AMSTRAD EDUCATIVO

N.º 6 - 295 Ptas.

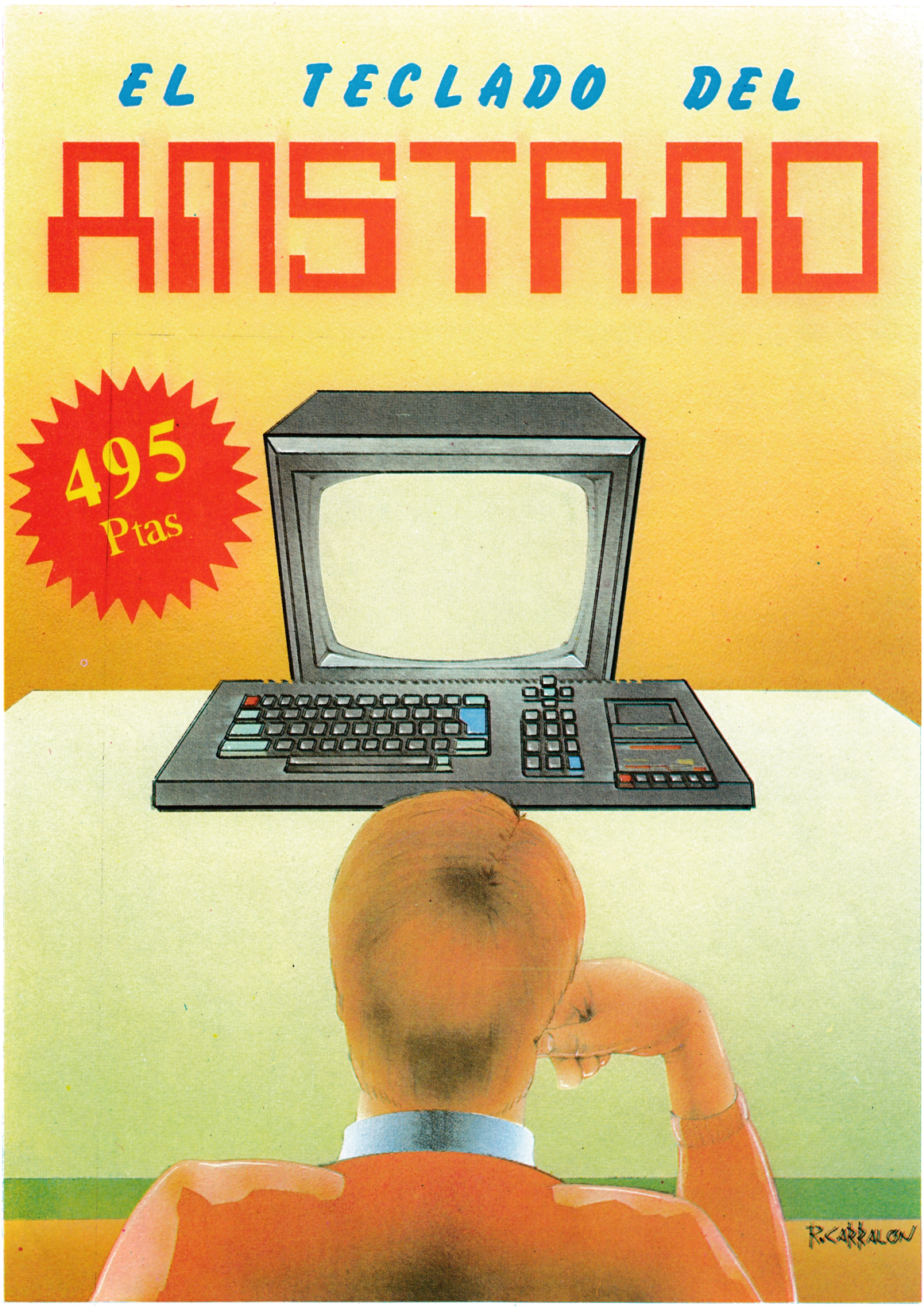
EL AMSTRAD Y EL CP/M

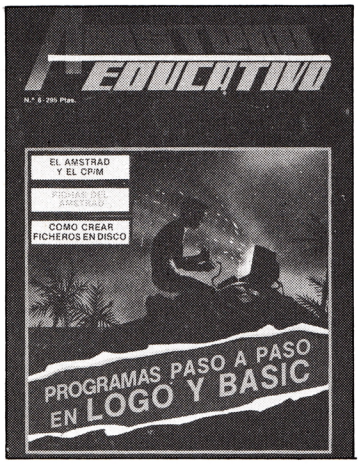
FICHAS DEL AMSTRAD

COMO CREAR FICHEROS EN DISCO

PROGRAMAS PASO A PASO EN LOGO Y BASIC

¡ya está a la venta!





Editorial

Mes a mes pretendemos mostrarte la forma de manipular tu ordenador, estudiando las diferentes partes y posibilidades con amplitud y detalle. Ejemplos prácticos y programas, paso a paso, contemplando este propósito.

Intentamos utilizar un lenguaje sencillo, con sentencias breves, apoyadas muchas veces con esquemas y gráficos para una mejor comprensión.

Por supuesto, te surgirán dudas, todo el equipo estamos a vuestra disposición para resolverlas. Enviádnoslas, así como los programas que vayáis creando. En próximos números podréis verlas publicadas.

Sumario

AMSTRAD EDUCATIVO

Edita: Grupo Editorial G.T.S., S.A. Bailén, 20-1.º Izda. 28005 Madrid. Telf.: 266 66 01-02. **Director:** Antonio Bellido. **Colaboran** en este número: Juan M. Pintor, Víctor J. Campo López. **Maquetación y Diseño:** Gorrindo. **Secretaria de Redacción:** Mercedes Jamart. **Publicidad:** Dpto. propio. Bailén, 20-1.º Izda. 28005 Madrid. Teléf.: 266 66 01-02. **Composición e Impresión:** 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

ERRORES POSIBLES HASTA EL MOMENTO

por A. Bellido

Desde un punto de vista lógico, y hasta el momento, los errores que el operador ha podido cometer o provocar son:

Primer tipo

Motivo: Equivocación al teclear una orden, dando lugar a una palabra que CP/M no admite y consiguientemente no entiende.

Mensaje de error: En pantalla aparecerá la palabra que pretendía ser una orden seguida un símbolo? y, finalmente, a renglón seguido, la impronta de la unidad de disco por defecto.

Ejemplo: Al escribir una línea de orden utilizado la orden TYPE ha sucedido lo que sigue:

```
A > TIPE LIBROS.SEC
TIP?
A >
```

Con lo cual el sistema avisa que no comprende la orden y queda a nivel operador, esperando instrucciones.

Medidas a tomar: Escribir de nuevo —y correctamente— la línea de orden completa.

Segundo tipo

Motivo: El fichero referenciado en la línea de orden no existe en el disco —en el área de usuaria activa— instalado en la unidad indicada.

Mensaje de error: Si la orden es DIR, REN o ERA, en pantalla aparecerá una expresión inglesa en el sentido de que tal fichero no existe en el disco de la unidad ordenada. Estos mensajes serán similares a uno de éstos: FILE NOT FOUND (fichero no localizado), NO LIFE (no hay ese fichero) o NOT FOUND (no localizado).

Con el resto de las órdenes —SAVE y TIPE— el mensaje de error considera en mostrar a renglón seguido, el nombre del fichero referenciado seguido del símbolo?

Ejemplo:

1.º Dentro de este tipo de error, la causa más usual se deriva de teclear mal el nombre del fichero, lo cual, en todo caso, no evita el hecho de que tal fichero no esté catalogado.

Las posibilidades en función de la orden implicada son:

```
I.—
A > ERA LOBROS. SEL
NO FILE
A >
II.—
A > TYPE LOBROS. SEL
LOBROS. SEL?
A >
```

En el supuesto, claro está, de que tal fichero no exista.

Medidas a tomar: Determinar si el nombre del fichero está escrito correctamente y si lo está, averiguar en qué disco y área de usuario está situado, para, finalmente, repetir la línea de orden correcta.

Tercer tipo

Motivo: Se pretende incluir en un disco un fichero con la misma referencia de otro ya ctalogado.

Mensaje de error: En pantalla aparecerá un mensaje en el sentido de que tal fichero existe: FILE EXISTS.

Ejemplo: Admitamos que conviene, por alguna razón, cambiar el nombre del fichero EJEMPLO. SIE por el de EJEMPLO. SEI, y este último ya existe en el directorio.

La orden oportuna sería:

```
A > REN EJEMPLO. SEI =
EJEMPLO. SIE
```

Al pulsar <cr> para hacerla efectiva, el sistema detecta que el nuevo nombre a asignar el fichero ya existe y nos enviaría el mensaje:

```
FILE EXISTS
A >
```

Quedando a nivel operador.

Medidas a tomar: Asignar un nombre distinto al fichero nuevo que se pretende incluir en el disco.

Cuarto tipo

Motivo: Fallo en el sistema físico. Cuando CP/M intenta cumplir la orden dada por el operador se encuentra con la imposibilidad de ejecutarla porque la unidad de disco especificada no está debidamente conectada o no hay disco instalado en ella o éste deteriorado.

Mensaje de error: Lo más usual será encontrarnos con un mensaje en el sentido apropiado, seguido de tres opciones:

Retry, Ignore o Cancel?

Indicando al operador que si desea intentar la ejecución de la orden nuevamente, deberá pulsar R y si, por el contrario, quiere que el sistema la ignore o cancele, deberá pulsar I o C.

No obstante lo dicho, este tipo de errores son interceptados por el BDOS al llevar a efecto la parte del proceso que le corresponde en la ejecución de la orden en cuestión, siendo un mensaje:

Donde x es el nombre (A, B, C, etc.) de la unidad de disco en que se produce el error y *tipo de error* representa una de estas dos alternativas.

BAD SECTOR
o
SELECT

La primera posibilidad se deriva bien de un efecto de lectura o escritura del propio dispositivo o bien de un disco muy gastado o deteriorado, y la segunda procede de intentar trabajar con una unidad de disco que no está en línea.

Ejemplo: Supongamos que una configuración de ordenador sólo incorpora una unidad de disco que, por ende, será la A. Si en estas condiciones diéramos la orden:

A > B:

CP/M detectaría la incongruencia y daría el mensaje:

BDOS ERR ON B: SELECT

Medidas a tomar: En el caso de un error que implique uno del tipo «SELECT», el sistema se reinicializa.

En el caso de un error del tipo BAD SECTOR, y si esta situación se repite con frecuencia, se debería revisar la electrónica de la unidad de disco y/o los discos que lo producen.

En todo caso, y para salir de un error que contenga el mensaje BAD SECTOR, lo mejor será sacar el disco e inicializar el sistema otra vez, ya que se corre el peligro de alterar el contenido del disco. También, si lo permite la máquina, un C es adecuado.

Mensaje de error: En el caso propuesto, el mensaje sería:

Errores, ejercicios

1.° ¿Qué tipo de error motivó este mensaje?: BDOS ERROR ON B: SELECT.
Respuesta: El operador olvidó añadir los dos puntos a B para cambiar la unidad por defecto.

2.° ¿Qué tipo de error motivó este mensaje?: B?
Respuesta: El operador olvidó añadir los dos puntos a B para cambiar la unidad por defecto.

3.° ¿Qué tipo de error motivó este mensaje?: NO LIFE.
Respuesta: Se invoca un fichero que no existe en el disco seleccionado área de usuario activo. Este mensaje procede de una de las órdenes: ERA, REN y DIR.

4.° Tras detectar un error y emitir el mensaje oportuno, ¿a qué nivel se queda CP/M?
Respuesta: A nivel operador o, dicho de otro modo, nivel sistema.

En estos momentos ya sabemos que los programas encargados de llevar a efecto las órdenes permanentes están incluidos en el CCP y que éste, el BIOS y el BDOS están almacenados en las dos pistas más externas de cualquier disco del sistema. El contenido de estas dos pistas se transmite íntegro, en el momento de la inicialización, a la RAM, ubicándose en ésta de acuerdo con lo visto páginas atrás.

Por otra parte, el disco del sistema original que acompaña a cada ordenador que trabaje bajo el control de CP/M incluye unos programas, suministrados igualmente por Digital Research, grabados en pistas distintas a las dos citadas.

Estos programas están destinados a ejecutar ciertas órdenes aceptadas por CP/M de la misma forma que las permanentes estudiadas en el epígrafe anterior, pero con una diferencia notable: Mientras los programas de las órdenes permanentes residen en la RAM —en la zona del CCP— aquellos otros son cargados en la memoria interna —zona del TPA— justo en el momento en que la orden es dada. Por esta razón son conocidos como «programas transitorios» y las órdenes correspondientes «órdenes transitorias».

