el sistema.

4.° —¿Qué carácter de control permite o prohibe, la salida de información a través de la impresora? .....

Respuesta.— P

5.° —¿Qué carácter de control borra la línea de orden completa? .....

Respuesta.— U y X

6.° —¿Qué carácter de control mueve el cursor una posición atrás? .....

Respuesta.— H

7.° —¿Qué misión tiene I? .....

Respuesta.— Hace saltar el cursor al comienzo de la próxima zona de tabulación.

Los ficheros en el AMSTRAD

Cómo crear ficheros en disco

ORDENADOR - COMPUTADOR

Según el diccionario de la Real Academia de la Lengua Española, **computar** significa **contar o calcular una cosa por números**. Y un **computador electrónico es un aparato electrónico que realiza operaciones matemáticas y lógicas con gran rapidez.**

En otros dicionarios se pueden encontrar definiciones en este sentido, pero más ampliadas:

"Computador: Ingenio que, por medio de circuitos lógicos, puede efectuar cualquier operación basada en la de contar, y que opera en el sistema binario de numeración y bajo los criterios del álgebra de Boole".

Así pues, un computador se limita a actuar de acuerdo con la definición anterior, y, consiguientemente, está incapacitado para actuar como una máquina de escribir, o un tubo de rayos catódicos... Pero si puede estar en condiciones de controlarlos y vincularse con ellos.

En este orden de cosas, es evidente que, si el computador no estuviera habilitado para comunicarse con el exterior, su utilidad sería nula, y, de aquí, la necesidad de unirlo con otros ingenios a través de los cuales pueda recibir mensajes del exterior (entradas) y, a su vez, emitir los suyos propios (salidas).

Concluyendo, un **computador** es una unidad física destinada a operaciones basadas en la de contar, y que opera en el sistema binario y bajo los criterios del álgebra de Boole, y un **ordenador** es una suma de distintas unidades físicas, cada una dedicada a su propia actividad y, todas ellas, dirigidas por un computador teniendo por fin el proceso de datos automáticamente.

Para completar estar puntualizaciones, necesarias por otra parte para salir del confusionismo existente y poder expresarnos con precisión, diremos que un sistema de computación equivale al concepto de ordenador.

CPU Y MEMORIA INTERNA

De todos los elementos que componen un moderno computador, sólo hay dos que realmente interesen ahora a nuestro propósito: La CPU y la memoria.

La CPU, acrónimo de **Central Processing Unit**, vale por unidad central de proceso o microprocesador.

En el microprocesador se realizan todas las operaciones de control, comparación y aritméticas que tienen lugar en un computador; es un circuito integrado de alta densidad.

La memoria de un computador, imprescindible para que el microprocesador cumpla su función, está formada por una serie de **chips**.

Una disposición física real de la distribución de estos elementos sobre el circuito impreso principal, no nos ayudará tanto como el siguiente esquema, en el cual las flechas nos indican los flujos de información y los bloques de las partes fundamentales a considerar.

Unas palabras sobre la forma de instruir al microprocesador cierran este epígrafe.

Un microprocesador sólo puede procesar instrucciones y datos en forma de números binarios.

El conjunto de instrucciones que interpreta un microprocesador recibe el nombre de código máquina de ese determinado microprocesador.

Así, los microprocesadores 8080 A, 8085 y Z80 tienen un juego de instrucciones común, y, consiguientemente, pueden ejecutar programas desarrollados con el mismo código máquina.

A los microprocesadores 8086 y 8088 les sucede igual entre ellos.

MEMORIA

Desde cualquier punto de vista, la memoria es la capacidad de retener y recordar información.

Según los especialistas, la memoria humana puede dividirse en tres sistemas principales: **memoria sensorial** a **corto plazo** y a **largo plazo.**

El almacenamiento sensorial de información es sumamente breve, llegando, en la visión, a una décima de segundo.

Haga la prueba de recordar el tacto de la última cosa que tocó y comprobará que o bien no puede, o bien le cuesta trabajo.

La memoria a corto plazo conlleva un mayor tiempo de retención de la información. Tal sería el caso de las reuniones de trabajo o las citas para comer en una fecha más o menos próxima.

La memoria a largo plazo se refiere a ese tipo de información que nuestro cerebro retiene constantemente. Un ejemplo típico y extremo podría ser la tabla de sumar que aprendimos de niños en el colegio.

Para concluir con esta breve incursión en terrenos que me son extraños, añadiré que una persona con buena memoria será, normalmente, más brillante y sufrirá menos inconvenientes que otra que tenga mala memoria.

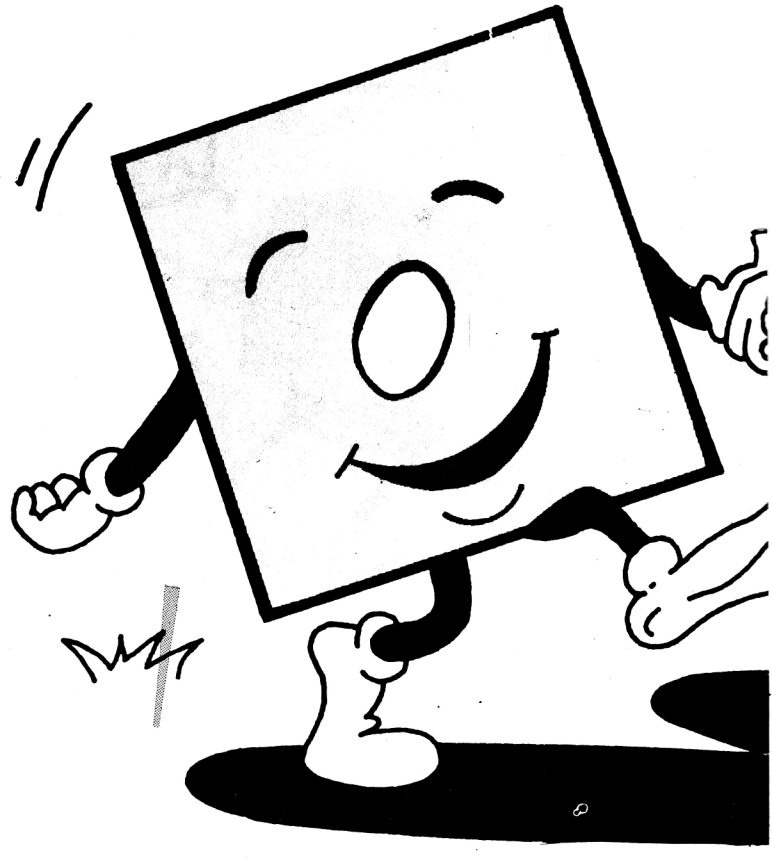
Veremos ahora la analogía existente entre lo expuesto anteriormente y la memoria de un ordenador.

Al trabajar en modo directo o inmediato a través del teclado de un ordenador, estamos utilizando una especie de memoria sensorial del computador, ya que, ejecutada la orden (al apretar ENTER), nada ha quedado en la memoria de éste.

Cuando utilizamos nuestra máquina como una calculadora, estamos aprovechando este tipo de memoria. El siguiente ejemplo en BASIC es clarificador: PRINT 525 * 1570 * 270.

La memoria a corto plazo podría equivaler a los programas que tecleamos o introducimos en la memoria del computador procedentes de un cassette o un disquete.

Este tipo de información, se mantiene en memoria mientras no se desconecte la máquina; siempre, claro está, que deliberadamente no instruyamos al computador en otro sentido.



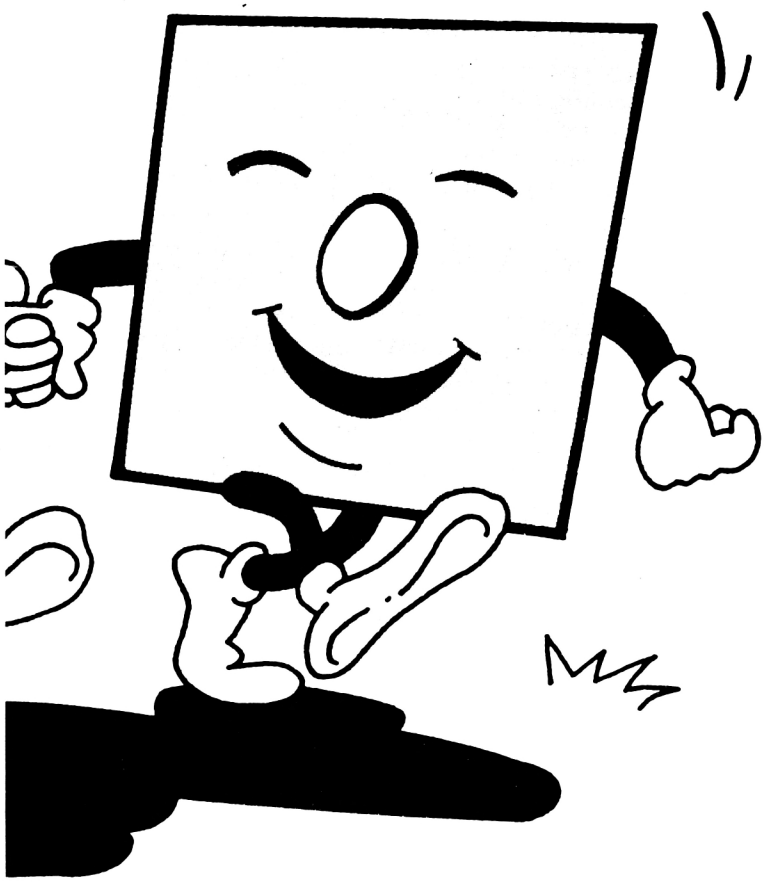
La información, sea cual fuere, se almacena dentro de la memoria en unidades del tipo *byte*, quedando a disposición del microprocesador.

A cada uno de los lugares donde se puede depositar un byte se le conoce como posición de memoria. Todas las **posiciones de memoria** está numeradas de tal forma que podemos referirnos a cualquiera de ellas en función de su dirección de memoria.

Las direcciones de memoria comienzan en la dirección 0 y pueden llegar —para un microprocesador de 8 bits— hasta la 65.535.

Un símil válido podría ser el de una calle con un vecino por casa y todas las casas en la misma acera. En este caso, cada vecino sería un **byte** o **unidad de información**.

Los habitantes de cada casa están representados por diferentes combinaciones de **8 bits.** Es decir, por un **byte**.



Dentro del computador, la parte de su electrónica destinada a guardar esta información se denomina RAM —acrónimo de Random Access Memory— y, en lenguaje informático, es conocida como "volátil", puesto que todo su contenido desaparece al privarla de energía eléctrica.

Otra característica de la memoria RAM es que tanto sirve para guardar información en ella como para extraerla, según nos interese.

La memoria a largo plazo se distingue de las otras dos en que la información que contiene no desaparece al cortar el suministro eléctrico al computador.

Dentro de este tipo podemos establecer la siguiente clasificación:

Situada dentro de la memoria interna y situada fuera del propio computador.

La memoria ROM —acrónimo de Read Only Memory— pertenece al primer tipo, ocupando una zona de la memoria interna total.