Todos y cada uno de estos programas transitorios son ficheros del tipo COM (por command) y sus nombres completos son:

- MOVCPM. COM
- SYSGEN. COM
- STAT. COM
- PIP. COM
- SUBMIT. COM → XSUB
- ED. COM
- DUMP. COM
- ASM. COM
- LOAD. COM
- DDT. COM

Estas órdenes transitorias pueden ir precedidas del nombre de una unidad de disco y dos puntos —por ejemplo, B:—, con lo cual su programa transitorio será cargado en la RAM desde el disco instalado en la unidad indicada, sin que por ello cambie la unidad de disco por defecto. Ejemplo:

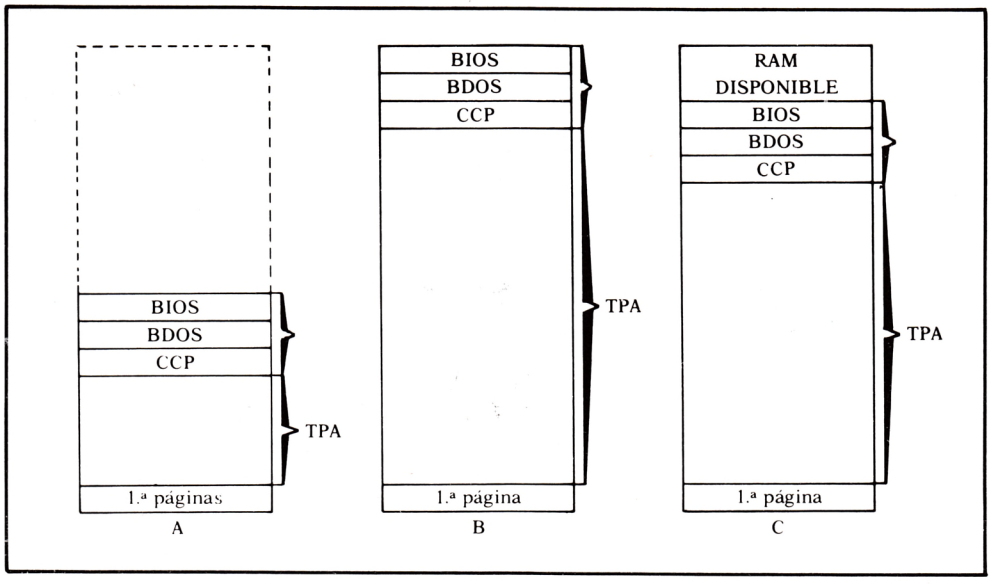
A > B:MOVCPM

Al pulsar <cr>, CP/M buscará el programa MOVCPM-COM en el disco instalado en la unidad B, lo cargará en la TPA y volverá a nivel operador con la impronta correspondiente a la actual unidad de disco por defecto que en el ejemplo es A >.

Este conjunto de programas son útiles para mantener y manejar los ficheros contenidos en disco y son conocidos como «utilidades de mantenimiento». El término inglés equivalente es *house keeping.*

En los epígrafes siguientes se estudian las sintaxis requeridas por las órdenes transitorias que ponen en acción estos programas y los distintos efectos que pueden producir.

MOVCPM (MOVE CP/M: MOVER CP/M)



Función asignada: Mover el bloque BIOS-BDOS-CCP a una zona de la RAM distinta a la que tenga asignada a la inicialización.

Comentarios: Tras una orden MOVCPM se produce una transferencia al TPA de una copia de las pistas reservadas del disco del sistema que esté en uso, un cambio en la ubicación del DOS y, según los parámetros usados en la orden, se reinicializa el sistema ajustado a la nueva estructura de la RAM o, por el contrario, la situación se mantiene a la espera de órdenes.

La primera de las dos últimas alternativas se sigue cuando se pretende una modificación eventual de la estructura de la RAM, de tal forma que ante una inicialización del sistema todo volverá a sus lugares originales, ya que nada ha cambiado en el disco del sistema.

Mediante la segunda posibilidad que ofrece MOVCPM se puede grabar en disco la modificación realizada. Para ello CP/M ofrecerá dos procedimientos derivados de dos órdenes: SYSGEN y SAVE.

SAVE ya se ha estudiado y el propio MOVCPM indica el número de páginas.

SYSGEN se estudia en el próximo epígrafe.

Formas de obtenerla y consecuencias: Instalar en la unidad de disco por defecto un disco que contenga el programa MOVCPM:. COM y utilizar una de estas opciones:

1.ª posibilidad: Teclear MOVCPM seguido de <cr>. Por este procedimiento se ordena un desplazamiento del bloque BIOS-BDOS-CCP hacia la parte más alta del RAM, obteniéndose el máximo de área posible para el TPA. Tras esta orden el DOS queda en condiciones de ser usado de acuerdo con su nueva ubicación.

2.ª posibilidad: Teclear MOVCPM, un espacio, un número comprendido entre 20 y la capacidad media en K bytes, total, actual de la RAM. Al pulsar <cr> el bloque BIOS-BDOS-CCP se instala en la memoria, admitiendo el número dado en la orden como si fuera el «techo» de la RAM. Si este número es mayor que el tamaño real de la RAM disponible, la reacción del computador es impredecible, siendo lo más probable que el sistema se quede bloqueado («colgado»). Tras esta orden, el sistema queda en condiciones de ser usado de acuerdo con su nueva ubicación.

3.ª posibilidad: Teclear MOVCPM, un espacio, un número en las condiciones expuestas anteriormente y un asterisco. Al pulsar, <cr> el resultado es similar al obtenido en el caso anterior, pero con la diferencia de que se ofrece la posibilidad de grabar en disco la nueva ubicación asignada al bloque BIOS-BDOS-CCP.

4.ª posibilidad: Teclear MOVCPM, un espacio, seguido de dos asteriscos. Al pulsar <cr>, el resultado es indéntico al ofrecido en la opción anterior, pero con el bloque BIOS-BDOS-CCP ubicado en la zona de memoria más alta ofrecida por la RAM del computador.

Ejemplos:

1.º Disponemos de un computador con una capacidad de 64 bytes de memoria interna y de un disco del sistema que asigna al bloque BIOS-BDOS-CCP una ubicación correspondiente a un tamaño máximo de 32 K bytes de RAM. ¿Qué orden CP/M deberíamos emplear para reubicar el DOS de forma que el TPA sea lo mayor posible?
El planteamiento de este ejemplo corresponde, esquemáticamente, a las figuras A y B presentadas al comienzo de este epígrafe, siendo nuestro objetivo desplazar el blo-

que BIOS-BDOS-CCP representado en la A hasta la posición que ocupa la B.
La orden que cumpliría con lo propuesto es:
A > MOVCPM
Seguido de <cr>, siempre, claro está, que el programa MOVCPM. COM esté contenido en el disco instalado en la unidad por defecto, y éste sea el A. Si admitiéramos que el disco referido estuviera en la unidad B y la unidad por defecto fuera la A, nuestra orden, bajo estos supuestos, debe ser:
A>B:MOVCPM
En todo caso, y tras ejecutarse la orden, la impronta de la unidad de disco por defecto —A> en el ejemplo— nos indicará que todo sucedió correctamente, y espera nuevas instrucciones.
La orden, tal y como ha sido dada en este ejemplo, sólo modifica la ubicación del DOS en la RAM, con lo cual una nueva inicialización del sistema lo llevaría a su zona original.
Esta orden es útil para modificar eventualmente la estructura interna de la RAM.

2.° Las premisas de este ejemplo son idénticas a las del anterior, y el efecto que se persigue, el mismo, pero ¿qué orden CP/M debe ser dada si queremos tener la posibilidad de grabar en disco la modificación realizada?
A > MOVCPM ☆☆
Al pulsar <cr> se nos ofrecerá la doble posibilidad de grabar el resultado mediante SYSGEN o SAVE —de las cuales hablaremos más tarde— con un mensaje de este tipo:
READY FOR SYSGEN OR "SAVE n CP20. COM"
Siendo n el número de páginas a salvar.
Seleccionar SAVE provoca la creación de un fichero COM conteniendo el DOS propiamente dicho. Util para programadores en código máquina.

3.° Disponemos de un computador con una capacidad de 64 K bytes de RAM y de un disco del sistema ajustado a esta memoria.
¿Qué orden CP/M deberíamos emplear para bajar el bloque BIOS-BDOS-CCP lo suficiente para liberar 2 K bytes en el «techo» de la RAM?
La figura B representa el estado inicial de la RAM y la C el que se pretende conseguir.
En esta ocasión el disco está instalado en la unidad por defecto y éste es el B; por tanto, la orden es:
B > MOVCP 62
Al pulsar >cr<, el programa MOVCPM. COM sería cargado en la TPA y ejecuatría, a continuación, la orden dada.
Este ejemplo, como el anterior, sólo implica una modificación de la ubicación del DOS.

4.° Las premisas de este ejemplo son idénticas a las de anterior y el efecto que se persigue, el mismo, pero, ¿qué orden debería ser dada si queremos tener la posibilidad de grabar en un disco la modificación realizada?
A > MOVCPM 62 ☆
Al pulsar <cr> CP/M nos ofrecerá la doble posibilidad de grabar mediante la orden SYSGEN o la orden SAVE, de las cuales hablaremos más tarde.

Ejercicios con MOVCPM

1.° ¿qué orden CP/M adapta el DOS (BIOS/BDOS/CCP) a la máxima dimensión de la RAM?
Respuesta: MOVCPM.

2.° ¿Cómo debe proceder el operador para volver a la situación anterior a MOVCPM?
Respuesta: Inicializar el sistema.

3.° ¿Qué orden CP/M ajustará el DOS temporalmente a una dimensión de memoria de 48 K?
Respuesta: MOVCPM 48

4.° ¿Qué efecto produce la orden MOVCPM?
Respuesta: Posiciona el DOS en lo más alto de la RAM y reinicializa el sistema en estas condiciones.

5.° ¿Qué efecto produce la orden MOVCPM 48?
Respuesta: Posiciona el DOS a partir de 48 K hacia abajo en la memoria RAM y reinicializa el sistema en estas condiciones.

6.° ¿Qué efecto produce la orden MOVCPM 48°?
Respuesta: Esta respuesta es igual a la anterior, pero no reinicializa el sistema, sino que permanece a la espera de respuesta a esta cuestión:
READY FOR "SYSGEN" OR "SAVE p cpm k. COM"
(preparado para "SYSGEN" o "GRABACION p cpm de k. COM")
Donde p es el número de páginas a «salvar» si se elige la opción «SAVE» y k es el número de *k bytes* del sistema (en nuestro caso, 48). Si se opta por esta alternativa, se creará un fichero conteniendo el DOS pero del tipo COM. Si se sigue el camino de SYSGEN, se grabará la nueva situación en las pistas del sistema.

7.° ¿Qué efecto produce la orden MOVCPM ☆☆?
Respuesta: El descrito en el ejercicio anterior, pero adaptado a la máxima capacidad de la RAM.

8.° Interpretar las dos líneas siguientes:
A > B:MOVCPM
MOVCPM?
A >
Respuesta: El disco instalado en la unidad B no contiene el programa transitorio MOVCPM. COM.

9.° Dar la orden oportuna para reconfigurar la estructura de la RAM con el DOS a 48 K, de forma que el operador pueda usarlo inmediata y temporalmente.
Respuesta: MOVCPM 48.

10.° Dar la orden oportuna para reconfigurar la estructura de la RAM con el DOS a 48 K, de forma que todo quede dispuesto para grabar esta nueva configuración en el disco.
Respuesta: MOVCPM 48 ☆.

COMO CREAR FICHEROS EN DISCO

por A. Bellido

LOCALIZACION

Supongamos que una lista desordenada, contiene los siguientes elementos: GARCIA, FERNANDEZ, ANTON, HERNANZ, DESOTO, ENTERRIA, BILBAO y COLL.

Y que necesitamos desarrollar un programa que nos permita localizar uno cualquiera de ellos, determinado por teclado.

Este podría ser, básicamente, un listado adecuado:

```
10 CLS
20 DIM a$(7):c=0
30 FOR x=0 TO 7:READ a$(x):NEXT x
40 DATA ,GARCIA,FERNADEZ,ANTON,HERNANZ,DESOTO,ENTERRIA,BILBAO,COLL
50 INPUT "Apellido a localizar en mayusculas ";ap$
60 IF c>7 THEN 80
70 IF ap$=a$(c) THEN 90 ELSE c=c+1:GOTO 60
80 PRINT "El apellido ";ap$;" no existe en la lista ":END
90 PRINT "El apellido ";ap$;" es el ";c+1;"de la lista":END
```

Como se puede apreciar este sistema de búsqueda exige la comparación de la cadena a buscar con todos los elementos de la lista, uno tras otro, hasta encontrar alguna coincidencia; si no hay tal, el proceso continua hasta el final.

Deliberadamente se ha hecho referencia al elemento C + 1 de la lista, en tanto en cuanto el programador no debe perder el punto de vista del usuario del programa, y este sería el caso si en lugar de poner C + 1 se colocara C, con lo cual, y dado que el subíndice mínimo de la matriz A$( ) es el ∅, puede darse el caso de que la línea 90 diera lugar a una impresión del tipo:

''El apellido ENTERRIA es el ∅ de la lista''.

Que es posible aparezca como incoherente a algunas personas.

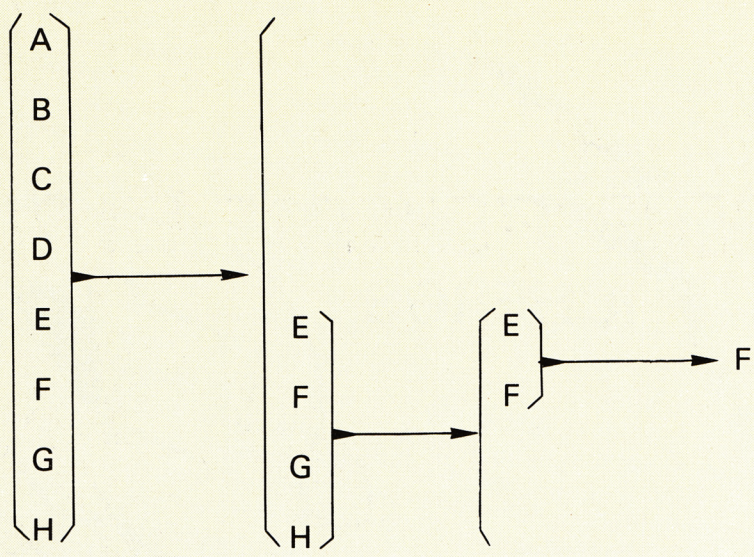
Pero, si partimos de la hipótesis de que la lista está clasificada, la localización de elementos será mucho más efectiva.

El método que vamos a seguir en este caso, recibe el nombre de:

Búsqueda dicotómica.

La lógica que se sigue para encontrar un elemento de la lista ordenada mediante este método consiste en averiguar si dicho elemento está contenido en la mitad inicial de la lista o en la final. Determinado esto, se procede a repetir este paso con respecto a los dos cuartos de la lista de la mitad donde ya sabemos que está. Este proceso se aplica nuevamente con los octavos correspondientes y, así, hasta ahorquillar y determinar el elemento buscado. Para fijar la idea, veamos un ejemplo gráficamente:

se trata de localizar el elemento F dentro de la lista A,



Un algoritmo que cumple estas exigencias es:

```
10 DIM A$(7)
20 FOR X=0 TO 7:READ A$(X):NEXT X:C=X-1
30 DATA ANTON,BILBAO,COLL,DESOTO,ENTERRIA,FERNANDEZ,GARCIA,HERNANZ
40 '*************************************************************

50 REM *** BUSQUEDA DICOTOMICA ***
60 INPUT "APELLIDO A LOCALIZAR EN MAYUSCULAS ";AP$
70 PE=0:UE=C
80 IF AP$<A$(PE) OR AP$>A$(UE) THEN 150
90 IF AP$=A$(PE) THEN S=PE:GOTO 160
100 IF AP$=A$(UE) THEN S=UE:GOTO 160
110 MITAD=INT((PE+UE)/2)
120 IF AP$=A$(MITAD) THEN S=MITAD:GOTO 160
130 IF AP$>A$(MITAD) THEN PE=MITAD ELSE UE=MITAD
140 IF UE-PE>1 THEN 110
150 PRINT "EL APELLIDO ";AP$;" NO EXISTE EN LA LISTA":END
160 PRINT "EL APELLIDO ";AP$;" ES EL ";S+1;" DE LA LISTA":END
```

Las tres líneas iniciales tienen por misión rellenar los primeros elementos, del ∅ al 7, con una serie de apellidos clasificados por orden alfabético.

En la línea 60, se permite la introducción por teclado del apellido a localizar.

En la línea 70, comienza la búsqueda dicotómica, valorando las variables PE (por primer elemento) y UE (por último elemento). Tanto PE como UE representan números de subíndice de los elementos de la matriz A$, que, al principio, deben ser O y C (C = 7).

A continuación, en la línea 80, se confirma que el apellido a buscar está dentro del campo alfabético comprendido entre el primer elemento de la matriz y el último.

En las líneas 90 y 100, comprobamos si el apellido es igual al primer o al último elemento de la lista. La variable S contendrá el subíndice del elemento buscado, siempre, claro está, que exista en la lista.

La línea 110 de la orden de dividir la lista en los sucesivos segmentos mediante la semisuma de los subíndices mayor y menor, y en 120 se comprueba que el elemento buscado coincide o no, con el elemento A$ (MITAD).

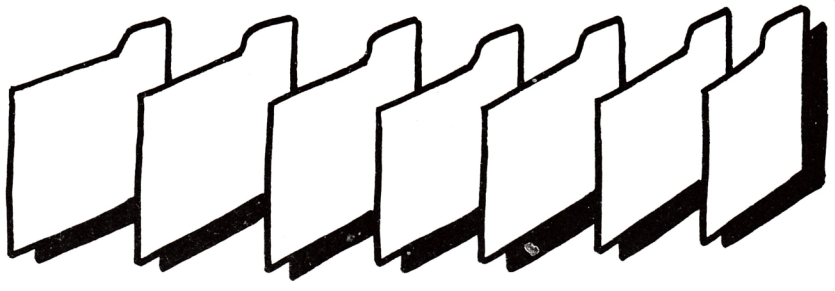
La siguiente línea, clave del procedimiento, hace que el primer subíndice de la lista pase a ser el primero de la sublista actual que inicia el nuevo campo alfabético o, por el contrario, el último subíndice de la lista pase a ser el último del nuevo campo alfabético.

Todo ello dependiendo de que AP$ sea mayor o no que el elemento de la vista de subíndice MITAD.

El resto del programa se explica por sí solo.

FICHEROS

por A. Bellido



Conceptos generales

La mayoría de las palabras utilizadas entre las personas que, por oficio o afición, se expresan en términos informáticos son voces que en el lenguaje cotidiano tienen un significado parecido y cuya estructura mental las hacen fáciles de asimilar a la función que cumplen en el terreno de la Informática.

Tal es el caso de la expresión **fichero.**

Convencionalmente _____ >, un fichero es un cajón con cartulinas dispuestas vertical y ordenadamente que contienen, o pueden contener, algún tipo de información relativa a un objeto común.

Estas cartulinas son denominadas **fichas.**

La información de las fichas está distribuida en un cierto número de apartados sobre los que se escriben los datos oportunos.

Estos mismos conceptos aplicados a los ficheros en disco, nos lleva a sustituir **ficha** por **registro** y a los **apartados** de cada ficha por **campo.** Esto ya se vió con detalle en el epígrafe dedicado a FICHERO. CONCEPTO.

Vemaos ahora varias clasificaciones de los ficheros informáticos según diferentes puntos de vista.

1.º En función de sus datos:
- Ficheros maestros: Contienen datos de carácter permanente con muy baja tasa de modificación.
- Ficheros de movimiento: Contiene datos susceptibles de ser modificados con frecuencia.
- Ficheros históricos: Contienen datos cronológicamente archivados.

2.º En función del soporte:
- Ficheros en cinta magnética: Sus datos están grabados en un soporte de este estilo.
- Ficheros en disco: Sus datos están grabados en este soporte.
- Ficheros en tarjetas perforadas: Sus datos están contenidos en este tipo de soporte.

3.º En función de su organización:
- Fichero secuencial: El acceso a sus datos es secuencial. Para añadir, modificar o borrar un registro hay que leer los anteriores o posteriores.
- Fichero aleatorio: El acceso a sus datos es directo. Este tipo permite una localización más rápida de un registro. A estos ficheros también se les conoce por fichero de acceso directo.
- Fichero secuencial-indexado: Este tipo es una combinación de los dos anteriores, ya que, por

una parte, los registros tienen una orden secuencial pero por otra, utilizan unos registros de índice que permiten el acceso directo.

4.° En función de su situación:
- — Ficheros en línea (on line): Se mantienen en situación de disponibles para el computador constantemente.
- — Ficheros fuera de línea (off line): Están archivados hasta que son necesarios.
- — Ficheros actualizados: Todos sus datos se mantienen al día.
- — Ficheros desactualizados: Existen datos que no corresponden con la última información llegada.

Según esto, nuestro curso se va a dirigir a los ficheros en disco, de movimiento y cuya organización sea bien por secuenciales, bien aleatorias.

Para concluir este epígrafe, un consejo.

En lo que sigue, el lector no debiera perder de vista en ningún momento la doble perspectiva que todo programador ha de contemplar constantemente: La suya propia como diseñador del programa y la del posible usuario del mismo. Por esta razón, en algunos pasajes del libro, se hace referencia bien a la labor del programador bien a las acciones que lógica y consecuentemente llevará a efecto el usuario al utilizar el programa.

En suma, un programa es un trabajo que, una vez concluido, está destinado a ser manejado previsiblemente por personas distintas a su autor.

Opciones de uso de un fichero

Todo fichero, para que sea realmente operativo, debe ofrecer las siguientes alternativas de uso:

1.°.— Introducir nuevas fichas (ALTAS).
2.°.— Eliminar fichas (BAJAS).
3.°.— Modificar el contenido de fichas (PUESTA AL DIA).
4.°.— Consultar el contenido de fichas (CONSULTA).
5.°.— Listar fichas (LISTADOS).

Cada una de estas opciones exige seguir un determinado proceso para ser llevada a efecto. Hagamos un análisis de los pasos que debemos dar en cada caso:

1.°.— **Para introducir nuevas fichas (ALTAS)**
1. — Abrir el fichero.
2. — Coger una ficha en blanco.
3. — Rellenarla.
4. — Introducirla en el fichero.
5. — Si hay alguna ficha más a rellenar, se repite el proceso desde el paso 2.
6. — Si no hay más fichas a rellenar se cierra el fichero.

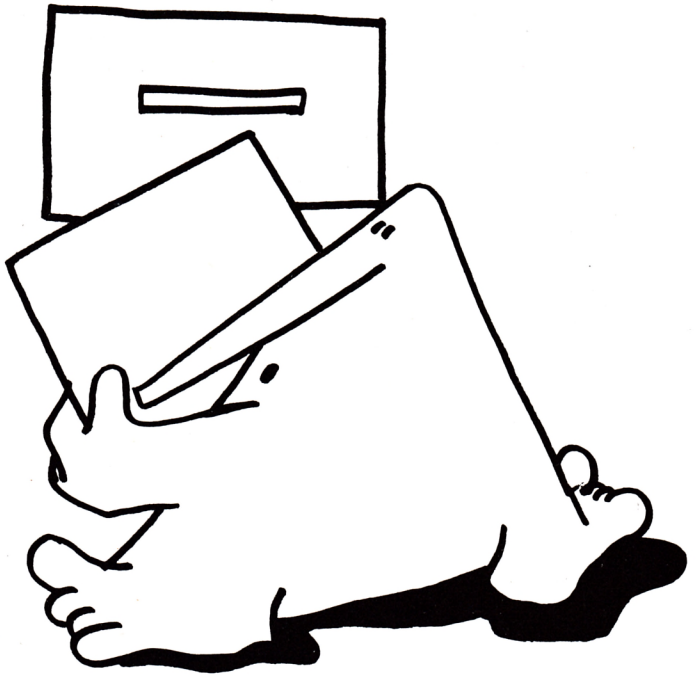
2.°.— **Para eliminar fichas (BAJAS)**
1. — Abrir el fichero.
2. — Localizar el fichero.
3. — Extraerla del fichero.

4. — Si hay alguna ficha más a eliminar, se repite el proceso desde el paso 2.
5. — Si no hay más fichas a eliminar se cierra el fichero.

3.°.— **Para modificar el contenido de fichas (PUESTA AL DIA).**
1. — Abrir el fichero.
2. — Localizar la ficha a eliminar.
3. — Extraer del fichero.
4. — Efectuar la modificación.
5. — Introducirla en el fichero.
6. — Si hay alguna ficha más a modificar se repite el proceso desde el paso 2.
7. — Si no hay más fichas a modificar se cierra el fichero.