En la memoria ROM no se puede guardar información. Sólo se puede extraer, con lo cual su contenido permanece invariable en el tiempo.

El conjunto de programas contenidos en la ROM permiten que ciertas operaciones sean posibles al conectar la máquina, como veremos en su momento.

Finalmente, y aún dentro de la memoria a largo plazo, tenemos los soportes de memoria externos al computador, cuya misión es almacenar información y mantenerla a disposición de éste.

Ejemplos típicos son las casetes y discos magnéticos.

Gracias a los soportes externos, tenemos la posibilidad de transferir el contenido de la memoria RAM a estos, antes de desconectar la máquina, y, de esta forma, no perder su información.

Los *bytes* contenidos en la RAM, son transferidos secuencialmente al soporte externo para su almacenamiento, y, en sentido contrario y de la misma forma, son recuperados por la memoria RAM al cargar una información procedente del soporte externo.

En este sentido, es usual llamar **escritura** al proceso de grabación de información en el soporte externo, y **lectura** al de recuperación de la misma por parte de la memoria RAM.

Con el objeto de simplificar y dejar este último punto suficientemente claro, diremos que la RAM **lee** en el disco o **escribe** en él, según carge o grabe información en/o de el disco.

Para concluir con este epígrafe, se hace imprescindible resaltar que, mientras un ser humano puede sobrevivir con poca o ninguna memoria, un computador quedaría descalifi-

cado como tal, siendo manifiesta su incapacidad operativa.

ESQUEMA TIPO DE MICROORDENADOR. FLUJOS DE INFORMACION

Las flechas indican el sentido en que circula la información cuando llega el momento de su transferencia entre ordenador y periférico.

Periférico es el nombre que, de forma genérica, recibe cualquiera de las máquinas susceptibles de ser conectadas con el computador.

TRANSMISION

Página atrás, definimos la información como una transmisión de datos.

De aquí, parece poder deducirse que el efecto de transmitir resulta en un traslado de información de un ente físico a otro.

Al ente que emite la información se le llama **emisor.**

Al que la recibe**, receptor.**

También, ha quedado establecido que la memoria almacena información, y si suponemos que tanto el emisor como el receptor en la transmisión tienen memoria, el traslado de información correspondiente no lleva implícito que el emisor pierda el contenido de su memoria en beneficio de la del receptor.

En otras palabras, después de una transmisión, el emisor y el receptor comparten el conocimiento transmitido.

Nóisimsmart ed oidem le raidutse a somesap.

Bueno, no se asuste, lo que he querido decir es: Pasemos a estudiar el **medio** de transmisión.

El único problema que le ha surgido es que he utilizado, como medio de transmisión de ese mensaje, la escritura en español, pero con las letras de cada palabra colocadas en orden inverso y las palabras invertidas.

Dicho de otra forma, yo, el **emisor** de información, me he comunicado con usted el **receptor**, a través de un **medio**, la escritura, pero con un **código** no acordado.

El **soporte** de esta información es el papel, o lo que es igual está memorizada mediante la escritura en papel.

En este punto se exige hacer una recapitulación.

Para ello, podemos decir que, para que una transmisión sea eficaz, requiere de **un emisor, un receptor, un medio, un código** y **ds soportes** de memoria (uno para el emisor y otro para el receptor).

Con estos antecedentes, apliquemos los conceptos anteriores a la transmisión de información entre el computador y una unidad de disco —disk drive— conectado a él; este periférico tiene reservado un capítulo. Consiguientemente, el puesto de **receptor** y **emisor** lo tomarán, según las circunstancias bien el computador bien el disco.

El **medio** es el sistema de enumeración en base dos, en forma de **bytes**.

El **código** es el ASCII, explicado anteriormente.

El **soporte de memoria del computador** es la RAM, volátil ya que su contenido se pierde en el momento de la desconexión.

El **soporte de memoria de la unidad de disco** son los discos, de los que hablaremos en otro epígrafe.

FICHERO. CONCEPTO

Un fichero es un grupo de datos relacionados entre sí.

Nada impide que un fichero contenga un solo dato.

Dentro del ordenador, y con respecto a él, se individualiza cada información merced al concepto de fichero.

A cada fichero —o información agrupada e individual— se le asigna un nombre, el cual le distinguirá de entre los demás.

La denominación de un fichero está sometida a ciertas reglas que veremos en su momento.

Baste de momento saber que un fichero puede contener cualquier información:

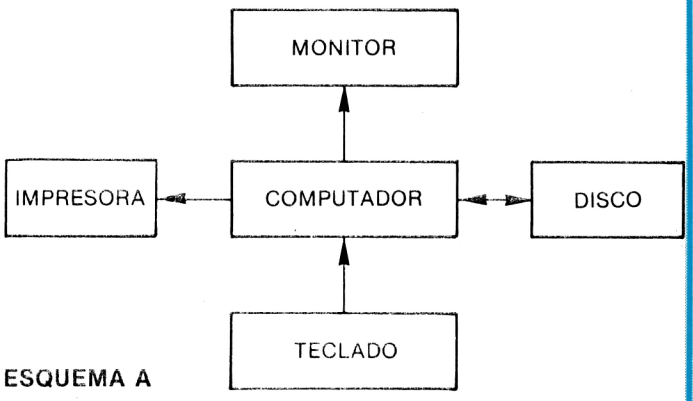
* Relación de los equipos de la liga de fútbol.
* Nombres de los participantes en un campeonato de mus.
* Un programa en BASIC.

Y que nombres válidos para estos ficheros serían:

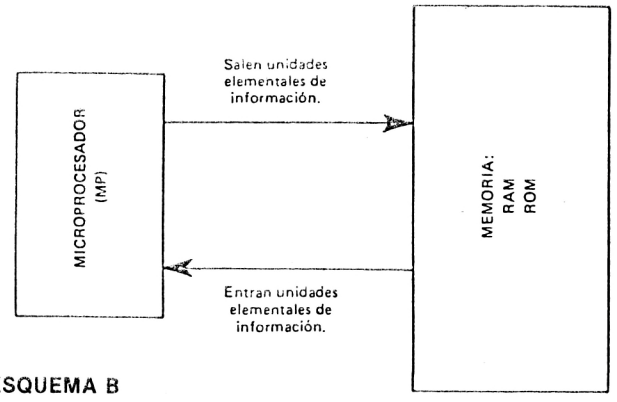
* LIGA
* MUS
* CUENCAS

Para concluir con esta breve introducción al concepto de fichero, añadiremos que toda transmisión de información entre la memoria interna del computador y un disco exige referirse a ella por su nombre —nombre del fichero— para diferenciarla de cualquier otra posible.

Estas ideas se irán consolidando y ampliando a lo largo de la lectura de los sucesivos epígrafes.



ESQUEMA A



ESQUEMA B

EL BASIC DEL AMSTRAD (II)

CONCEPTOS PREVIOS

Un programa es una secuencia de instrucciones susceptibles de ser interpretadas por un computador, el cual las ejecutará, una tras otra, de principio a fin.

Línea de programa

La secuencia de instrucciones que conforma el programa, se establece gracias a las líneas de programa, cada una de las cuales está estructurada de la siguiente forma:

N.° de línea Comando Argumento Fin de línea

El número de línea establece el orden de ejecución de las instrucciones, desde el menor al mayor, sin que por ello deban ser consecutivos.

Los números de líneas deben estar comprendidos entre 0 y 9999.

El comando puede ser cualquier palabra del BASIC, la cual, cumplirá su función sobre el argumento.

El *Fin de línea* se establece al apretar la tecla ENTER, con lo que la línea en cuestión pasa a la memoria del computador.

Una línea de programa puede estar formada por una sola sentencia como la esquematizada más arriba, o bien por varias, con lo que se tendría una línea multisentencia como la que sigue:

Número de línea Comando Argumento: Comando Argumento Fin de línea

Como puede apreciarse, una sentencia se separa de la siguiente por medio de dos puntos (:), excepto la última que requiere el ENTER para establecer el Fin de línea.

Modos de operar

Cuando una sentencia es introducida en el AMSTRAD sin número de línea, se ejecutará inmediatamente al apretar ENTER. Esta forma de operar se conoce como MODO DIRECTO.

En modo directo también se pueden introducir multisentencias.

El MODO PROGRAMA exige introducir las sentecias con sus números de línea correspondientes y sólo serán ejecutadas secuencialmente mediante RUN y ENTER.

Modo Programa. Cómo se introduce un programa

El objeto fundamental de este curso es mostrarte los fundamentos en que se basa la programación, pero, previamente, debes tomar contacto con algunos conceptos que, aun no entendiéndolos en su plenitud, han de comenzar a ser conocidos para evitar dudas.

Una vez diseñado un programa, y con el fin de introducirlo en el computador por primera vez, es necesario apretar todas y cada una de las teclas correspondientes a los caracteres escritos en las líneas del programa. Es decir, hay que teclear todos los números, letras, símbolos y signos escritos por el programador en cada línea del programa, incluidos los espacios.

Las mayúsculas y los caracteres situados en la parte superior de las teclas se consiguen apretando simultáneamente SHIFT y la tecla apropiada.

Un ejemplo te ayudará a comprenderlo mejor. Vamos a teclear una línea del programa:

30 INPUT "Longitud de la base: " ; b

Apretando cada una de las letras que contienen estos caracteres, hay que seguir los siguientes pasos:

— Teclear la palabra de la orden (INPUT) y otro espacio.

— Teclear la palabra del comando (INPUT) y otro espacio (opcional).

— Teclear las comillas ("), el texto ("Longitud de la base"), los dos puntos (:), un espacio, las comillas ("), punto y coma (;) y el nombre de la variable (b).

— Finalizada la línea, hay que apretar la tecla ENTER, RETURN o la que el MANUAL indique para que el computador la almacene en su memoria.

Al ir tecleando la línea, ésta va apareciendo en la pantalla del monitor.

Cuando, —una vez finalizada— se introduce en la memoria del computador, el cursor se queda sólo al comienzo de una línea en blanco.

Recuerda que, aunque las líneas del programa se tecleen en desorden, el computador siempre las memoriza siguiendo la secuencia de sus números.

Por último, debes saber cómo manejar algunos signos (el punto, la coma, los dos puntos, y el punto y coma) que, además de su valor ortográfico, tienen un valor diferente en el lenguaje de programación BASIC.

Todos estos signos tendrán valor ortográfico únicamente si son tecleados entre comillas (") dentro de una expresión alfanumérica.

En programación BASIC, estos signos cumplen diferentes funciones:

El punto (.) se utiliza para separar la parte entera de la parte decimal de un número, en lugar de la coma decimal. Por ejemplo, hay que teclear 2.53 en lugar de 2'53 o de 2,53.

La coma (,) se utiliza para:

— Separar los datos de salida en las sentencias PRINT. Por ejemplo: **PRINT A, B, C.** Teclea esta línea en modo directo para ver qué pasa.

— Separar los datos de salida en la instrucción DATA. Por ejemplo: **DATA A, B, C.** Teclea la línea en modo directo.

— Para separar dos expresiones obligando a que la segunda se imprima a partir de la siguiente zona de tabulación de la pantalla ocupada por la primera expresión. Por ejemplo: **PRINT "2+2=",2+2.** Tecleala en modo directo.