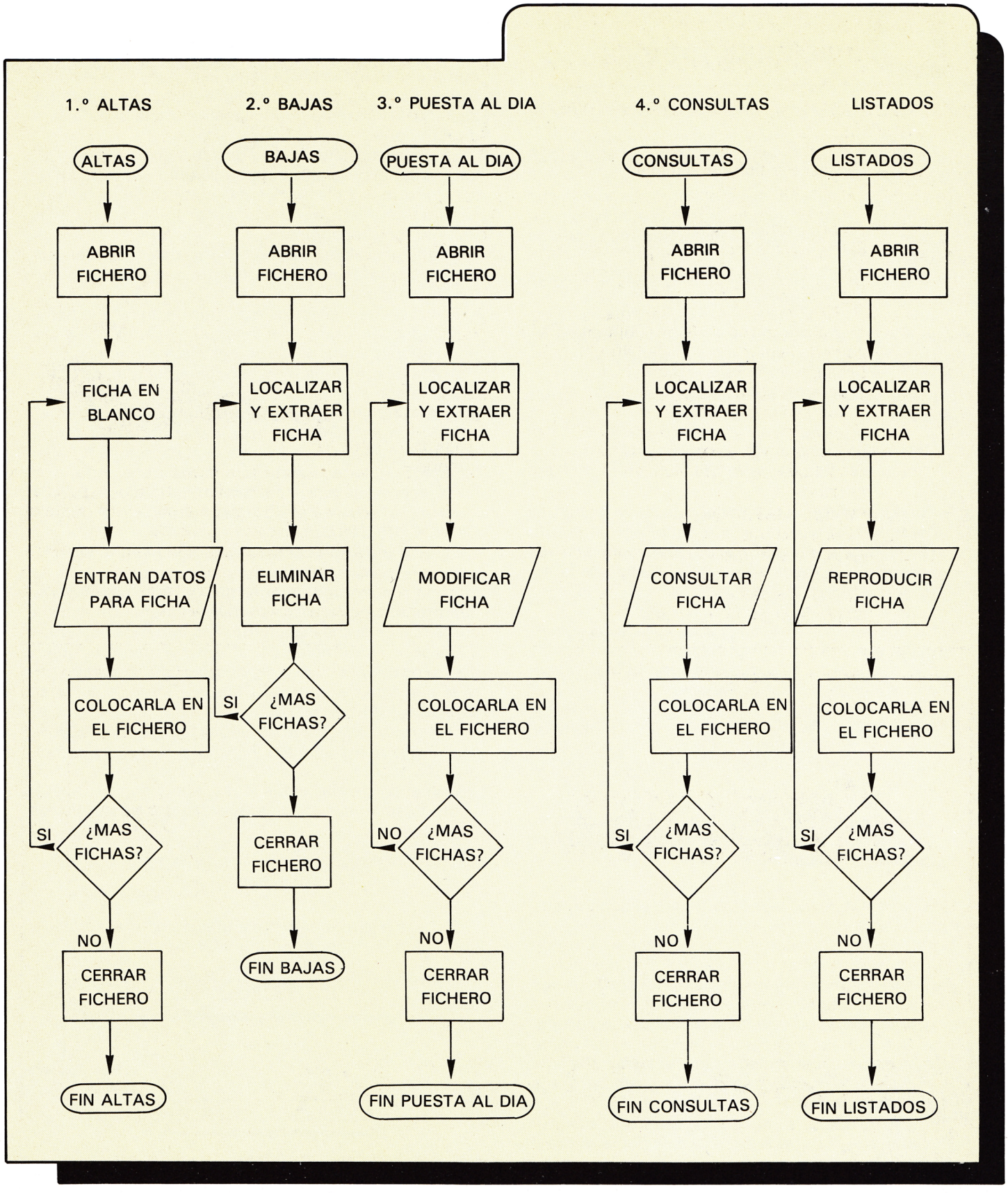
4.°.— **Consultar el contenido de fichas (CONSULTAS)**
1. — Abrir el fichero.
2. — Localizar la ficha a consultar.
3. — Extraer del fichero.
4. — Efectuar la consulta.
5. — Si hay alguna ficha más a consultar se repite el proceso desde el paso 2.
6. — Si no hay más fichas a consultar se cierra el fichero.

5.°.— **Listar fichas (LISTADOS)**
1. — Abrir el fichero.
2. — Localizar la primera ficha a partir de la cual se inicia el listado.
3. — Extraer del fichero.
4. — Reproducir su contenido.



5. — Introducirla en el fichero.
6. — Si no es la última ficha del listado pedido, se localiza la siguiente ficha y se repite el proceso desde el paso 2.
7. — Si es la última ficha del listado se cierra el fichero.

Veamos ahora, una representación esquemática de los procesos descritos anteriormente.



Hasta aquí un análisis de las opciones de uso de un fichero y los procesos que se deben seguir para ponerlas en práctica.

Volveremos con frecuencia a los esquemas anteriores.

Decisiones previas

Es aconsejable que antes de iniciar este epígrafe, el lector repase los destacados "DISCOS" y "FICHEROS ORGANIZACION".

Con toda probabilidad, el lector habrá observado que hasta aquí se ha dado por supuesto que el fichero existía. En otras palabras, que el fichero se había creado con anterioridad.

Es el caso que, para poder manipular un fichero, se debe empezar por confecciole, darle forma...

Olvidándonos de los ficheros convencionales y centrando nuestra atención en los ficheros en disco, recordamos que su creación pasa por asignarles un nombre.

El **nombre del fichero** está sometido a las siguientes normas:

1.° **Debe** estar formado por una cadena de hasta ocho caracteres. Por ejemplo: SELLOS.

2.° **Puede** añadirse una segunda cadena de hasta tres caracteres separada de la anterior por un punto. Por ejemplo: SELLOS. ESP. Este apéndice del nombre suele recibir el apelativo de **tipo de fichero.**

3.° Ciertos caracteres no se pueden utilizar como parte del nombre o del tipo del fichero, para confirmar este punto, hay que consultar el Manual del ordenador que esté en servicio.

6.° Nada impide dar nombre a un fichero mediante una variable alfanúmerica a la que se le haya asignado previamente la cadena correspondiente. Por ejemplo: A$ = "SELLOS ESP"

7.° El nombre del fichero no debe coincidir con ninguno de los que actualmente contenga el disco destinado a recibirlo.

Una vez establecido un nombre para el fichero, queda por decidir el tipo de acceso a los registros.

Los dos modos fundamentales de acceder a un determinado registro son el secuencial y el directo.

Para dejar claro la diferencia existente entre ambos, supongamos un fichero convencional cuyas fichas están perfectamente ordenadas y con algún tipo de indicativo que sobresale de las mismas. En estas condiciones, conocer el contenido de una determinada ficha sólo exige seleccionarla visualmente y desplazar la mano para extraerla.

En otras palabras, se accedió a ella directamente.

También podría darse el caso de que por la forma física del fichero o por su organización —o desorganización— la localización de una ficha exigiera investigar a partir de la primera, continuar con la segunda y así sucesivamente hasta encontrar la deseada.

En este caso se ha seguido un proceso secuencial de acceso a la ficha.

Bien, con los ficheros en disco sucede algo similar: En los secuenciales, la búsqueda de un registro se inicia en el primero y a partir de aquí, uno tras otro, se comprueban todos hasta llegar al objetivo. En los aleatorios, cada registro tiene asignado un número en base al cual el sistema puede localizar cada uno de ellos directamente.

Estas formas alternativas de acceder a los registros, va a exigir al programador una distinta concepción de los algoritmos a diseñar para la confección de las rutinas que permitan un funcionamiento adecuado a los ficheros correspondientes.

La diferencia básica entre ficheros de acceso secuencial y acceso directo a los registros, estriba en la forma de definir estos.

En los secuenciales, los registros de un mismo fichero, pueden tener cualquier longitud, separándose y distinguiéndose unos de otros por ciertas marcas.

Antes de proseguir, hay que remarcar que los ficheros en disco contienen exclusivamente caracteres, o más exactamente, los números binarios equivalentes a los códigos ASCII de esos caracteres. Por tanto, cuando aquí hagamos referencia al caracter "A", por ejemplo, lo que realmente estaremos manejando, será el correspondiente número binario del código ASCII de la letra "A" que es el 65. (Ver epígrafe "INFORMACION. CONCEPTO).

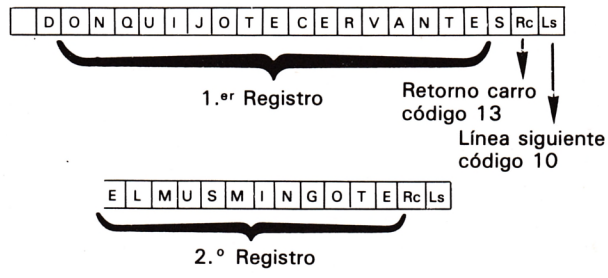
En este orden de cosas, los códigos ASCII 13 y 10 están reservados para dos funciones complementarias en toda transmisión de información. Así, cuando tras el envío de ciertos datos se quiere ordenar un salto al comienzo de la línea siguiente, nos veremos obligados a enviar un "retorno de carro" (RC) al margen izquierdo y un salto a la "línea siguiente" (LF). El código para el "retorno de carro" (RC) es el 13 y el de la "línea siguiente" (LF) es el 10.

Dicho esto, es fácil aceptar que las marcas de fin de registro sean los códigos 13 y 10 o lo que es lo mismo, RC y LF.

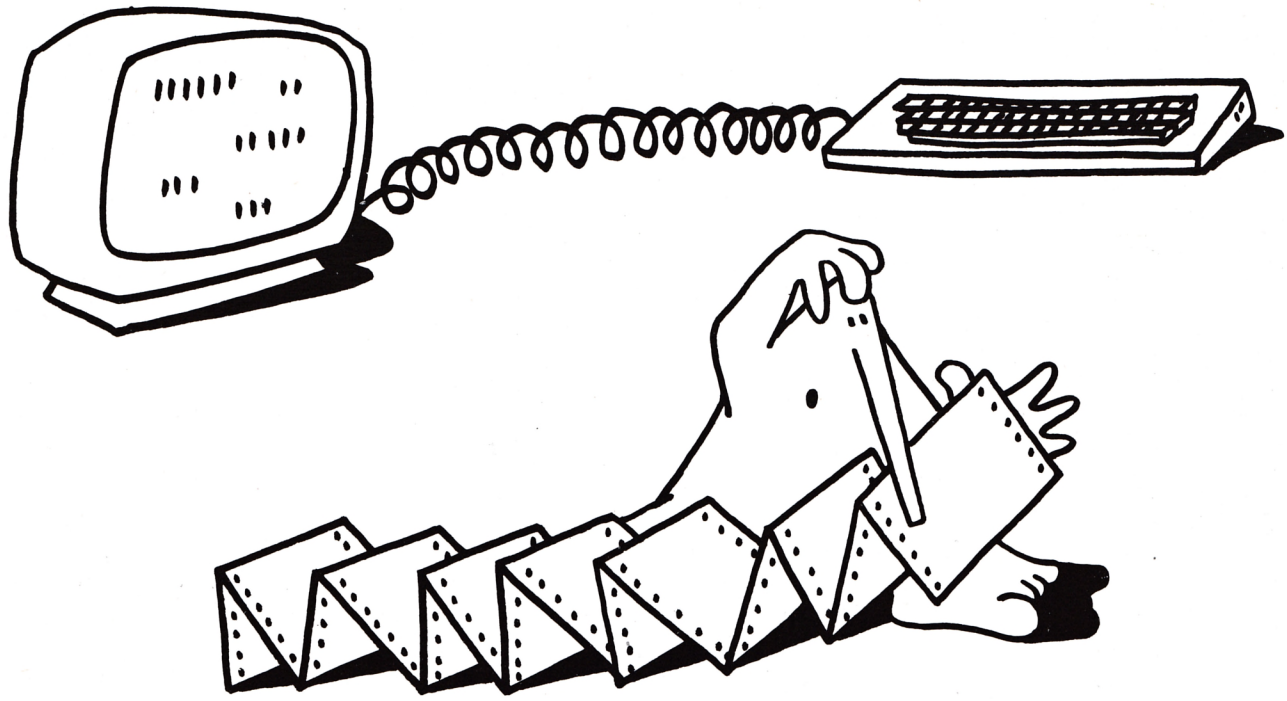
Con el fin de fijar estas ideas, supongamos que un fichero secuencial está compuesto por dos registros tales como:

DON QUIJOTE: CERVANTES
EL MUS: MINGOTE

En base a esto, una representación esquemática del contenido del fichero en cuestión podría ser:



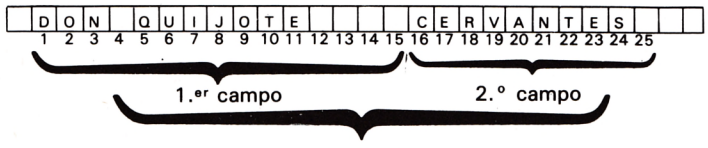
De aquí es fácil concluir que en un fichero secuencial los registros no tienen por qué tener la misma longitud entre ellos ya que existen unas marcas (RC y LF) que sirven de indicadores de fin de registro.

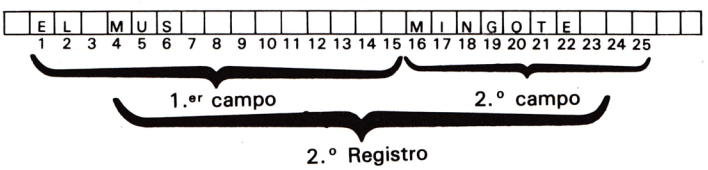


Por el contrario, en los ficheros de acceso directo o aleatorios, la longitud de los registros debe estar predefinida y subdividida entre los campos, sin que existan ningún tipo de marca para separar unos registros de otros ni en los campos entre sí, es el propio sistema quien, por medio de un contador de caracteres, sabrá cuando empieza un registro y un campo, y cuando acaban. Para ver esto con claridad, admitamos que queremos introducir los dos registros del ejemplo anterior en un fichero aleatorio.

Para empezar, tenemos que admitir una longitud para todos los registros del fichero, y subdividir esta en tantas longitudes como campos conforman sus registros. Vamos a dar, los registros una longitud de 25 caracteres distribuidos en 15 caracteres para el primer campo y 10 para el segundo. Esto es todo lo que el sistema ''sabe'', pero es suficiente, ya que a partir de un prefijado lugar del disco (Taladro del índice. Ver figura 1) puede llevar la cuenta de los caracteres y por tanto de los campos y por tanto de los registros del fichero.