Los dos puntos (:) sirven para separar comandos en las líneas multisentencia. Por ejemplo: **INPUT A: PRINT A.** Tecleala como en los ejemplos anteriores.

El punto y coma (;) puede ser utilizado:

— Para separar datos, como las comas.

— Para obligar a que dos expresiones se impriman en pantalla una a continuación de la otra. Por ejemplo:

```
10 PRINT "Me llamo";
20 PRINT "Juan"
```

Al ejecutar este programa, tecleando **RUN** y **ENTER**, aparecerá impreso en pantalla: Me llamo Juan.

Cómo se ejecuta un programa

Cuando hayas escrito un programa para resolver un problema y lo hayas tecleado, habrás puesto una secuencia de órdenes en la memoria del computador. Pero, para hacer trabajar este programa, es decir, para hacer que el ordenador obedezca una tras otra las instrucciones de la secuencia necesitas **correrlo** o **ejecutarlo**.

Para **correr** o **ejecutar** los programas, el BASIC tiene el comando **RUN** y, si a continuación, la tecla **ENTER**, harás comenzar la ejecución del programa que está en la memoria del computador, y podrás trabajar con él para resolver el problema planteado.

En caso de que al teclear una línea de programa cometas un error de sintaxis, el ordenador lo detectará y le avisará al tratar de ejecutar el programa, situándose el cursor en la línea que contenga el error. Entonces, deberás corregirlo siguiendo las instrucciones que te indicamos a continuación.

DEPURANDO UN PROGRAMA

Cómo se depura un programa

Es muy posible que cualquier programa que hayas escrito, tecleado y ejecutado por primera vez no funcione correctamente. Esto les suele suceder también a los mejores programadores y teclistas.

La causa de que el programa no funcione correctamente será —casi con seguridad— algún tipo de error que h. yas cometido al escribirlo o teclearlo. El propio ordenador te indicará el error cometido mediante un mensaje que aparecerá en pantalla. No te desanimes: dispones de medios para hacerlo funcionar correctamente.

La tarea de hacer funcionar correctamente un programa, aprovechando razonablemente la memoria del computador y corrigiendo su errores, se llama **depurar el programa**. Este trabajo podrás realizarlo utilizando las facilidades de **edición** que te ofrece el AMSTRAD.

Como primera norma, debes acostumbrarte a inicializar todo el sistema cuando vayas a comenzar a trabajar sobre un nuevo programa en otras palabras, limpiar la memoria interna del AMSTRAD de cualquier contenido previo. Esto lo conseguirás apretando las teclas CONTROL, SHIFT y ESC (CONTROL, MAY y ESC en el modelo con teclado español) simultáneamente. Otro sistema, más expeditivo y menos recomendable, es cambiar a OFF el interruptor que está marcado POWER a la derecha o detrás de la carcasa.

Hecho esto, y una vez comenzada la introducción de un programa, puede darse el caso de que una determinada línea no nos interese mantenerla en el listado y, consiguientemente, deseemos borrarla, para lo cual bastará escribir su número seguido de RETURN.

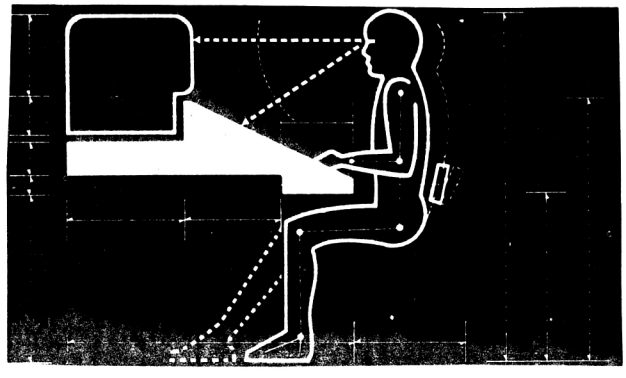
Para estudiar este caso, y otros que se propondrán más tarde, veamos un sencillo programa, el cual deberás teclear en tu computador exactamente como se indica.

Primero escribe

```
10 PRINT "En un lugarr de la Mancha"
20 STOP
30 PRRINT "de cuyo nombre quiero
   acordarme"
```

Al acabar cada una de estas líneas aprieta RETURN, con el fin de hacerla pasar a memoria y colocar el cursor al comienzo de otra línea.

Una vez hecho esto, escribe 20 seguido de ENTER; en este momento la línea 20 habrá desaparecido... aunque, de momento, no se pueda apreciar.



Fichas del AMSTRAD (II)

POR A. BELLIDO

Guía de palabras BASIC

DEFREAL

Vale todo lo dicho para DEFINT, pero referido a los números reales.

DEFSTR

Vale todo lo dicho para DEFINT, pero referido a las variables alfanuméricas.

```
DEFSTR B : b="ABCD" : PRINT B
ABCD
```

DEG

Ordena cambiar de radianes a grados.

Radianes es la unidad para iniciar el sistema.

DELETE
DELETE —n.° de línea
DELETE n.° de línea—

Borra todas, o desde el principio al **—n.° de línea**, o desde el **n.° de línea—** al final, las líneas de programas.

```
10 DEFFN A(x,y)=x+y*2
20 FOR x=1 TO 10 : LET Z=FN (A,10) :
PRINT Z;
30 NEXT
40  DELETE —10
```

Al ejecutar el listado, desaparece del mismo la línea 10.

DERR

Esta función guarda el número del código del último error derivado del disco.

PRINT DERR
TIPO DE ERROR

142 El rumbo (**stream**) no está adecuado.
143 ..... Fin de fichero en el disco.
144 ..... Comando incorrecto.
145 ..... Fichero existente.
146 ..... Fichero no existente.
147 ..... Directorio lleno.
148 ..... Disco lleno.
149 ..... Disco cambiado.
150 ..... Fichero de sólo lectura.
154 ..... Fín de fichero (**End of file**).

DI

Impide cualquier tipo de interrupción en la ejecución de una parte de un programa.

Con **EI** se permite nuevamente las interrupciones.

DIM ¡nombre (elementos)!

Dimensiona una matriz denominada **nombre** reservando tantas variables como indica el número dado por **elementos**. Si no se especifica, se reservan 10.

El subíndice de la primera variable es 0.

```
DIM MULTI$ (2,3)
```

Reserva 12 variables denominadas MULTI$ (0,0) , MULTI$ (0,1),..., MULTI$ (2,3)

DRAW ¡expresión 1, expresión 2!|, expresión 3||, expresión 4|

Dibuja una recta desde la posición que a la sazón ocupe el cursor de gráficos y hasta la

posición determinada por **expresión 1** (abscisa) y **expresión 2** (ordenada).

Las abcisas (eje horizontal) van de 0 a 639: 640 **pixels**.

Las ordenadas (eje vertical) van de 0 a 399: 400 **pixels**.

La **expresión 3**,si se usa, determina el color de la tinta en el campo de 0 a 15.

Con la **expresión 4,** si se usa, se fija el modo lógico de impresión de **pixels** entre los que están en la pantalla y los que se activan con DRAW: Normal=0, XOR=1, AND=2, OR=3.

```
5 CLEAR: MODE 1
10 INPUT "tinta"; x: INPUT "modo"; z
20 DRAW 400,0,x,z: PLOT 439,0: DRAW 639,0
30 DRAW 0,0,x,z
40 IF INKEY$=" " THEN 35
50 GO TO 5
```

DRAWR ¡expresión 1, espresión 2! | *, expresión 3*|| *,expresión 4* |

Donde **expresión 1** y **expresión 2** sn desplazamientos relativos de abscisa y ordenada, y el resto coincide con lo dicho en DRAW.

En el ejemplo anterior cambiar esta línea

```
30 DRAWR 0,100,x,z
```

EDIT ¡número!

Muestra en pantalla la línea de programa cuyo número es **número** y queda en condiciones de ser manipulada.

Ver APENDICE 2.

END

Da por terminada la ejecución de un programa.

EOF

Detecta si el fin de un fichero, (End of file), en disco ha sido alcanzado.

ERASE ¡matriz 1, matriz 2,...matriz n!

Borra las matrices denominadas **matriz 1, matriz 2,...,matriz n,** dejando disponible la memoria equivalente.

ERL

Esta función guarda el número de línea donde se produjo el último error.

```
10 ON ERROR GOTO 30
20 q+a
30 PRINT ERL
40 STOP
```

ERR

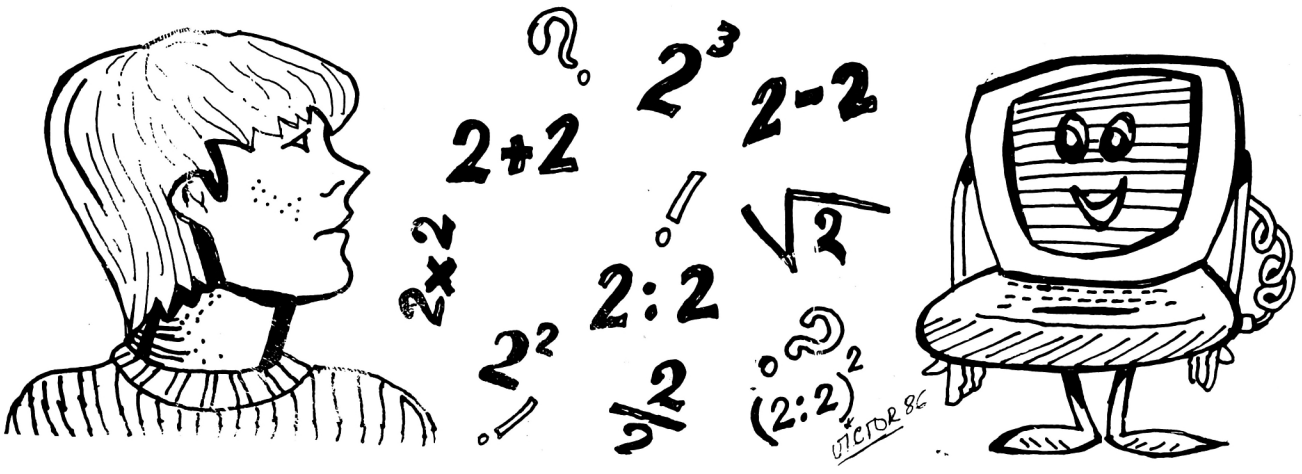
Esta función guarda el número del código del último error producido.

```
10 ON ERROR GOTO 30
20 PRINT ERL, ERR
40 STOP
```

ERROR ¡expresión!

Este comando genera, a todos los efectos, el tipo de error cuyo código es el dado por **expresión.**

```
5x=2
10 IF INKEY$ =" " THEN 10 ELSE ERROR x
```



Programas comentados en

BASIC

El juego de las 21 cerillas

ESTE sencillo juego es ya un clásico, dentro de los que podemos denominar matemáticos o de base matemática.

A pesar de su sencillez, tiene la particularidad de ofrecer una estrategia ganadora, para el jugador que actua en segundo lugar.