Una representación esquemática del contenido real del fichero aleatorio en disco podría ser:



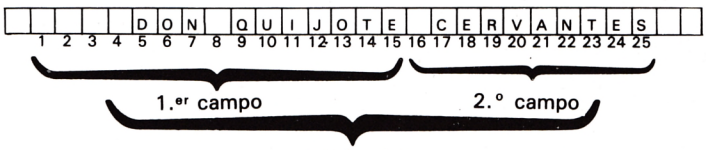


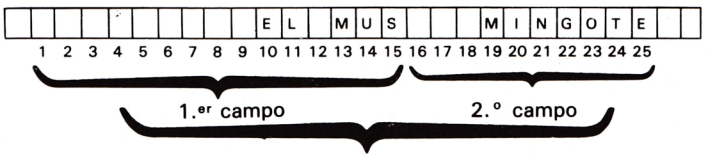
Es fácil de observar, a diferencia de lo que sucede en los secuenciales, que los caracteres no ocupados por los datos en los campos quedan rellenos por espacios.

De lo expuesto hasta el momento, podemos deducir que con los ficheros secuenciales hay un mejor aprovechamiento del disco ya se ocupa justo el espacio que se necesita dentro de él, mientras que en los aleatorios los datos más cortos que los campos, dejan espacios sin ocupar físicamente. Por otra parte, el acceso a los registros de un fichero aleatorio es mucho más rápido que en los secuenciales.

Hecho este comentario, volvamos a los ficheros aleatorios.

Si reflexionamos sobre el contenido del último esquema, aún se puede plantear la duda sobre la forma de ajustar los textos a introducir en los campos. Observe que otra forma de representación, igualmente válida es:





La única diferencia entre este esquema y el anterior, estriba en la forma de introducir los textos en sus campos: En el primer caso los caracteres se instalan de izquierda a derecha (ajuste a la izquierda) en sus respectivos campos y en el segundo caso, los caracteres se ajustan a la derecha.

Concluyendo, si a lo expuesto sobre los dos modos de acceder a los registros se añade que las instrucciones a utilizar en el momento de escribir las sentencias BASIC oportunas son distintas, queda justificado el por qué de la necesidad de decidir el tipo de fichero a emplear: Secuencial o aleatorio.

INPUT | INPUT "texto"; variable | ENTRADA

INTERPRETACION

Produce una interrupción en la ejecución de un programa permitiendo, de este modo, la entrada de información a través del teclado.

POSIBILIDADES

Cuando un programador introduce un comando INPUT, está provocando una parada en el programa para, de esta forma, dar un contenido a la variable.

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

Si el "texto" no existe, sólo el *prompt* (1) típico del ordenador aparecerá en pantalla indicando que está a la espera de información. En este caso, el dato que se introduzca será asignado a la variable.

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

FORMA DE TECLEAR LA INSTRUCCION

Teclear **INPUT,** el texto entre comillas (''), **punto y coma** (;) **y el nombre de la variable,** seguido de **ENTER,** para que aparezca **el texto** en pantalla hasta que se introduzca por teclado **el valor de la variable,** seguido de ENTER.

Teclear **INPUT y el nombre de la variable,** seguido de **ENTER,** para que la pantalla quede **en blanco** hasta que se introduzca por teclado **el valor de la variable.**

EJEMPLO:

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

SOLUCION:

PROGRAMA	COMENTARIOS
1> INPUT "¿Número de pasos?"; N	Con las dos primeras líneas provocaremos, al ejecutar el programa, dos interrupciones para asignar valores a las variables **N** y **P.**
2∅ INPUT "Valor de un paso?"; P	
3∅ PRINT "El importe de su conferencia es";	Con la tercera línea, ordenamos la impresión de texto que figura tras el comando PRINT.
4∅ PRINT N ☆ P	Con la cuarta línea, ordenamos la impresión de resultado de multiplicar el contenido actual de las variables N y P.

Cuando ejecutamos este programa, tecleando RUN y ENTER, en la pantalla aparecer escrito **¿Número de pasos?** y se parará a la espera del contenido de la variable N.

Si suponemos que el número de pasos ha sido **25,** tecleamos este número y, a continuación, ENTER, para indicar al ordenador que hemos concluido, aparecerá escrito en la pantalla; **¿Valor de un paso?** y volverá a pararse a la espera del contenido de la variable P.

Si asignamos a esta variable el valor **1∅, lo cual implica teclear 1∅** seguido de ENTER aparecerá en la pantalla **El importe de su conferencia es 25∅.**

Observe que el resultado de **N ☆ P** se imprime justo a continuación del texto de la línea 3∅, debido a que éste concluye en el programa con un punto y coma (;).

Escriba un programa que, una vez ejecutado, le pida el nombre de un escritor famoso; a continuación, le pida el título de una de sus obras y, finalmente, imprima en la pantalla ambas cosas.

SOLUCION:

PROGRAMA	COMENTARIOS
1∅ INPUT "¿Nombre del escritor?"; e∅	En las líneas 1∅ y 2∅ hemos definido el nombre de dos variables de caracteres.
2∅ INPUT "Título de una obra suya? "; t$	
3∅ PRINT e$	En las líneas 3∅ y 4∅ hemos ordenado la impresión, de las variables de caracteres definidas anteriormente, una debajo de la otra.
4∅ PRINT t$	

INPUT | INPUT "texto"; variable | ENTRADA

EJERCICIOS:

1. Escribir programas para resolver problemas referidos al propio entorno en los que intervengan las operaciones aritméticas:
   - 1.1. Costes:
     - 1.1.1. Calcular el valor de N unidades de una mercancía sabiendo el precio P de una unidad.
     - 1.1.2. Calcular el coste total de N unidades de una mercancía, al precio P por unidad y de M unidades de otra, al precio Q por unidad.
     - 1.1.3. Calcular la cantidad a devolver cuando se pagan X ptas. por la compra de N unidades de una mercancía, al precio P por unidad, y de M unidades de otra, al precio Q por unidad. Hallar el valor de cada mercancía, el valor total de ambas y la cantidad a devolver.
   - 1.2. Ahorro:
     - 1.2.1. Calcular el dinero que le sobra a una persona si tiene X y gasta Y.
     - 1.2.2. Calcular el dinero que le sobra a una persona si tiene X y gasta Y en diversiones y Z en libros (*).
     - 1.2.3. Calcular el dinero que ahorra una persona en M meses, si cada mes gana S y gasta V en vivienda; X en alimentación; Y en vestuario, y Z en transportes y diversiones.
   - 1.3. Areas:
     - 1.3.1. Hallar áreas de polígonos y figuras circulares.
     - 1.3.2. Hallar áreas laterales y totales de poliedros y cuerpos redondos.
     - 1.3.3. Hallar la base, la altura, el lado o la apotema de un polígono conociendo su área y los otros datos (*).
   - 1.4. Volúmenes:
     - 1.4.1. Hallar volúmenes de poliedros y cuerpos redondos.
     - 1.4.2. Hallar una dimensión de un poliedro o cuerpo redondo conocidos el volumen y las otras dimensiones (*).
   - 1.5. Móviles:
     - 1.5.1. Hallar la velocidad V de un móvil que recorre el espacio E en el tiempo T.
     - 1.5.2. Hallar el espacio E que recorre o el tiempo T que tarda un móvil conociendo los otros datos (*).
2. Escribir programas en los que intervengan variables alfanuméricas referidos a las diferentes materias de estudio.
   - 2.1. Lengua y Literatura:
     - 2.1.1. Vocabulario:
       Escribir el programa que, ejecutado, imprima en pantalla nombres, adjetivos y verbos relacionados con la casa (*).
     - 2.1.2. Gramática:
       Escribir el programa que, ejecutado, imprima en pantalla:
       — las clases de palabras que pueden formar el sintagma nominal;
     - 2.1.3. Literatura:
       Escribir el programa que, ejecutado, imprima en pantalla una relación de obras literarias clasificadas por autores (*).
   - 2.2. Geografía e Historia:
     - 2.2.1. Geografía:
       Escribir el programa que, ejecutado, imprima en pantalla una relación de accidente costeros clasificados por los mares donde están (*).
     - 2.2.2. Historia:
       Escribir el programa que, ejecutado, imprima en pantalla una relación de personaje históricos y sus principales hazañas (*).
   - 2.3. Ciencias Naturales:
     - 2.3.1. Escribir el programa que, ejecutado, imprima en pantalla una relación de órganos de cuerpo humano clasificados por aparatos (*).
     - 2.3.2. Escribir el programa que, ejecutado, imprima en pantalla una relación de vertebrados clasificados (*).
3. Escribir programas en los que intervengan combinadas variables numéricas y de caracteres que una vez ejecutados, sirvan para:
   - 3.1. Calcular la edad aproximada de una persona sabiendo el año en que nació y el actual.
   - 3.2. Calcular los días transcurridos desde el principio del curso escolar y los días que faltan para que termine (*).
   - 3.3. Calcular las horas de clase a lo largo de todo el curso escolar (*).
4. Idear sus propios programas utilizando los comandos conocidos: PRINT, LET e INPUT.

NOTA: Es importante que conserve todos estos programas y los compare con las del capítulo siguiente.

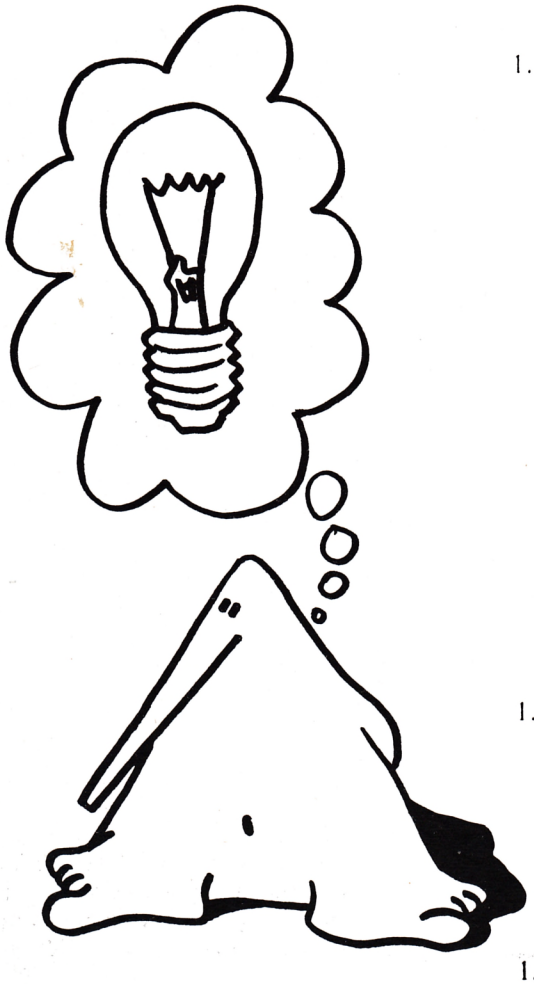
SOLUCIONES:

1.1.1.

```
 1 REM Ficha:" INPUT " Ej.1.1.1
10 PRINT "Unidades: ";
20 INPUT N;
30 PRINT N
40 PRINT "Precio Unidad: ";
50 INPUT P
60 PRINT P
70 PRINT "TOTAL: ";N*P
```

1.1.2

```
 1 REM Ficha:" INPUT " Ej.1.1.2
10 PRINT "Unidades A: ";
20 INPUT N;
30 PRINT N
40 PRINT "Precio Unidad A: ";
50 INPUT P
60 PRINT P
70 PRINT "COSTE UDS. A: ";N*P
80 PRINT "Unidades B: ";
90 INPUT N1;
100 PRINT N1
110 PRINT "Precio Unidad B: ";
120 INPUT P1
130 PRINT P1
140 PRINT "COSTE UDS. B: ";N1*P1
150 PRINT
160 PRINT "COSTE TOTAL: ";N*P+N1*P1
```



1.1.3.

```
 1 REM Ficha:" INPUT " Ej.1.1.3
10 PRINT "Importe pagado: ";
20 INPUT E
30 PRINT E
40 PRINT "Unidades A: ";
50 INPUT N;
60 PRINT N
70 PRINT "Precio Unidad A: ";
80 INPUT P
90 PRINT P
100 PRINT "COSTE UDS. A: ";N*P
110 PRINT "Unidades B: ";
120 INPUT N1;
130 PRINT N1,
140 PRINT "Precio Unidad B: ";
150 INPUT P1
160 PRINT P1
170 PRINT "COSTE UDS. B: ";N1*P1
180 PRINT
190 PRINT "COSTE TOTAL: ";N*P+N1*P1
200 PRINT "Cantidad a devolver: ";
210 PRINT -N*P+-N1*P1+E
```

1.2.1.

```
 1 REM Ficha:" INPUT " Ej.1.2.1
10 PRINT "Dinero que tiene: ";
20 INPUT X
30 PRINT X
40 PRINT "Gastado: ";
50 INPUT Y
60 PRINT Y
70 PRINT "Dinero que le sobra: ";X-Y
```