REGLAS DEL JUEGO

De un montón formado por 21 cerillas, los jugadores, por turno, deben tomar 1, 2, 3 ó 4 cerillas, de tal manera que pierde el que coje la última.

ESTRATEGIA

Un jugador nunca pierde, si conoce la estrategia del juego, que se deriva del análisis de las propias reglas.

Observemos los números que intervienen y busquemos alguna relación entre ellos.

— Son 21 cerillas, recogidas 20 sobra 1 para el perdedor.

— Sólo pueden tomarse 1, 2, 3 ó 4 en cada turno.

Si realizamos las combinaciones de estos cuatro números tomándolos por parejas y sumándolos, obtendremos:

1+1	2	2+1	3	3+1	4	4+1	**5**
1+2	3	2+2	4	3+2	**5**	4+2	6
1+3	4	2+3	**5**	3+3	6	4+3	7
1+4	**5**	2+4	6	3+4	7	4+4	8

Y estableciendo la frecuencia con que se presenta, cada suma, veremos que:

SUMA	N.° VECES QUE APARECE
2	1
3	2
4	3
5	4
6	3
7	2
8	1

La suma 5, es la más frecuente y aparece en todas las columnas de posibles combinaciones. ¿Qué nos indica esto? Pues dos cosas, primero que sea cual fuere el número de cerillas que tome el primer jugador, cojiendo el complemento a 5 (5 - elegidas por el primer jugador), puede siempre realizarlo y segundo, si mantiene esta forma de juego (estrategia) en la cuarta jugada se habrán retirado $5 \times 4 = 20$ cerillas, de manera que al primer jugador le quedará únicamente una, lo cual le hará perdedor.

Conclusiones:

De este razonamiento se desprenden tres cosas fundamentales:

1.ª El ganador debe comenzar a jugar en segundo lugar.

2.ª Cojerá siempre 5 —las elegidas por el primero.

3.ª El primer jugador pierde siempre, si se cumplen las dos primeras condiciones.

PROGRAMA

Una vez conocida la estrategia, vamos a realizar un pgm. de manera, que el ordenador asuma el papel de segundo jugador. El primer jugador será sin duda un amigo, al que deseas gastar una pequeña broma, pues evidentemente se enfrenta a un rival invencible.

Variables utilizadas

C — Contador de juegos.
N — Número de cerillas que quedan en el montón.
I — Cerillas que toma el jugador.
X — Cerillas que toma el ordenador.
XS — Var. del INPUT para nuevo juego.
ZS — "VEZ" o "VACES" según que el número de partidas sea uno o más.

Comentarios

20 Inicializa el contador de juegos.
30 Pantalla en 40 columnas.
40-60 Presentación. Reglas del juego.
70-80 Cabecera del juego.
90 Inicializa la var. N a 21.
100 Selección del número de cerillas a tomar por el jugador.
110-120 Control del INPUT. Recuerda que CHRS(7) provoca un pitido (BEEP) y CHRS(11), sube el cursor a la línea anterior, de esta forma borra el texto del INPUT.
130 Decrementa el montón en I unidades tomadas.
140-150 Escribe jugada del jugador y resto del montón.
160 Juega el ordenador, según estrategia descrita, y dec ementa el montón.
170 Escribe jugada realizada por el ordenador.
180 Controla el n.° de cerillas del montón. Si hay más de una el juego continua por la línea 100.
190-210 El juego ha terminado. Actualiza el contador C y ZS toma el valor apropiado (singular-1 juego o plural-varios).
220-230 Escribe el número de partidas perdidas (ya que ganadas no puede haber ninguna).
240-250 INPUT para nuevo juego y control del mismo.
260-275 Si no se va a jugar más, se poduce un comentario jocoso, en letras grandes (MODE ∅). CHRS(10) lleva el puntero abajo.
280 FIN DEL PROGRAMA.
500-530 Subrutina de continuación del juego. Los caracteres 224 y 225 corresponden a la cara seria y sonriente y pertenecen a los caracteres standard del Amstrad.

MEJORAS

Como todo pgm., es susceptible de mejoras y el expuesto anteriormente es muy elemental, te proponemos mejorarlo a fin de dar más cuerpo y vistosidad al propio programa y más desarrollo al juego.

— La primera mejora que se nos ocurre es obvia, utilizar color y gráficos ilustrativos. Un montón de cerillas representando en la pantalla del que van desapareciendo, las que se van tomando del mismo, le dará mayor realismo, puede añadirse además dos cajas o montones donde se depositen las tomadas por cada jugador.

— A fin de personalizar el juego, podía preguntar el pgm. el nombre del contrincante, de manera, que luego aparezca en las frases más o menos jocosas, del final de partida.

— La tercera, un poco más seria bajo el punto de vista de la programación, sería el dotar al ordenador de dos estrategias, la primera "ganadora" que es la que hemos desarrollado y la segunda, de tipo "aleatorio", es decir, que facilite el poderle ganar.

De esta forma, una vez tu amigo, se haya dado por vencido, podais vosotros jugar y ganar, para demostrar que no es tan difícil. Evidentemente habreis de recurrir a un truco sutil, que permita pasar al ordenador, de una estrategia a otra.

```
10 REM JUEGO DE LAS 21 CERILLAS
20 C=O
30 MODE 1
40 PRINT "DE UN MONTON CON 21 CERILLAS, PUEDES TOMAR CADA VEZ 1, 2, 3 ó 4"
50 PRINT :PRINT "EL QUE COJE LA ULTIMA CERILLA ES EL PERDEDOR"
60 PRINT "PtPtPtPtPtPtPtPtPtPtPtPtPtPtPtPtPtPtPtPtPtPtPtPtPtPtPtPt"
70 PRINT : PRINT " TU YO QUE DAN"
80 PRINT " ____________________"
90 N=21
100 INPUT "Cuantas cojes";I
110 I =INT(I)
120 IF I<1 OR I>4 THEN PRINT CHR$(7)+CH R$(11)+ "NO HAGAS TRAMPAS " :GOTO 100
130 N=N-I
140 PRINT CHR$(11);
150 PRINT SPC(3); I;SPC(11);N
160 X=5-I : N=N-X
170 PRINT SPC(9); X;SPC(5);N
180 IF N>1 THEN GOTO 100
190 C=C+1
200 IF C=1 THEN Z$="VEZ " : GOTO 220
210 Z$ = " VECES"
220 PRINT :PRINT "HAS PERDIDO ";C;Z$
230 PRINT "=========================================="
240 PRINT :INPUT "Quieres jugar denuevo (S/N) ";x$
250 IF X$="S" OR X$="s" THEN GOSUB 500
260 MODE 0
265 PRINT :PRINT "NO HAS GANADO": PRINT CHR$(10)+ "NI UNA PARTIDA! "CHR$(225)
270 PRINT CHR$(10)+ "HACES BIEN DE-JANDOLO" : PRINT CHR$(10)+ "Y DEDI-CANDOTE A ...":PRINT CHR$(10)+"OTRA COSA"
275 FOR K=1 TO 6: PRINT CHR$(10):NEXT K
280 END
500 MODE 0:PRINT "Q U E M O R A L !"
510 FOR K=1 TO 300:LOCATE 10,9:PRINT CH
CH R$(224):NEXT K
520 CLS: MODE 1: GOTO 70
530 RETURN
```

Envíanos tu programa mejorado, con estas u otras ideas y lo publicaremos.

INTRODUCCION A LOS

FICHA NUMERO 1: CARGA DEL LENGUAJE LOGO EN EL AMSTRAD CP664

En las fichas numeradas, que seguirán a algunos capítulos veremos gran variedad de temas. En principio, la única relación que unos de estos temas guardarán con otros, es que todos se referirán al lenguaje Logo.

Es una sección pensada para comentar todo tipo de cosas de interés. Aquí aparecerán análisis de programas, curiosidades, paradojas, experiencias, etc.

En general, decir que se va a cargar un programa en un ordenador, significa que se va a instalar ese programa en su memoria para poder trabajar con él. Ese programa podría ser un lenguaje capaz de generar otros programas. Así que, cargar el lenguaje Logo, significa instalar el lenguaje en la memoria del AMSTRAD, de modo que "comprenda" las palabras y la sintaxis de este lenguaje; después de lo cual podremos escribir y ejecutar programas en Logo.

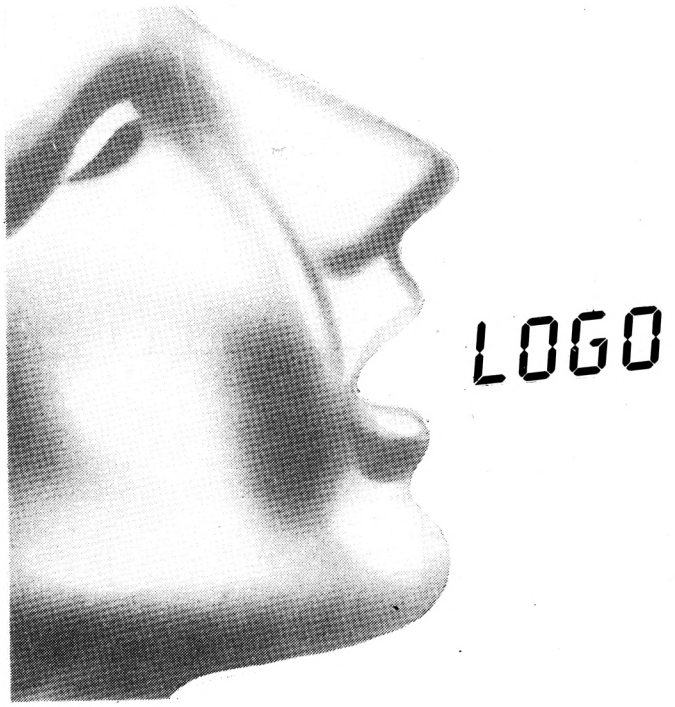
Con el ordenador se suministra un disco muy importante. Debemos poner mucha atención para que no se estropee. En la cara "A" contiene el sistema operativo CP/M y en la "B" el lenguaje DR. Logo.

Lo mejor es hacer copias de los discos importantes que poseamos. De este modo no se puede estropear el original, con las graves consecuencias que esto traería consigo.

Lo primero que vamos a ver en consecuencia, es la forma en que debemos proceder para copiar el disco.

No es este el lugar indicado para estudiar el concepto de sistema operativo. Bastenos saber, de momento, que para que un ordenador pueda ejecutar programas en distintos lenguajes (por ejemplo), es necesario que un programa "subyacente" gobierne algunas acciones de la máquina; por ejemplo, leer la información de los discos o escribirla en ellos.

Siempre que encendamos o apaguemos el AMSTRAD, debemos asegurarnos de que no hay disco alguno introducido en la unidad (podría resultar dañado). Al encender la máquina, se encuentra siempre bajo el control del sistema operativo AMSDOS, propio del AMSTRAD. Después de encender introduzcamos el disco suministrado, con la cara "A" hacia arriba. Para cargar el sistema operativo CP/M basta escribir |CP/M (el carácter | se obtiene pulsando simultáneamente las teclas SHIFT-@). Es indiferente que se escriba en mayúsculas o minúsculas. Después de algunos segundos aparece el mensaje.



PROGRAMAS EN LOGO

POR JUAN MARTINEZ PINTOR

"CP/M 2.2-Amstrad Consumer Electronics plc", y el prompt del CP/M " A≦", con el cursor esperando órdenes.

El sistema operativo CP/M contiene, grabados en el disco, varios programas con utilidades diversas. Una de ellas es un programa que nos permite copiar el contenido de un disco en otro. Trabajando con una unidad de disco este programa es el DISCCOPY.

Para ejecutarlo, basta escribir en la consola el nombre del programa. Escribamos pues DISCCOPY, pulsemos la tecla "enter", y esperemos acontecimientos. A partir de este momento, la máquina está bajo el control del programa DISCCOPY, del sistema operativo.

Para ejecutarlo, basta escribir en la consola el nombre del programa. Escribamos pues DISCCOPY, pulsemos la tecla "enter", y esperemos acontecimientos. A partir de este momento, la máquina está bajo el control del programa DISCCOPY, del sistema operativo.

En todo el proceso de copia, cada vez que el ordenador se refiera al "SOURCE DISC" introducimos la cara "B" del disco suministrado con el equipo; cada vez que se refiera al "DESTINATION DISC", introducimos nuestro disco en blanco, en el que queremos copiar el anterior. El orificio protector de escrituras accidentales, debe estar abierto en el disco principal (para no permitir la escritura en él), y cerrado en el disco copia (para poder escribir en él).

La operación de sacar un disco y meter otro, ha de repetirse varias veces, porque cada vez se copian 8 pistas y son 40. Cuando se termine el proceso de copia, el programa preguntará si se va a hacer alguna otra copia. Si contestamos que no, el ordenador pasará de nuevo a estar bajo el control del sistema operativo.

Vamos a ver a continuación como poner al ordenador en situación de trabajar con Logo.

Se puede cargar el lenguaje Logo, tanto si la máquina está bajo el sistema operativo CP/M como bajo el AMSDOS (al encenderla por ejemplo).

Bajo AMSDOS, una vez introducido el disco que contiene el Logo, basta escribir |CP/M (como hicimos antes para cargar el CP/M, con la otra cara). Bajo el CP/M, basta escribir la palabra LOGO.

En ambos casos el ordenador lee en el disco el programa que contiene el lenguaje Logo, y después de unos segundos, lo ejecuta. Esto equivale a que el control del aparato pasa a tenerlo el lenguaje Logo. Aparecerá entonces la pantalla borrada y una interrogación "?". Es el "prompt" del Logo. A su derecha el cursor (un cuadrado) esperando nuestras órdenes para empezar a trabajar.

CAPITULO II
MODOS DE OPERACION

* Una "instrucción" es una palabra o lista de palabras que indica al ordenador lo que debe hacer.

* Las instrucciones que provocan una acción, se llaman "órdenes".

* Las instrucciones que producen un resultado, se llaman "operaciones".

* Toda instrucción del tipo operación ha de ir precedida por una del tipo orden que la afecte, o por otra operación.

* Los "modos de operación" son los estados distintos en que puede estar el ordenador al trabajar en Logo.

* Cuando se carga Logo en el AMSTRAD, se sitúa en un modo desde el que se accede a todos los demás. Se llama "nivel superior" o "toplevel".

* Al comenzar la ejecución de un programa, el Logo abandona el nivel superior. Es el "modo ejecución".

* Un "procedimiento" es un programa escrito en Logo.

* Otro modo de operación es el "modo definición". Se usa para definir los nuevos procedimientos.

* Todo procedimiento consta de tres partes: "nombre", "cuerpo" y "final".

* El nombre es una palabra: identifica al procedimiento a todos los efectos.

* El cuerpo consiste en todas las instrucciones que forman el procedimiento.

* El final indica donde termina el procedimiento.

* Para entrar en modo definición, se escribe la palabra "to" seguida del nombre del nuevo procedimiento.

* Para concluir la definición se escribe la palabra "end" al principio de una nueva línea.

* Si se pulsa la tecla "esc" el Logo abandona lo que esté haciendo.

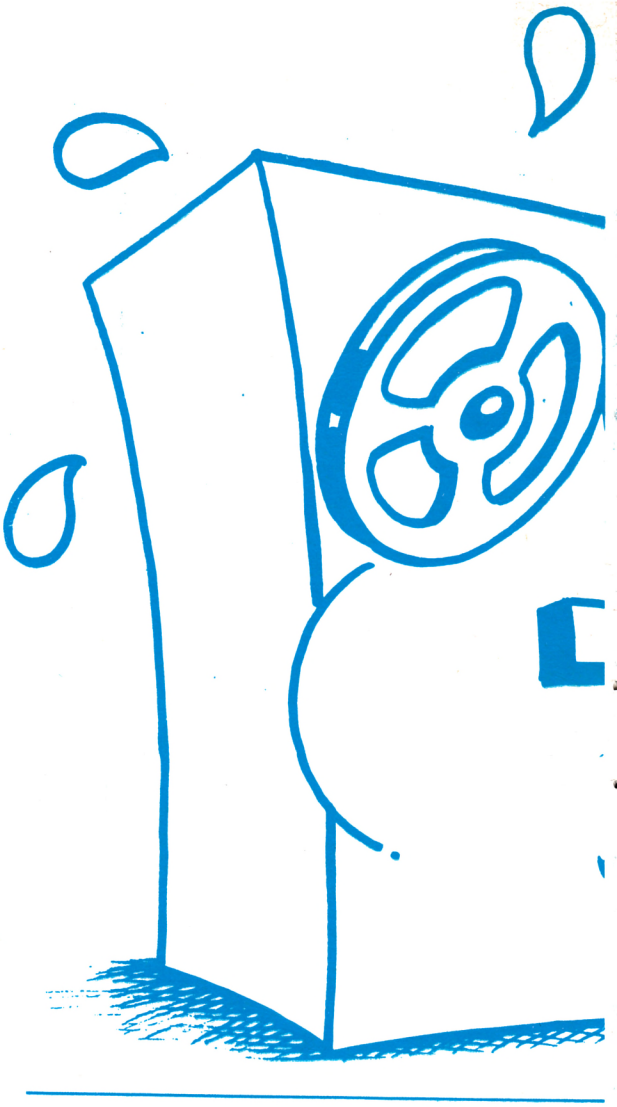
* El "modo edición", se puede utilizar para definir procedimientos y para modificar los ya definidos.

II.1. TIPOS DE INSTRUCCIONES: ORDENES Y OPERACIONES

Según se ha visto en el capítulo 1, al comenzar a trabajar en Logo el ordenador conoce unas cuantas docenas de palabras.

Una instrucción es una palabra o lista de palabras, que indica al ordenador lo que debe hacer cuando la ejecute. El Logo cuenta entre sus características el hecho de que es posible definir nuevas palabras y, por tanto, nuevas instrucciones, en función de otras que ya conoce o que se definirán posteriormente.

Cuando el ordenador ejecuta un programa escrito en un determinado lenguaje, realiza acciones de muy distinta naturaleza. Por ejemplo, borrar la pantalla, realizar cálculos, escribir resultados, hacer un gráfico. Estos ejemplos son suficientes para percibir que podemos diferenciar las instrucciones que producen un resultado, de aquellas otras que no lo producen.



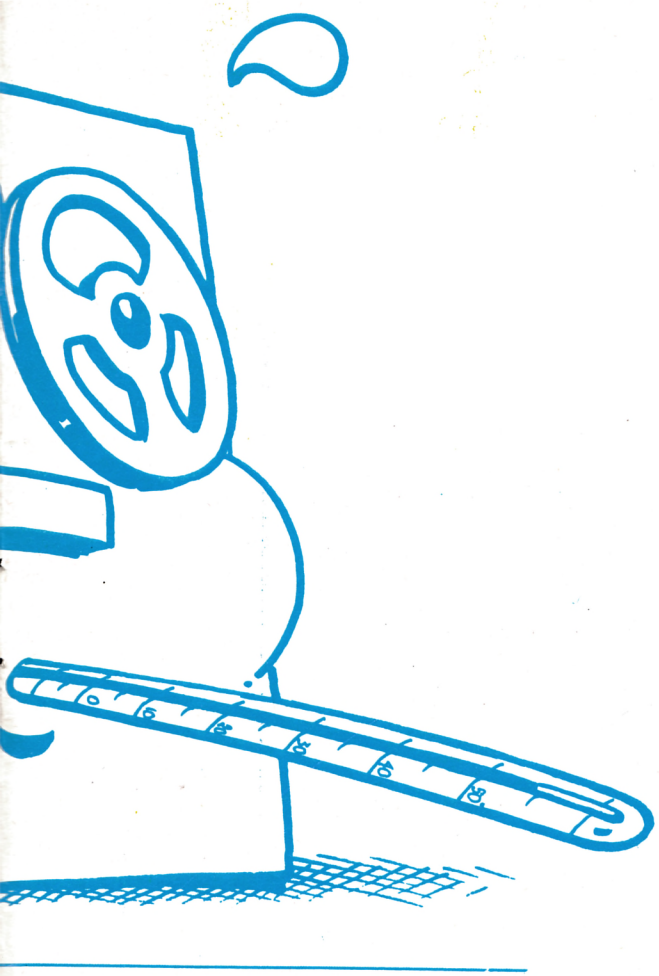
Las instrucciones que provocan que el ordenador lleve a efecto una acción, se llaman **ORDENES**. Por el contrario, las instrucciones que al ser ejecutadas, producen un resultado, se llaman **OPERACIONES**. Unas y otras son o bien primitivos o bien procedimientos. Es decir, el programador de Logo puede definir nuevas órdenes y nuevas operaciones.

EJEMPLO II.1

— Una instrucción que haga que la pantalla se borre es una orden. No produce resultado alguno; produce una acción.

— La instrucción que se utiliza para sumar dos números es una operación; da como resultado la suma de estos dos números.

Podemos utilizar y entremezclar en nuestros programas ordenes y operaciones, teniendo siempre presente la diferencia



conceptual existente entre ambas. En efecto, hay un hecho que debemos mantener siempre presente:

"Toda instrucción del tipo operación ha de ir precedida por una orden que la afecte e indique lo que se debe hacer con el resultado producido, o bien por otra operación uno de cuyos datos sea el resultado producido".

De otra manera, se produciría un error o no se haría nada con el resultado de la operación.

EJEMPLO II.2

— Para la suma se utiliza el signo "+". Una instrucción del tipo operación podría ser entonces "3+98". Pero si se escribe en un programa una instrucción así, aunque el ordenador calcule correctamente el resultado, no sabrá que ha de hacer con él, puesto que no se le ha indicado, haciendo preceder la operación por una orden.

II.2 ORDENES DIRECTAS: "TOPLEVEL"

Cuando el ordenador se prepara para trabajar en un lenguaje, en general, puede hallarse situado en uno de varios estados o modos, según lo que se este haciendo en cada momento. No es lo mismo, por ejemplo, que se este ejecutando un programa o que se este definiendo uno nuevo. Estos estados distintos, son lo que llamamos "modos de operación".