1.2.3.

```
 1 REM Ficha:" INPUT " Ej.1.2.3
10 PRINT "Dinero que gana al mes: ";
```

```
 20 INPUT S
 30 PRINT S
 40 PRINT "Gastado en vivienda: ";
 50 INPUT V
 60 PRINT V
 70 PRINT "Gastado en alimentacion: ";
 80 INPUT X
 90 PRINT X
100 PRINT "Gastado en vestuario: ";
110 INPUT Y
120 PRINT Y
130 PRINT "Gastado en Transportes y diversiones: ";
140 INPUT Z
150 PRINT Z
160 PRINT "Numero de meses: ";
170 INPUT M
180 PRINT M
190 PRINT "AHORRO: ";-V*M-X*M-Y*M-Z*M+S*M
```

1.3.1.

```
  1 REM Ficha:" INPUT " Ej.1.3.1
  2 PRINT "AREA DE POLIGONO REGULAR"'"Descomponer en
    triangulos"
 10 PRINT "Longitud del lado: ";
 20 INPUT L
 30 PRINT L
 40 PRINT "Longitud de la apotema: ";
 50 INPUT A
 60 PRINT A
 70 PRINT "Numero de lados: ";
 80 INPUT N
 90 PRINT N
100 PRINT "AREA: ";L*A/2*N
```

```
  1 REM Ficha:" INPUT " Ej.1.3.1
  2 PRINT "AREA DE TRIANGULOS"
 10 PRINT "Longitud de la base: ";
 20 INPUT L
 30 PRINT L
 40 PRINT "Longitud de la altura: ";
 50 INPUT A
 60 PRINT A
 70 PRINT "AREA: ";L*A/2
```

```
  1 REM Ficha:" INPUT " Ej.1.3.1
  2 PRINT "AREA DE PIRAMIDES REGULARES"
 10 PRINT "Longitud de un lado de la base: ";
 20 INPUT L
 30 PRINT L
 40 PRINT "Numero de lados: ";
 50 INPUT N
 60 PRINT N
 70 PRINT "Longitud de la apotema: ";
 80 INPUT A
 90 PRINT A
100 PRINT "AREA DE PIRAMIDE REGULAR: ";L*N*A/2
```

1.3.2.

```
  1 REM Ficha:" INPUT " EJ.1.3.1
  2 PRINT "AREA DE CILINDROS"
 10 PRINT "Longitud del radio de la base: ";
 20 INPUT R
 30 PRINT R
```

```
40 PRINT "Longitud de altura: ";
50 INPUT A
60 PRINT A
70 PRINT "AREA LATERAL: ";2*PI*R*A
```

```
 1 REM Ficha:" INPUT " EJ.1.3.2
 2 PRINT "AREA DE CILINDROS"
10 PRINT "Longitud del radio de la base: ";
20 INPUT R
30 PRINT R
40 PRINT "Longitud de la altura: ";
50 INPUT A
60 PRINT A
70 PRINT "AREA TOTAL: ";2*PI*R*A+2*PI*R^2
```

1.4.1.

```
 1 REM Ficha:" INPUT " Ej.1.4.1"
 2 PRINT "VOLUMEN DEL PARALELEPIPEDO"
10 PRINT "Longitud del largo  de la base: ";
20 INPUT L
30 PRINT L
40 PRINT "Longiud del alto de la base: ";
50 INPUT A
60 PRINT A
70 PRINT "Altura de la figura: ";
80 INPUT A1
90 PRINT A1
100 PRINT "VOLUMEN: "
110 PRINT
120 PRINT ,L*A*A1
```

```
 1 REM Ficha:" INPUT " Ej.1.4.1"
10 PRINT "Longitud del diametro: ";
20 INPUT D
30 PRINT D
40 PRINT "SUPERFICIE DE LA ESFERA: ";PI*D^2
```

```
 1 REM Ficha:" INPUT " Ej.1.4.1"
10 PRINT "Longitud del diametro: ";
20 INPUT D
30 PRINT D
40 PRINT "VOLUMEN DE LA ESFERA: ";PI/6*D^3
```

1.5.1.

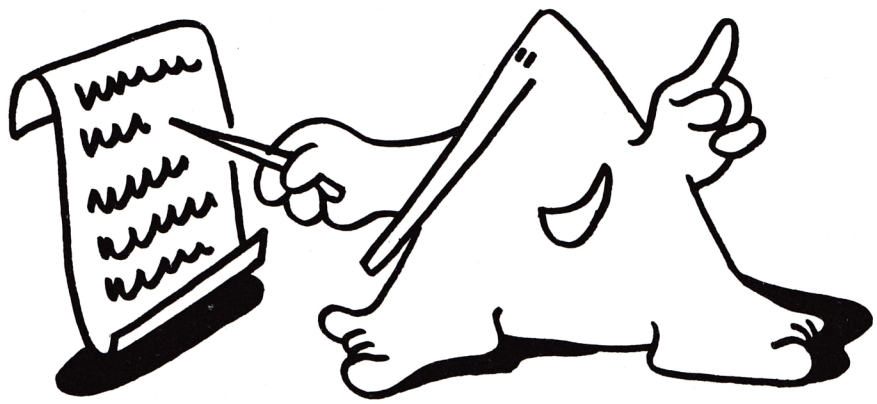
```
 1 REM Ficha:" INPUT " Ej.1.5.1"
10 PRINT "VELOCIDAD DE UN MOVIL"
20 PRINT "Espacio recorrido : ";
30 INPUT E
40 PRINT E
50 PRINT "Tiempo empleado: ";
60 INPUT T
70 PRINT T
80 PRINT "VELOCIDAD: ";E/T
```

3.1.

```
 1 REM Ficha :" INPUT " Ej.3.1"
10 PRINT "EDAD APROXIMADA DE UNA PERSONA"
20 PRINT "Ano de nacimiento: ";
30 INPUT A
40 PRINT A
50 PRINT "Ano actual: ";
60 INPUT A1
70 PRINT A1
80 PRINT "Edad :";A1-A;" anos"
```

GUIA DE PALABRAS BASIC

GUIA DE PALABRAS BASIC



ON

ON ¡expresión! GOTO ¡lota de números de línea!

Produce un salto incondicional a la línea seleccionada por **expresión** dentro de la **lata de números de líneas.**

Ejemplos:!

```
10 CLS: PRINT "Pulse tecla de la opción elegida".
20 PRINT "1...ALTAS"
30 PRINT "2...BAJAS"
40 PRINT "3...MODIFICACIONES"
50 INPUT T
60 ON T GOTO 100, 200, 300
70 GOTO 10
100 REM COMIENZAN ALTAS
110 ...
190 GOTO 10
200 REM COMIENZAN BAJAS
210 ...
290 GOTO 10
300 REM COMIENZAN MODIFICACIONES
310 ...
390 GOTO 10
```

OPENIN

OPENIN ¡nombre de fichero!

Abre el fichero **nombre de fichero,** previamente creado, para permitir la entrada de información en él.

Ejemplos:

```
10 OPENIN "AUTORES".
20 OPEN 9, "CERVANTES", "EL QUIJOTE"
30 CLOSEIN
```

OPEN OUT

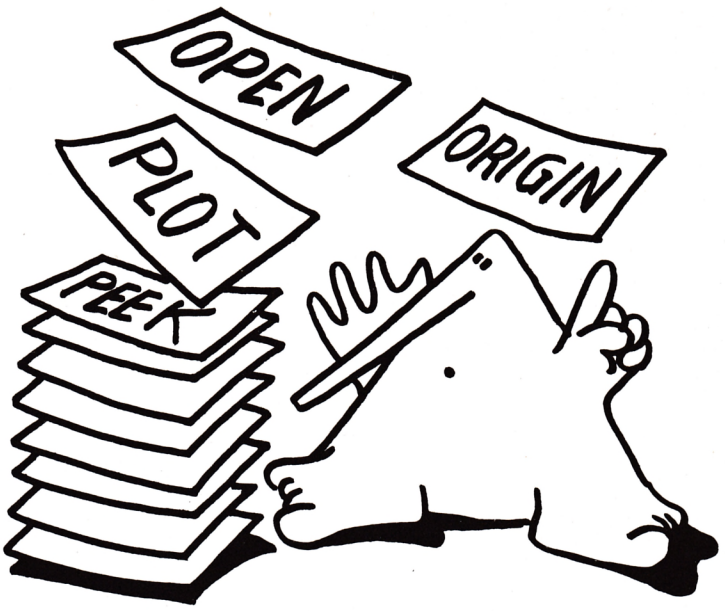
OPEN OUT ¡nombre de fichero!

```
10 OPEN OUT "AUTORES"
20 INPUT "AUTOR?:"; A$
30 INPUT "TITULO?¡"; T$
40 WRITE  9, A$, T$
50 CLOSE OUT
```

ORIGIN

ORIGIN ¡expresión 1, expresión 2!
Cambia el origen de coordenadas de los gráficos en pantalla a los valores dados por **expresión 1** y **expresión 2.**
Ejemplos:

```
10 ORIGIN 50,50
```



PAPER

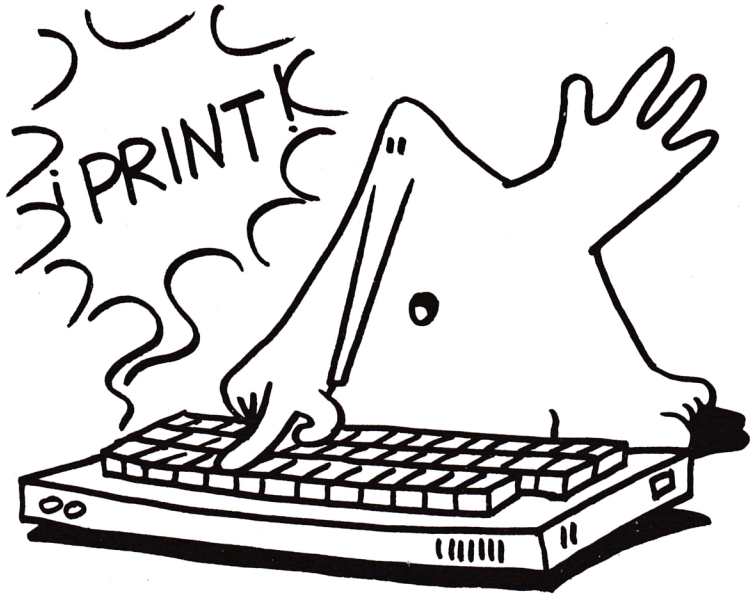
PAPER ¡expresión!
Deja el color de fondo de los caracteres escritos en pantalla al dado por el valor **expresión** (entre 0 y 15).
Ejemplo:

```
10 POR P = OTO 15
20 CLS
20 FOR X = 1 TO 300:NEXT X
40 PRINT "PRUEBA"
50 PRINT P
60 NEXT P
```

PEEK

PEEK ¡(expresión)!
Retoma el contenido de la dirección de memoria dada por **expresión.**

```
10 INPUT "dirección de memoria?:" d
20 PRINT "El contenido de d es:"; PEEK (d)
30 GOTO 10
```



PEN

PEN ¡expresión 1!, ¡expresión 2!
Deja el color de la tinta en escritura al dado por **expresión 1.** El color de fondo es dado por **expresión 2.** Se puede prescindir de **expresión 1** o de expresión 2.
Ejemplo:

```
10 INPUT "TINTA?:"; T
20 PEN T
30 PRINT "PRUEBA"
40 PRINT T
```

Esta función retorna el valor de

```
PRINT P1
3,14159265
```

PLOT

PLOT ¡expresión 1!, ¡expresión 2!

Sitúa un punto en la pantalla en la posición dada por **expresión 1** y **expresión 2.**

```
10 PLOT 600, 300
```

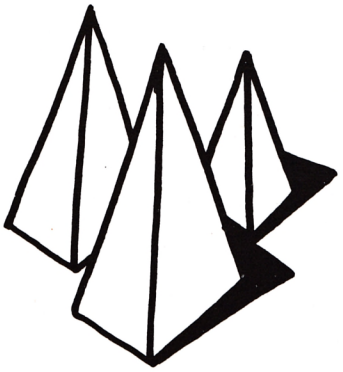
PLOTR

PLOTR ¡expresión 1!, ¡expresión 2!
Sitúa un punto en la pantalla en la posición relativa a la posición actual del cursor y dada por **expresión 1** y **expresión 2**

```
10 PLOT 600, 300
20 PLOTR 10, 10
```

PROGRAMA comentado en BASIC

Las Torres de Hanoi



por Víctor J. Campo López

Presentamos este mes un juego en apariencia sencillo, pero como podrá observar, su nivel de dificultad va creciendo en progresión geométrica, a medida que se aumenta el número de pisos de la torre inicial.

El juego consta de tres torres, numeradas 1, 2 y 3. La primera contiene un número de discos (pisos), que determina el jugador, estos discos están colocados desde la base hacia arriba, de mayor a menor. Las otras dos torres (2 y 3) están inicialmente vacías.

Ha de conseguir pasar todos los discos de la torre 1, a la torre 3, pudiendo para ello auxiliarse con la torre 2.

La única condición del juego, es que en ninguna torre, puede situarse un disco mayor sobre otro menor. Es decir, en todos los movimientos, el disco a mover se situara sobre otro mayor.