Cuando se carga el lenguaje Logo en el AMSTRAD, se situa este en un "modo" desde el que se puede acceder a todos los demás. Es el modo llamado **NIVEL SUPERIOR** o **TOPLEVEL**. A este estado es al que regresa la máquina cuando termina la ejecución de los programas. Y es a partir de este estado desde el que comienza su ejecución.

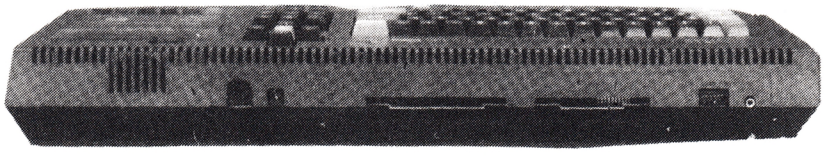
Además, cuando el ordenador se halla en toplevel permite al usuario la "ejecución directa" de muchas instrucciones. Para ejecutar una instrucción desde el nivel superior, solo tenemos que escribirla en el teclado, y después pulsar la tecla "enter". Entonces la instrucción se ejecuta y, a continuación, se regresa al nivel superior.

Podemos distinguir el estado en que esta la máquina, por algunas características de la pantalla. Así, cuando esta en toplevel, vemos que en la línea en la que se escribiran los próximos caracteres que se tecleen, aparece un signo de interrogación. Este carácter se llama **PROMPT**. A su derecha vemos un cuadrado, que nos indica donde se escribira el próximo carácter. Es el **CURSOR**.

En nivel superior podemos incluso escribir varias instrucciones seguidas, para que se ejecuten consecutivamente. Basta para ello, no pulsar enter entre ellas, y separarlas por un espacio en blanco. Puede suceder que se nos acabe la línea de escritura (el renglón), en un momento en que nos deje una palabra dividida en dos. Entonces el ordenador escribirá el caracter "!" al final de la línea y podremos seguir como si nada pasara. La palabra queda entonces dividida ante nuestra vista, pero no a la vista del ordenador, que la ejecutará normalmente. Esto sin embargo solo puede hacerse una vez antes de pulsar enter: es decir, si se siguen escribiendo instrucciones en el renglón de abajo, al acabarlo se tendrá que pulsar enter, porque el ordenador no permite seguir escribiendo. Esta limitación es válida no solo en el nivel superior, como ya veremos.

DOCTOR LOGO

Programas comentados en LOGO



DISPARO (LOGO) (Movimiento Absoluto) (DD)

Instrucciones de manejo:

Enciéndase la máquina e introduzcase el disco que contiene el Dr. Logo. Escríbase ICMP y el lenguaje Logo se instalará en el ordenador.

Para cargar el programa basta escribir **load "disparos**. Después de unos segundos el programa se carga en la memoria. El ordenador informa en pantalla de los nombres de los procedimientos cargados:

diana defined
inicio defined
asigp defined
res defined
disparos defined

Como el lenguaje Logo consume mucha memoria no se debe cargar este programa después de haber estado trabajando con Logo en otra cosa. Lo mejor es reinicializar antes el sistema.

Para comenzar a jugar escríbase el nombre del programa: disparos. Después "enter". Aparece en la pantalla un resumen de las instrucciones. Cuando se haya leído basta pulsar una tecla para que el programa continue.

El juego consiste en realizar 10 disparos con la tortuga a una di na que aparece en pantalla. Las posiciones de la tortuga y la diana las elige el ordenador al azar en cada jugada.

Cuando el ordenador pide los datos del disparo se deben escribir dos números separados por un espacio en blanco y luego "enter". El primer número indica la inclinación que debe tomar la tortuga para dar en la diana. Cuando mira hacia arriba la tortuga tiene una inclinación de 0 grados. La orientación aumenta de 1 en 1 en el sentido de giro de las agujas del reloj. De modo que si se desea que mire exactamente hacia la derecha se debe dar una inclinación de 90 grados. El segundo número representa el número de pasos que se estima debe recorrer la torguga para dar en el blanco. Para hacer la estimación lo mejor es la experiencia; téngase, no obstante, en cuenta que la pantalla tiene 640 pasos de tortuga desde su extremo izquierdo al derecho.

Por ejemplo, si para dar en el blanco se estima que la tortuga debe tener una inclinación de 32 grados y que debe recorrer 208 pasos, entonces basta escribir, en la línea de datos:

32 208

y luego "enter". Se verá en pantalla el camino recorrido por la tortuga y la posición final (a menos que se salga de los límites de la pantalla). En la línea de datos aparecerá la puntuación obtenida en ese disparo. Pocos segundos después aparecerá también el número de puntos acumulados hasta ese momento del juego, y el número de jugadas.

En caso de dar en la diana se obtienen 35 puntos. Según la distancia sea mayor se van perdiendo puntos.

Al final de los 10 disparos se termina el juego. En pantalla aparece el número total de puntos y la puntuación obtenida de 0 a 10.

```
① to disparos
② setpal 0[2 0 1]
③ setpal 1[2 1 0]
④ inicio
⑤ label "regr
⑥ make "num :num+1
⑦ jugada
⑧ setsplit 1 ss
⑨ type [PUNTOS:] type :puntos
⑩ type "
⑪ type [JUGADAS:] type :num
   wait 250
⑫ if :num<10 [go "regr]
⑬ res
⑭ end
⑮ to inicio
   cs ct ts
⑯ repeat 15[type "  ]
   pr "DISPAROS
   repeat 15[type "  ]
   repeat 8[type "  ]
   repeat 4[pr []]
⑰ pr [* APARECERA UNA DIANA EN ALGUN
     LUGAR DE]
   type "  type "
   pr [LA PANTALLA]
   pr [* EN OTRO LUGAR APARECRA LA
     TORTUGA]
   type [* SE TRATA DE HACER DISPA
     ROS CON LA TOR]
   (type "  [TUGA DE MODO QUE SE ACER
     QUE TODO LO]) pr []
   type "  type "
   pr [POSIBLE A LA DIANA]
   pr [* PARA ELLO BASTA ESCRIBIR CU
     ANDO SE PI]
   type "  type "
   pr [DA LA ORIENTACION QUE DEBE
     TOMAR Y LA ]
   type "  type "
   pr [DISTANCIA QUE DEBE RECORRER]
   pr [* ESCRIBASE LA ORIENTACION (0
     A 359) Y]
   type "  type "
   pr [LA DISTANCIA SEPARADOS POR UN
     ESPACIO]
   make "puntos 0
⑱ make "num 0
   repeat 3[pr []]
   type [PULSAR ENTER]
   make "se rc
⑲ end
   to jugada
   setpc 1
⑳ cs fs
   diana
   sit
   setsplit 1 ss
㉑ setpal 2[2 2 0]
   type [ESCRIBIR LOS DATOS DEL DIS
     PARO:]
   make "intento r1
㉒ setpc 2 fs
㉓ sound [1 20 50]
   seth first :intento
㉔ fd first bf :intento
㉕ repeat 3[ht wait 40 st wait 15]
㉖ make "posfi list first tf first
     bf tf
㉗ ss
   make "disx (- first :pos first
     :posfi)
㉘ make "disy (- first bf :pos first
     bf :posfi)
   make "dx :disx*:disx
㉙ make "dy :disy*:disy
   make "dc :dx+:dy
㉚ asigp
   end
   to diana
   ht
   make "dx random 320
㉛ make "dy random 200
   make "az1 random 2
   make "az2 random 2
   if :az1=1 [make "dx -:dx]
   if :az2=1 [make "dy -:dy]
   if :dy<-184 [make "dy :dy+16]
㉜ make "pos list :dx :dy
   dot :pos
㉝ pu setpos :pos
   seth 0
   bk 10 pd lt 90 fd 10 rt 90
   repeat 4[fd 20 rt 90]
㉞ end
   to sit
   ht
   make "tx random 320
   make "ty random 200
   make "rx random 360
   make "az3 random 2
   make "az4 random 2
   if :az3=1 [make "tx -:tx]
㉟ if :az4=1 [make "ty -:ty]
   if :ty<-184 [make "ty :ty+16]
   make "tos list :tx :ty
   pu setpos :tos pd
   seth :rx st
   end
   to asigp
   if :dc>10000 [make "inc 0]
㊱ if and (or :dc<10000 :dc=10000)
     :dc>8000 [make "inc 2]
㊲ if and (or :dc<8000 :dc=8000)
     :dc>6000 [make "inc 4]
   if and (or :dc<6000 :dc=6000)
     :dc>4000 [make "inc 7]
   if and (or :dc<4000 :dc=4000)
     :dc>2000 [make "inc 9]
   if and (or :dc<2000 :dc=2000)
     :dc>1000 [make "inc 11]
   if and (or :dc<1000 :dc=1000)
     :dc>500 [make "inc 15]
```

```
if and (or :dc<500 :dc=500)
  :dc>200 [make "inc 18]
if and (or :dc<200 :dc=200)
  :dc>50 [make "inc 20]
if and (or :dc<50 :dc=50)
  :dc>10 [make "inc 23]
if and (or :dc<10 :dc=10)
  :dc>1 [make "inc 24]
if :dc=1 [make "inc 25]
if :dc=0 [make "inc 35]
type [PUNTOS DEL DISPARO:]
type :inc
```

```
make "puntos :puntos+:inc
wait 250
ct ss
end
to res
ts rt
type [PUNTOS:] pr :puntos
pr [] pr []
make "punt int(100*(:puntos/35))
type [PUNTUACION (0 A 10):]
pr :punt/100
end
```

COMENTARIOS

NOTA: Los números rodeados con un círculo se corresponden con los comentarios del final del programa.

① Es el procedimiento principal
② Pone color del fondo
③ Pone color a los graficos
④ Esta es la llamada al procedimiento inicio
⑤ Marca el lugar al que debe volver el control tras un "go"
⑥ Asigna a la variable "num" un nuevo valor
⑦ Llamada a procedimiento
⑧ Indica el numero de lineas de la pantalla mixta. Instala esta pantalla.
⑨ Escribe en pantalla sin retorno de carro
⑩ Escribe un espacio en blanco
⑪ Produce una pausa
⑫ Si el valor de la variable "num" es menor que 10 el control pasa al lugar marcado con "regr"
⑬ Llamada a procedimiento
⑭ Final del procedimiento
⑮ Borra los graficos. Borra el texto instala la pantalla de texto
⑯ Escribe 4 lineas en blanco
⑰ Instala el valor 0 en la variable "puntos"
⑱ Instala en la variable "se" el siguiente caracter que se escriba en el teclado
⑲ Selecciona la pluma 1 para los graficos
⑳ Ordena que la ventana mixta tenga una linea de texto. Selecciona la pantalla mixta
㉑ Almacena en la variable "intento" la lista de datos que se escriba a continuacion en el teclado
㉒ Emite un sonido
㉓ Hace mirar a la tortuga en la direccion absoluta dada por el primer elemento de la lista "intento"
㉔ Hace avanzar a la tortuga tantos pasos como indique el segundo elemento de la lista
㉕ La tortuga hace intermitencia
㉖ Almacena en la variable "posfi" las coordenadas de la tortuga tras el disparo
㉗ Resta el primer numero de la lista "posfi" del primer numero de la lista "pos" y guarda el resultado en la variable "disx"
㉘ Hace que "dx" almacene el cuadrado de "disx"
㉙ La variable "dc" almacena el cuadrado de la distancia entre tortuga y diana
㉚ Almacena en la variable "dx" un numero elegido al azar y comprendido entre 0 y 319
㉛ Para que la linea de datos no oculte a la diana, si su coordenada y es menor que 184, se aumenta en 16
㉜ Dibuja un punto en la posicion cuyas coordenadas almacena la variable "pos"
㉝ Dibuja el cuadrado que rodea a la diana
㉞ Si la variable "az3" vale 1, se toma negativo el valor de "tx"
㉟ Si el cuadrado de la distancia ("dc"), es mayor que 10000, se otorgan 0 puntos al disparo
㊱ Si "dc" es menor o igual que 10000 y mayor que 8000 se otorgan 2 puntos
㊲ Escribe el valor de la variable "inc" que son los puntos del ultimo disparo
㊳ Escribe el total de puntos conseguidos

El programa técnico del mes

SISTEMAS DE UNIDADES

Autor: Víctor J. Campo López

GENERALIDADES

La utilización de unidades de medida en varios sistemas y su conversión de unos a otros, origina en muchos casos, errores en los cálculos, a menos que se domine bien, el manejo de los factores de conversión. Estos errores, suponen un pequeño martirio para estudiantes y profesionales, que tratan a menudo con diferentes unidades.

De todos son conocidos los formularios, agendas, prontuarios, etc., usados comúnmente para salvar esta dificultad.

Tres son los sistemas adoptados y reconocidos universalmente, el Sistema CEGESIMAL (C.G.S.), el GIORGI, también conocido como MKS, y por último el sistema TECNICO.

El programa que proponemos, tiene cuatro opciones:

1.— GRAFICO
2.— CALCULO
3.— EJERCICIOS
4.— PARAR

— La primera opción, ofrece un gráfico con todas las unidades de los tres sistemas, relacionadas con unas reglas nemotécnicas, fáciles de recordar por su extremada sencillez. A la vista del cuadro y de acuerdo con las instrucciones que se ofrecen, en forma de textos móviles, pueden realizarse toda clase de conversiones, con suma facilidad.

La finalidad última de dicho gráfico, es la de poderse memorizar, de esta manera en cualquier momento, podremos hacer operaciones de cambio, sin tener que recurrir a ningun manual.

— La segunda opción, de tipo práctico, ofrece la posibilidad de realizar las operaciones de conversión, con el ordenador. El programa le permitirá elegir que tipo de unidad desea, y que cantidad, calculando las correspondientes a los otros dos sistemas.

— La opción tercera, es un complemento del gráfico, en el sentido de que le permite practicar las conversiones, mediante unos ejemplos que el propio ordenador le facilitará, donde él mismo asumirá el papel de profesor, indicándole sus aciertos y fallos (admitirá un márgen de error de una centésima en sus respuestas), así como el número de ejercicios realizados.

— Por último, la cuarta opción, permite detener el programa, de manera que puede acceder al listado, si así lo desea, e incluso introducir variaciones en el mismo.

En las tres opciones básicas, se utilizarán unicamente unidades de Masa, Fuerza, Trabajo y Potencia. Las unidades de Longitud, Tiempo, etc., se consideran tan sumamente fáciles de manejar en los tres sistemas, que suponemos son del dominio de todos.

DESCRIPCION DEL GRAFICO. OPCION 1

El gráfico propuesto se compone de tres triángulos, que representarán de menor a mayor, a:

1) Unidades de MASA
2) Unidades de FUERZA
3) Unidades de TRABAJO Y POTENCIA

Cada vértice de los triángulos, corresponde a un sistema de unidades. El superior es el C.G.S., el izquierdo el GIORGI o MKS y el derecho el TECNICO.

Las unidades correspondientes se sitúan junto a los vértices. En los lados, se han reflejado los factores de conversión entre unidades, que se utilizarán, de acuerdo con el sistema de signos, indicado en el propio gráfico (izquierda o arriba — multiplicar, derecha o abajo — dividir).

GRAFICO

Los factores de conversión resultan muy fáciles de recordar. Observe que todas las bases de los triángulos, tiene como factor <9.8>, esto indica que para pasar del sistema GEORGI al TECNICO, habrá que dividir por 9.8 y para pasar del TECNICO al GIORGI, multiplicar por 9.8, con independencia de que las unidades sean de masa, fuerza, trabajo o potencia.

Por tanto, recuerde que el factro 9.8, es común a las bases de los tres triángulos.

En los lados izquierdos, existen tres factores, el primero 10↑3 para el interior (masa), 10↑5 para el medio (fuerza) y 10↑7 para el triángulo exterior.

Para recordar estos números, observe que en el triangulo interior, se encuentra el gr.m y el Kg.m, su relación es evidente y Kg = 1000 gr, es decir 10↑3, y los otros tienen como factor, las siguientes potencias impares de 10 (5 y 7).

Conocida la relación anterior, la correspondiente a los lados derechos, resulta aún mas sencilla, estos factores son sencillamente el producto de los otros dos lados, en cada triángulo.

Como conclusión, podemos decir, que son solo dos los números a recordar: el 9.8, común a los tres ubicado en las bases y el 10^3, que es la relación entre el kilogramo y el gramo, conocido por todos.

Estos dos números, el convenio de signos y por supuesto conocer a que sistema corresponde cada unidad, nos permitirá, de forma sencilla, obtener rápidamente la conversión entre unidades.

Quizá le sirva la ayuda, recordar que las unidades menores, c rresponden al sistema C.G.S. y la mayores al TECNICO.

La opción 1, del programa, le dará una explicación más completa del gráfico, con ejemplos.