Para conocer un poco la posible estrategia de movimientos, le recomendamos comenzar con tres pisos y una vez completado el juego pasar a cuatro pisos. Le sorprenda como un piso más aumenta considerablemente el grado de dificultad y por tanto el número de movimientos a realizar.

El máximo nivel (8 pisos), resulta increíblemente difícil, aunque no es imposible. Paciencia y atención le serán imprescindibles.

Una última recomendación, al jugar se dará cuenta de que para conseguir un determinado cambio a de realizar una secuencia de movimientos y que existe una secuencia mínima, que es obviamente la que debemos utilizar, anote estos movimientos, la experiencia acumulada le resultara fundamental en los siguientes niveles.

Comentarios al programa

A continuación, como en números anteriores pasamos a describir las líneas fundamentales del programa, para su compresión e introducción de mejoras, por parte del lector, pues de eso se trata, de mejorar programando bajo el atractivo de un juego.

10 — 50. Líneas de presentación y explicación suscinta de las reglas del juego. Recuerde que CHR$ (24), produce la inversión del vídeo (cambia colores tinta-papel).

60. Establece una ventana que será utilizada para los mensajes del juego. Dicha ventana abarca las 80 columnas del modo 2 establecido en la línea 20 y desde la línea de pantalla 20 hasta la 25 (última).

70. Se establece que para la ventana, los colores del papel y de la tinta serán los contrarios de los normales (vídeo inverso). La sentencia CLS seguida del número de ventana produce el borrado de la misma con los nuevos colores.

Dado que muchos usuarios sólo disponen de monitor verde, no se utilizan estos de forma que realcen la presentación, esto se lo dejamos en última instancia al usuario, como posible mejora.

80. Emite un mensaje por la ventana No 1.

90. La llamada a esta rutina de la ROM produce una pausa continua, hasta tanto no se presione una tecla.

100. Dimensiona dos var indexadas (matrices). T de 9 filas por 3 columnas y A de 1 fila por 3 columnas.

110-120. Asigna a A$ un asterisco y a D$ 25 asteriscos.

130-140. Solicita por la ventana establecida el número de discos de la torre inicial y controla que no se sobrepase el límite de ocho. La var. AL almacena dicho número.

210-260. El bucle de longitud igual al número de pisos seleccionado, carga los valores iniciales de los elementos de la matriz T. La var. NM contiene el número de movimientos realizados.

270. Inicializa la matriz A.

300-410. Los dos bucles se encargan de dibujar las torres en la posición en que se encuentran. Observe como la función LEFT es la encarga de producir la fragmentación de la cadena D$ que tiene 25 asteriscos, con el fin de representar únicamente los necesarios, según el piso en que se encuentre. La posición viene marcada por el algoritmo incluido en la función TAB. Para su mejor compresión puede a mano un ejemplo con números reales de las expresiones de la línea 350.

510. Deja libre la ventana para posteriores mensajes.

520. Controla si las dos primeras torres no contienen discos, en caso afirmativo transfiere el con-

```
10 ' TORRES DE HANOI
20 MODE 2:LOCATE 17,5: PRINT CHR$(24);"
L A S    T O R R E S   D E   H A N O I "
30 LOCATE 17,6: PRINT "
                        ";CHR$(24): PRINT

40 PRINT TAB(10)"DEBE MOVER TODOS LOS DI
SCOS DE LA TORRE 1 A LA TORRE 3."
50 PRINT : PRINT TAB(10)"DE FORMA QUE NO
 SE COLOQUE UN DISCO MAYOR SOBRE OTRO ME
NOR"
60 WINDOW #1,1,80,20,25
70 PAPER #1,1:PEN #1,0:CLS #1
80 LOCATE #1,5,3:PRINT #1,"P R E S I O N
 E   U N A   T E C L A   P A R A   C O
N T I N U A R"
90 CALL &BB18:CLS
100 DIM T(9,3),A(3)
110 A$="*"
120 D$=STRING$(25,"*")
130 CLS #1:LOCATE #1,5,3:INPUT #1,"DE CU
ANTOS DISCOS QUIERE LA TORRE INICIAL (1
A 8)";AL
140 IF AL>8 THEN  PRINT "DEMASIADOS DISC
OS":GOTO 130
200 ' COMIENZO DEL JUEGO
210 NM=1
220 FOR I=1 TO AL
230 T(I,1)=AL-I+1
240 T(I,2)=0
250 T(I,3)=0
260 NEXT
270 A(1)=AL:A(2)=0:A(3)=0
300 ' DIBUJO TORRES
310 CLS
320 FOR AP=AL TO 1 STEP -1
330 PRINT TAB(10)A$;TAB(30)A$;TAB(50)A$;
340 FOR I=1 TO 3
350 PRINT TAB(I*20-10-T(AP,I))LEFT$(D$,T
(AP,I)*2+1);
360 NEXT I
370 PRINT
380 NEXT AP
390 FOR K=1 TO 60: PRINT "=";:NEXT
400 PRINT TAB(9)1;TAB(29)2;TAB(49)3
410 PRINT
500 ' MOVIMIENTO
510 CLS #1
520 IF A(1)=0 AND A(2)=0 THEN 710
530  PRINT #1,TAB(20)"MOVIMIENTO NUMERO
";NM
540 PRINT #1
550 PRINT#1,"DESDE LA TORRE ?";
560 z$=INKEY$:IF Z$="" THEN 560 ELSE GOS
UB 800
570 IF Z$="" THEN 560 ELSE O=VAL(Z$):PRI
NT #1,O,,
580 IF A(O)=0 THEN  PRINT #1,CHR$(24);"L
A TORRE";O;"ESTA VACIA";CHR$(24):PRINT #
1,TAB(40)"PRESIONE PARA CONTINUAR":CALL
&BB18:GOTO 500
590 IF O>3 THEN 530
600 PRINT #1,"HACIA LA TORRE ?";
610 z$=INKEY$:IF Z$="" THEN 610 ELSE GOS
UB 800
620 IF Z$="" THEN 610 ELSE F=VAL(Z$):PRI
NT #1,F
630 IF A(F)=0 THEN 650
640 IF T(A(F),F)<T(A(O),O) THEN  PRINT #
1,"NO ES VALIDO,PRESIONE PARA CONTINUAR"
:CALL &BB18:GOTO 500
650 NM=NM+1
660 A(F)=A(F)+1
670  T(A(F),F)=T(A(O),O)
680 T(A(O),O)=0
690 A(O)=A(O)-1
700 GOTO 300
710 ' TODOS LOS DISCOS EN LA DERECHA
720 CLS #1:LOCATE #1,10,3: PRINT #1,"ENH
ORABUENA, LO CONSIGUIO !, EN ";NM; "MOVI
MIENTOS"
730 LOCATE #1,15,4:PRINT  #1,"SE ATREVE
CON ";AL+1;"PISOS (S/N)"
740 LOCATE #1,43,4:INPUT #1,Z$
750 IF Z$="S" OR Z$="s" THEN al=al+1:GOT
O 140 ELSE RUN
800 IF VAL(Z$)>=1 AND VAL(Z$)<4 THEN RET
URN
810 PRINT #1,TAB(10)"EL   N U M E R O ";
Z$;"  D E   T O R R E   N O   E S   V A
L I D O"
820 PRINT #1,TAB(18)"PRESIONE UNA TECLA
PARA CONTINUAR"
830 CALL &BB18
840 CLS #1:GOTO 530
```

trol a la 710 y siguientes, de fin del juego.

530. Indica el número de tirada con movimientos que se va a efectuar.

550. Origen del movimiento. Basta con presionar el número de la torre, puesto que se va a utilizar INKEY$ y no INPUT para la entrada.

560. Si no se ha pulsado nada se vuelve a la misma línea en un bucle indefinido, que finaliza exclusivamente, con la pulsación de una tecla. En caso contrario se transfiere el controla la subrutina de la línea 800 encargada de verificar la validez de la entrada.

570. Como la entrada se ha realizado con var alfanumérica esta línea se encarga mediante la función VAL, a pasar a numérica.

580. Controla si la torre seleciona como origen del movimiento contiene o no discos a mover.

600-620. Proceso similar con la torre de destino.

630-640. Verifican la validez del movimiento destino. Recuerde que no puede colocarse un disco mayor sobre otro menor.

650. Incremento del contador de movimientos.

660-700. Alteración de los valores de la matriz en función de las var. D y F (torres de Origen y Fin), para proceder al dibujo de la nueva situación, por la transferencia a la línea 300.

710-720. Se ha conseguido el objetivo, la var. NM indica en cuántos movimientos.

730-750. Le invita a comenzar un nuevo juego con un grado mayor de dificultad. Si la respuesta es distinta de ''S'' o ''s'', el juego comienza mediante la orden RUN.

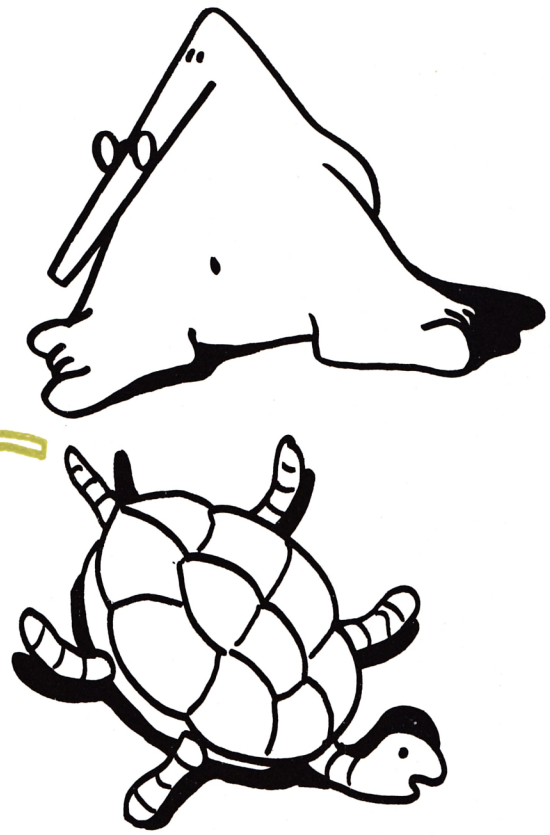
800-840. Subrutina de control a fin de verificar que se ha pulsado un 1, 2 ó 3, como número de torre, cualquier otro número o letra se interpreta como error, indicándose por el mensaje de la 810. En caso de ser válido la línea 800 contiene el RETURN de la subrutina.

Mejoras

Sin duda cuando haya jugado con el programa se le ocurriran un sinfín de modificaciones, tendentes a mejorar la presentación o calidad del programa, no lo dude hagalas, ello le facilitará una mayor compresión del programa y le permitira aumentar sus conocimientos, que en definitiva es el fin que nos proponemos con estos programas.

Logo del Amstrad

Capítulo VI. Gráficos tortuga: segunda parte



VI.1. Las plumas de la tortuga

Supongamos que en este punto comienza una sesión de trabajo con Dr. Logo. Así pues comenzamos por cargarlo en la memoria del ordenador. A estas alturas ya habremos observado, con toda seguridad, que el cursor y todos los caracteres que escribamos en estas condiciones se escriben en tono amarillo sobre fondo azul. Si le damos a la tortuga la orden de que aparezca, st, **podemos observar que, de igual modo, siempre al comenzar es amarilla y dibuja en este color sobre un fondo tan azul como el anterior.**

Por otra parte es probable también que a estas alturas nos hayamos ya preguntado si el hecho de que la tortuga siempre sea visible en la pantalla de gráfico, es inevitable o no. Este capítulo esta dedicado a resolver todas las dudas en relación con los colores y con el estado, en general, de la pantalla y de la tortuga.

La pantalla de gráficos está formada por 640 puntos horizontales (podemos decir entonces que esta formada por 640 columnas) y 400 verticales (400 filas). Una cosa es la observación que cualquier persona puede hacer sobre la ubicación en un momento determinado de la tortuga, y otra muy distinta la manera en que esa situación puede objetivarse para poder hacer, por ejemplo, cálculos con ella.

Es bien conocido el hecho de que fue el matemático francés Descartes el que introdujo los sistemas de referencia ''cartesianos'' en los que se basa nada menos que la Geometría analítica. La pantalla de los gráficos de Dr. Logo está dotada de un sistema de referencia cartesiano. Aunque no es este el lugar para describir con detalle este concepto, digamos que un tal sistema de referencia consiste en dos rectas perpendiculares, llamadas ejes de coordenadas de este sistema de referencia, y el punto por ellas definido (el punto de corte), llamado origen de coordenadas. Se puede demostrar que, dada esta estructura en el plano, si dos puntos tienen iguales sus proyecciones sobre los ejes de coordenadas entonces son el mismo punto. En consecuencia, dados **dos** puntos distintos es seguro que tienen distinta alguna de las proyecciones sobre los ejes. Recordemos ya, finalmente, que estas proyecciones suelen llamarse coordenadas del punto y que el origen de coordenadas tiene las coordenadas (0,0).