SIGNOS
x
:
x
:
C.G.S.
Ergio Ergio/s
1.— MASA
2.— FUERZA
3.— TRABAJO Y POTENCIA
3
dina
2
$9{,}8\times10^{7}$
gr.m
$10^{7}\ 10^{5}\ 10^{3}\ 1$
$9{,}8\times10^{5}$
$9{,}8\times10^{3}$
Kg.m
9,8
UTM
Nw
9,8
Kp
GIORGI O MKS
Julio
9,8
TECNICO
Jul/s = watio
Kgm
Kgm/s

```
10  '.....................................
15  '.
20  '.      SISTEMAS DE UNIDADES          .
30  '.....................................
32  '
33 CLS:GOSUB 8000: FOR K=1 TO 1000:NEXT K
35 DIM U$(3,3) :GOSUB 1500
36 DIM PRE$(20),R(20):GOSUB 2500
40 MODE 1:GOTO 9000
50 MODE 0
55 PRINT CHR$(24)
56 LOCATE 5,9: PRINT "          "
60  LOCATE 5,10: PRINT " F  I  N "
70 LOCATE 1,17:END
100 '  ..........  GRAFICO  ..........
102 INK 0,0 :INK 1,20:BORDER 1:CLS
105 MODE 2: RESTORE 160
110     FOR T=1 TO 3
120 READ X,Y:PLOT X,Y
130       FOR P=1 TO 3
140 READ X,Y: DRAW X,Y
150  NEXT P,T
160 DATA 152,104,485,104,320,368,152,104
170 DATA 208,136,429,136,320,304,208,136
180 DATA 280,184,357,184,320,240,280,184
190       FOR P=1 TO 23
200 READ X,Y,Z$
210  LOCATE X,Y: PRINT Z$
220  NEXT P
230  PRINT CHR$(24):LOCATE 38,1: PRINT " C.G.S. "
235  LOCATE 62,20: PRINT " TECNICO "
240 LOCATE 6,20: PRINT " GIORGI O MKS ": PRINT CHR$(24)
250 DATA 35,2,"Ergio   Ergio/s"
260 DATA 23,20,"Julio",16,21,"Jul/s=watio"
270 DATA 60,21,"Kgm   Kgm/s"
280 DATA 40,4,"3",40,8,"2",40,12,"1"
290 DATA 39,6,"dina",25,18,"Nw",53,18,"Kp"
300 DATA 39,18,"9.8",39,20,"9.8",32,15,"Kg.m   9.8   UTM"
310 DATA 39,10,"gr.m",27,11,"7",32,11,"5",37,11,"3"
320 DATA 25,12,"10",30,12,"10",35,12,"10"
330 DATA 50,9,"9.8*10^7",47,11,"9.8*10^5",44,13,"9.8*10^3"
340  LOCATE 2,4: PRINT "×  ————————"+CHR$(243)+" :"
350  LOCATE 2,10: PRINT "×  "+ CHR$(242)+"————————  ."
360   LOCATE 2,5: PRINT CHR$(240)+"                "+CHR$(148)
370 LOCATE 2,6: PRINT CHR$(148)+"                "+CHR$(148)
380 LOCATE 2,7: PRINT CHR$(148)+"                "+CHR$(148)
390 LOCATE 2,8: PRINT CHR$(148)+"                "+CHR$(148)
400 LOCATE 2,9: PRINT CHR$(148)+"                "+CHR$(241)
410 LOCATE 5,2: PRINT "SIGNOS"
420 LOCATE 58,1: PRINT "1 - MASA"
430 LOCATE 58,3: PRINT "2 - FUERZA"
440 LOCATE 58,5: PRINT "3 - TRABAJO Y POTENCIA"
450   FOR P=1 TO 3500: NEXT P
480 GOSUB 9500
490 PRINT CHR$(24)
500 STOP
1000 '  ..............  C A L C U L O ..................
1010 MODE 1
1015 LOCATE 3,1: PRINT "SELECCIONE TIPO DE UNIDADES"
```

```
1020     FOR K=1 TO 3: LOCATE 10,K*3
1030 PRINT K;" - ";U$(K,0): NEXT K
1040 Z$=INKEY$:IF Z$="" THEN  GOTO 1040
1050 T1=VAL(Z$)
1060 IF T1<1 OR T1>3 THEN  GOTO 1040
1062 CLS: PRINT "   QUE UNIDAD DE ";CHR$(24);"  ";U$(T1,0);"  ";CHR$(24): PRINT
"DESEA CONVERTIR ?"
1063 PRINT : PRINT
1064      FOR K=1 TO 3: PRINT TAB(10);K;" ) ";U$(T1,K): PRINT : NEXT K
1070 Z$=INKEY$:IF Z$="" THEN  GOTO 1070
1080 T2=VAL(Z$)
1085 IF T2<1 OR T2>3 THEN  GOTO 1070
1090  PRINT "QUE CANTIDAD DE ";U$(T1,T2);
1095 INPUT C
1096  LOCATE 1,17
1100 ON T1 GOTO 1110,1200,1300
1110 IF T2=1 THEN  PRINT C;" gr.m=";c/1000;" Kgm = ";c/1000*9.8;" UTM"
1120 IF T2=2 THEN  PRINT C;  " Kg.m=";1000*c ;" gr.m = ";c/9.8;" UTM"
1130 IF T2=3 THEN  PRINT C;  " UTM = ";C*9.8;" Kg.m = ";c*9.8*1000;" gr.m"
1140  GOSUB 9980:GOTO 9000
1200 IF t2=1  THEN  PRINT C;" Dinas = ";c/10^5;" Nw = ";c/10^5/9.8 ;" Kp"
1210 IF t2=2  THEN PRINT c;  "Nw = ";c/9.8;" Kp = ";c*10^5;" Dinas"
1220 IF t2=3 THEN  PRINT c;" Kp = ";c*9.8;" Nw = ";c*9.8*10^5;" Dinas"
1230  GOSUB 9980:GOTO 9000
1300 IF t2=1 THEN  PRINT c;" Ergios = ";c/10^7;" Julios = ";c/10^7/9.8;" Kgm"
1310 IF t2=2 THEN  PRINT c;" Julios = ";c*10^7;" Ergios = ";c/9.8;  " Kgm"
1320 IF t2=3 THEN  PRINT c;" Kgm = ";c*9.8;" Julios = ";c*9.8*10^7;" Ergios"
1340 GOSUB 9980: GOTO 9000
1500 ' ..................  MATRIZ DE UNIDADES ..................................
1505 RESTORE 1530
1510 FOR F=0  TO 3 : FOR C=0 TO 3
1520 READ U$(F,C)
1525 NEXT C,F
1530 DATA " ","C.G.S.","GIORGI","TECNICO"
1535 DATA "MASA","gr.m","Kg.m","UTM"
1540 DATA "FUERZA","dina","Nw","Kp"
1550 DATA "TRABAJO","Ergio","Jul","Kgm"
1560 RETURN
2000 ' ..................  EJERCICIOS ..........................................
2005 CON=1
2010 k=INT(RND*18)+1
2020 CLS
2030  PRINT "PREGUNTA No ";CON: LOCATE 22,1: PRINT "ACIERTOS - ";AC: LOCATE 22,2
: PRINT "FALLOS   - ";FA
2035 PRINT : PRINT "-----------------------------------------------"
2040  LOCATE 1,10:PRINT PRE$(K): PRINT : PRINT
2050 INPUT "RESPUESTA ";RES
2060 IF (ABS(R(K)-RES))<=0.01 THEN AC=AC+1: LOCATE 14,20: PRINT CHR$(24);" CORR
ECTO ";CHR$(24):  GOTO 2100
2070 fa=fa+1: LOCATE 14,20: PRINT "FALLO ": PRINT : PRINT : PRINT "LA RESPUESTA
CORRECTA ES ";R(K)
2100 CON=CON+1
2110  LOCATE 1,25: PRINT "PRESIONE C-Continuar, M-Menu"
2130 Z$=INKEY$:IF Z$="" THEN 2130
2140 IF Z$="C" OR Z$="c" THEN 2010
2150 IF Z$="M" OR z$="m" THEN 9000
2160 PRINT CHR$(7): GOTO 2130
2499 STOP
2500 RESTORE 2520
2510      FOR K=1 TO 18:READ PRE$(K),R(K): NEXT K
2520 DATA "Cuantos gr.m son 3 Kg.m?",3000
```

```
2522 DATA "Cuantos gr.m son 10 UTM?",9800
2524 DATA "Cuantos Kg.m son 7234 gr.m?",7.234
2526 DATA "Cuantos Kg.m son 5 UTM?",49
2528 DATA "Cuantas UTM son 9.8 Kg.m?",1
2530 DATA "Cuantas UTM son 14000 gr.m?",1.429
2532 DATA "Cuantas Dinas tiene un NW?",100000
2534 DATA "Cuantas Dinas son 2 Kp?",1960000
2536 DATA "Cuantos Nw son 9.8 Kp?",96.04
2538 DATA "Cuantos Nw son 10^7 Dinas?",100
2540 DATA "cuantos Kp son 10 Nw?",98
2542 DATA "cuantos Kp son 134567 Dinas?",.1373
2544 DATA "Cuantos Ergios son .002 Julios?",20000
2546 DATA "Cuantos Ergios son .004 Kgm?",392000
2548 DATA "Cuantos Julios son 100 Kgm?",980
2550 DATA "Cuantos Julios son 4*10^9 Ergios?",400
2552 DATA "Cuantos Kgm son 100 Julios?",10.204
2554 DATA "Cuantos Kgm son 4*10^7 Ergios?",4.0816
2560 RETURN
8000 '. PRESENTACION .....
8010 PLOT 0,0
8020 FOR k=1 TO 300 STEP 10
8030 DRAWR 0,400-k: DRAWR 600-k,0
8040 DRAWR 0,-400+k+5: DRAWR -600+k+5,0
8050 NEXT k
8060 FOR K=2 TO 24 STEP 2:INK 1,K
8070 LOCATE 16,13:PRINT "UNIDADES"
8080 FOR I=1 TO 400:NEXT I
8090 NEXT K
8095 RETURN
9000 '.......... MENU ..............
9002 CLS: PRINT CHR$(24)
9005  LOCATE 16,4: PRINT "       "
9010  LOCATE 16,5: PRINT " MENU "
9020 LOCATE 16,6: PRINT "_______"
9030 LOCATE 12,7: PRINT "                "
9040  LOCATE 12,8: PRINT " 1 - GRAFICO   "
9050  LOCATE 12,9: PRINT " 2 - CALCULO   "
9060  LOCATE 12,10: PRINT " 3 - EJERCICIOS"
9070  LOCATE 12,11: PRINT " 4 - PARAR     "
9080 PRINT CHR$(24): LOCATE 11,22: PRINT "SELECCIONE OPCION"
9090 Z$= INKEY$: IF Z$="" THEN  9090
9100  X=VAL (Z$)
9105 IF X<1 OR X>4 THEN 9090
9110 ON X GOTO 100,1000,2000,50
9500 '..... TEXTOS PARA VENTANA MOVIL...
9505 '
9515  PRINT CHR$(24)
9520  LOCATE 5,22: PRINT "------------------------------------------------------------
-------------------------"
9524 RESTORE 9530
9525 FOR T=1 TO 13: READ T$:GOSUB 9900:GOSUB 9960: NEXT T
9530  DATA "                    ESTE GRAFICO CONTIENE TODAS LAS INSTRUCCIONES PAR
A LA CONVERSION ENTRE UNIDADES DE MASA, FUERZA, TRABAJO Y POTENCIA, EN LOS TRES
SISTEMAS   C.G.S  -  GIORGI  -  TECNICO  "
9540 DATA "                    EL TRIANGULO 1 QUE ES EL MENOR, REPRESENTA
  LAS UNIDADES DE <MASA>: El gramo-masa ( gr.m ) que pertenece al sistema C.G.
S., el kilogramo-masa (Kg.m ) del GIORGI y la Unidad Tecnica de Masa ( UTM
) del TECNICO "
9560 DATA "                    EL 2 CONTIENE LAS UNIDADES DE
FUERZA>: La dina perteneciente al C.G.S., el Newton ( Nw ) del GIORGI
y el Kilopondio ( Kp ) del TECNICO.   "
```

```
9570 DATA "                                   EL 3 CONTIENE LAS UNIDADES DE <T
RABAJO Y POTENCIA>: El Ergio del C.G.S., el Julio ( Jul ) del GIORGI
y el Kilogrametro ( Kgm ) del TECNICO "
9580 DATA "                                   Las unidades de potencia son las mism
as que las de trabajo, pero partidas por segundo, comun a los tres sistemas. El
 Jul/s. se denomina watio (w)."
9590 DATA "                                   LA RELACION NUMERICA ENTRE UNIDADES ES MUY
SENCILLA DE RECORDAR. OBSERVE, QUE EN TODAS LAS BASES DE LOS TRIANGULOS, EXISTE
 UN NUMERO, 9.8, IGUAL PARA LOS TRES. "
9600 DATA "                                   AHORA FIJESE EN LOS SIGNOS: IZQUIERDA Y ARR
IBA * (multiplicar), DERECHA Y ABAJO / (dividir) "
9610 DATA "                                   POR LO TANTO, PARA PASAR DE Kp a Nw (D
rcha. a Izq.), hay que multiplicar por <9.8> (1 Kp=9.8 Nw). DE IGUAL MANERA PAR
A PASAR DE Kgm a Julios, o de UTM a Kg.m. EVIDENTEMENTE EN SENTIDO CONTRARIO HAY
 QUE DIVIDIR. "
9620 DATA "                                   LOS LADOS DE LA IZQUIERDA DE LOS TRIANGULOS
, SE IDENTIFICAN CON UN <10 ELEVADO A 3> EN EL INTERIOR (MASA), <10 ^ 5> EN EL
MEDIO (FUERZA) Y <10 ^ 7> EN EL EXTERIOR (TRABAJO Y POTENCIA). "
9630 DATA "                                   AHORA LOS SIGNOS A CONSIDERAR SON ARRIBA (*
) Y ABAJO (/). POR EJEMPLO: PARA PASAR DE Nw a dinas, HAY QUE MULTIPLICAR
POR <10 elevado a 5>. DE Ergios a Julios <dividir por 10^7>, ETC. "
9640 DATA "                                   POR ULTIMO EL LADO DERECHO SE IDENTIFICA EN
 CADA TRIANGULO, POR EL PRODUCTO DE LOS OTROS DOS LADOS, CON EL CRITERIO DE SIGN
OS ANTERIOR. POR EJEMPLO DE gr.m a UTM, hay que dividir por <9.8 y por 10^3>
 "
9650 DATA "                                   RECUERDE 9.8 ABAJO. 10 ↑ 3, 5, 7 A LA IZQUI
ERDA Y EL PRODUCTO A LA DERECHA. MEMORICE LAS UNIDADES Y CON EL CRITERIO DE SIGN
OS Y NO HABRA MAS CONVERSIONES DIFICILES. "
9660 DATA "                                   UNA ULTIMA RECOMENDACION, PARA APRENDER BIE
N EL CUADRO, TRATE DE REPRODUCIRLO DE MEMORIA, HASTA QUE CONSIGA COMPLETAR TODOS
 SUS DATOS, LUEGO REALICE CONVERSIONES MENTALMENTE. LA OPCION <3> DEL MENU LE AY
UDARA. "
9890 RETURN
9900 ' ..... ESCRITURA EN VENTANA .....
9905    FOR K=1 TO LEN(T$)
9910  LOCATE 1,24: PRINT MID$(T$,K,79)
9920  SOUND 1,3,10 :    FOR Y=1 TO 60: NEXT Y
9930 NEXT K
9950 RETURN
9960  LOCATE 1,24: PRINT "                  POR FAVOR, PRESIONE UNA TECLA PARA CONTINUAR
                        "
9970 Z$=INKEY$ : IF Z$="" THEN  GOTO 9970 ELSE RETURN
9980 LOCATE 1,24: PRINT "   PRESIONE UNA TECLA PARA CONTINUAR"
9990 Z$=INKEY$ : IF Z$="" THEN  GOTO 9970 ELSE RETURN
```

Nota: Todos los símbolos ∧ que aparecen en el listado deben sustituirse por ↑

LA "BIBLIA" DEL

¡ya está a la venta!

AMSTRAD

FASCICULO 3

295 PTAS.

DOMINARLO ES UN JUEGO

¡ya está a la venta!