La pantalla de gráficos de Dr. Logo tiene incorporado un sistema de referencia: Los ejes de coordenadas son las rectas vertical y horizontal, que se cruzan precisamente en el punto central de la pantalla. De modo

que cuando la tortuga ejecuta la orden "cs", se situa en el punto de coordenadas [0 0], es decir en el origen de coordenadas. (Cuantos pasos puede dar la tortuga hacia arriba desde el origen de coordenadas sin salirse de la pantalla?: La mitad del número de líneas horizontales con que cuenta la pantalla, es decir, 200 pasos. Evidentemente el número de pasos que puede dar hacia abajo depende del número de líneas de texto con que cuente en ese momento la pantalla. Lógicamente en el modo de pantalla completa se dispondrá de los mismos 200 pasos. Por razones análogas, el número de pasos que la tortuga puede dar horizontalmente hacia la izquierda o hacia la derecha, es de 320.

En este punto surge un interrogante de los más interesantes. La pantalla tiene un número finito de puntos, por supuesto, y cada uno constituye una unidad indivisible. De modo que si se nos ocurre dar a la tortuga la orden de que avance una cantidad fraccionaria de pasos, nos vamos a encontrar con algún tipo de respuesta más o menos sorprendente. Tal vez aparezca el mensaje de error correspondiente, afirmando que la orden de avanzar no admite inputs fraccionarios. Lo mejor es probarlo ya: Escribamos por ejemplo:

fd 2.5

y esperemos a ver (suponemos que inmediatamente antes de esta orden se ha ejectuado la orden cs). Efectivamente la tortuga se ha movido y ha pintado en su trayecto. ¿Qué habrá pasado con el medio punto? (Lo habrá pintado? Probemos ahora a dar la orden:

fd 0.5

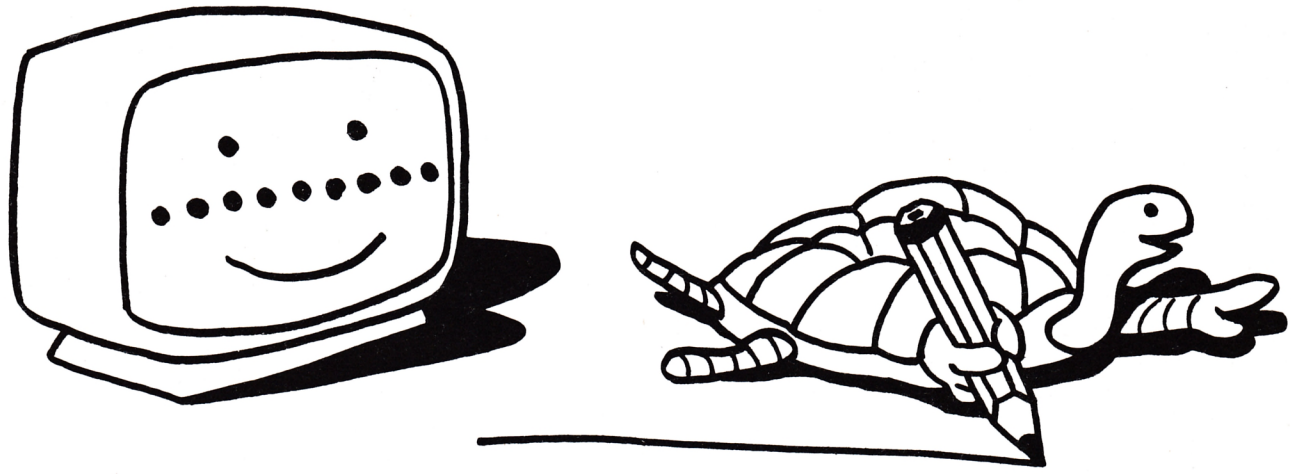
y veamos que es lo que pasa. ¡Aja!: al ejecutar la primera orden la tortuga avanzó 2 pasos, pero recordaba, de algún modo, que sólo la restaban 5 décimas de paso para dar el tercero. Por ello la segunda orden ha sido obedecida avanzando un paso: medio paso de esta orden más medio paso que restaba de la anterior. Más adelante veremos como podemos obtener las coordenadas "oficiales" o "reales" de la tortuga. Es decir, el número de pasos horizontales y verticales que ha dado, respecto del origen de coordenadas, y los decimales que ha sobrepasado estos valores la diferencia de las coordenadas "aparentes"). Veremos también, en su momento, que es posible ordenar a la tortuga que se situe en el punto que tiene unas determinadas coordenadas.

Pero de momento nos vamos a centrar en el estudio de los colores. Disponemos de la posibilidad de hacer nuestros gráficos tortuga en una amplia variedad de colores gracias a la versatilidad con que cuenta a este respecto Dr. Logo.

Ya hemos dicho que podemos pensar que la tortuga tiene incorporada una pluma que es la que dibuja cuando aquella se mueve por la pantalla. Esta pluma puede elegirse de entre 4 posibles. Así pues, tenemos 4 plumas, cada una de las cuales puede ser instalada en la tortuga para que sea con ella con la que pinte en adelante (hasta que volvamos a cambiarla). Por defecto, es decir, si no ordenamos nada a este respecto, la tortuga tiene instalada la pluma número 1. Las otras plumas son la 0,2 y 3. Podemos decidir el que sea otra pluma la instalada mediante la siguiente orden:

sector <n>

donde <n> designa el número elegido. Veamos un ejemplo que aclare por completo las ideas:



EJEMPLO VI.1.

Vamos a hacer que la tortuga dibuje un cuadrado con cada una de las 4 plumas posibles. Después de cada cuadrado escribirá el número de la pluma que ha usado y girará 90 grados, para poder ver los cuatro cuadrados de los distintos colores. Comenzamos definiendo el procedimiento "cuadrado":

```
to cuadrado
repeat 4 [fd 100 rt 90]
end
```

El procedimiento "colores" alterna las órdenes: asignación de pluma, dibujo del cuadrado, giro de 90 grados y escritura del número de la pluma usada:

```
to colores
setpc 0 cuadrado rt 90 por "0.
setpc 1 cuadrado rt 90 por "1.
setpc 2 cuadrado rt 90 por "2.
setpc 3 cuadrado rt 90 por "3.
end.
```

Ejecutemos ahora el procedimiento colores. Probablemente nos llevemos una pequeña sorpresa: El primer cuadrado (que corresponde a la pluma número 0), no se ha dibujado aunque si aparece el número 0 en la pantalla. El segundo cuadrado es amarillo (claro, es la pluma usual la que lo ha pintado (número 1), el tercero corresponde a la pluma número 2 y es azul; por último el de la pluma número 3 es rojo.

¿Por qué no se pintó el primero? La respuesta es clara: el fondo de la pantalla de gráficos y el dibujo tienen el mismo color, por eso no se puede ver el dibujo. Pero de ahí se deduce que la pluma que gobierna siempre el color del fondo es la de número 0. Por ello basta aprender a cambiar los colores de las plumas para poder cambiar inmediatamente el color del fondo.

Todos los colores que Dr. Logo puede manejar se obtienen combinando el rojo, el verde y el azul. Cada uno de ellos interviene en alguna medida en cada color. Esta medida de la intervención se obtiene asignando a cada uno de los colores uno de los números 0, 1, y 2. El modo en que Dr. Logo procesa la composición del color es mediante una lista que representa la proporción en que los colores principales intervienen. Podemos obtener esta lista con la siguiente orden:

pal <N>

que, naturalmente, es una operación. Recibe como dato el número <N>, que representa el número de la pluma, y da como resultado la lista de números que definen el color de esa pluma.

EJEMPLO VI.2.

Para obtener la composición de la pluma número 3 basta escribir en modo interactivo:

```
pal 3
```

obteniéndose como resultado.

```
[2 0 0]
```

De modo que, este momento, la pluma número 3 tiene el color rojo a un máximo nivel (2) y los colores verde y azul ambos a 0. En consecuencia, esta pluma pinta de color rojo ahora.

Obtengamos las composiciones iniciales de las distintas plumas:

```
pal
[0 0 1]

pal 1
[2 2 0]

pal 2
[0 2 2]

pal 3
[2 0 0]
```

Todo esto es muy interesante pero incompleto, mientras no podamos cambiar la composición de cada una de las plumas. Observemos además que, hasta ahora, sólo podemos escribir las letras en amarillo: como pinta la pluma número 1.

Para cambiar la composición de una pluma se tiene la orden:

septal <N> <L>

donde <N> es el número de la pluma cuya composición se desea cambiar y <L> es una lista formada por 3 números, cada uno de los cuales ha de estar comprendido entre 0 y 2: es la composición del color que asignamos a la pluma. Además, el texto se escribe del color asignado a la pluma número 1.

Después de esto es claro que podemos hacer todas las combinaciones imaginables entre colores del texto (todo en un color, el que tenga la pluma número 1), color del fondo (lógicamente todo de un color), y colores de los gráficos.

EJEMPLO VI.3.

¿Como haremos para cambiar el color del fondo a rojo? Puesto que, sabemos que el color rojo puro se obtiene con la "fórmula" [2 0 0], bastará cambiar la composición de la pluma número 0 a esta composición. Así:

```
setpal 0 [2 0 0]
```

Si teníamos todavía en pantalla el dibujo realizado en el ejemplo anterior, observaremos que el cuadrado superior derecho sigue sin aparecer. Además, el cuadrado superior izquierdo desaparece por completo, dado que su color es igual que el del fondo. Ejecutemos ahora

```
setpal 0 [0 0 1]
```

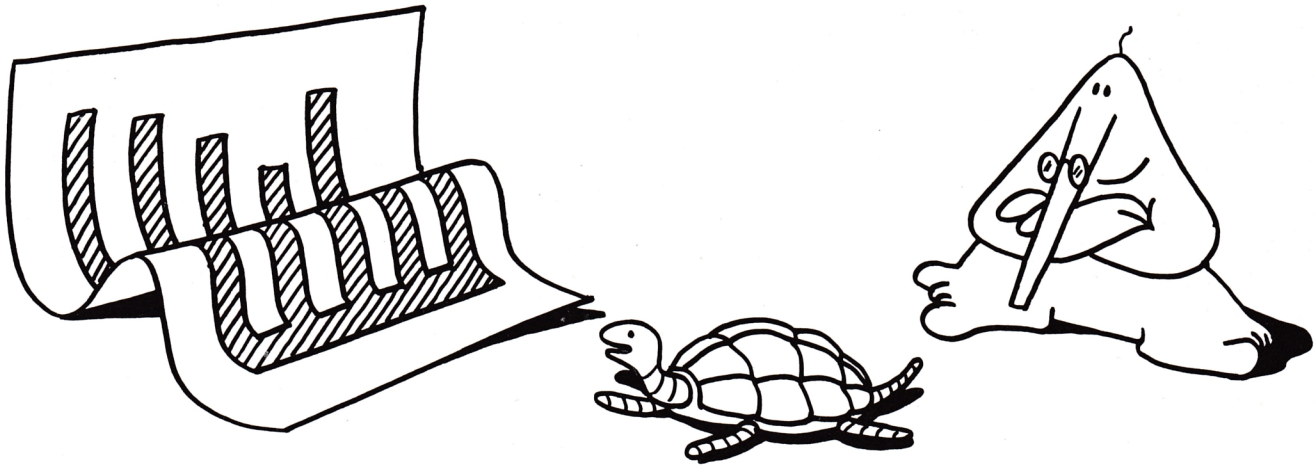
Como vemos, las cosas vuelven donde estaban, pero el cuadrado rojo no se borra. Cualquier cosa que la tortuga hubiera dibujado ahora la hubieramos hecho visible cambiando el color de la pluma 0. Pero el cuadrado superior derecho cmabia de color de la pluma 0, de modo que **nunca será visible.** Para confirmar esto podemos, por ejemplo, ejecutar la orden.

```
setpal 0 [2 0 0]
```

Vemos que, además de cambiar el color de todo el texto a rojo, también ha cambiado a ese color el cuadrado inferior derecho que fue pintado con la pluma número 1.

Resumiendo: Disponemos de 4 plumas, cada una de las cuales puede cargarse con un color obtenido como antes se ha descrito. La pluma número 0 gobierna siempre el color del fondo. La pluma número 1 gobierna siempre el color del texto. La tortuga puede cargarse con cualquier pluma y, por tanto, con el color que ésta tenga incorporado. Pero los dibujos realizados con una pluma determinada, cambian siempre que se cambie la composición del color de la pluma.

No es menester insistir en el hecho de que con estas técnicas se pueden conseguir multitud de efectos en pantalla. Uno de ellos nos permitiría, por ejemplo, realizar un dibujo "en modo invisible" y presentarlo todo de golpe una vez acabado.



EJEMPLO VI.4.

El siguiente programa realiza un dibujo. No se ve nada de él hasta que esta concluido, momento en que se presenta de golpe en pantalla. La idea es muy simple: instalemos un determinado color de fondo, por ejemplo rojo. Carguemos una pluma, por ejemplo, la 1 con este color. Asignemos esta pluma a la tortuga. Pongamos la tortuga a pintar; pintara del mismo color del fondo, de modo que no veremos nada. Finalmente cambiemos la composición de la pluma de la tortuga a otro color (por ejemplo azul).

```
to golpe
setpal 0 [2 0 0]
setpal 1 [2 0 0]
setpc 1
repeat 10 [repeat 4[ + fd 50 rt 90] rt 36]
setpal 1 [0 0 2]
end
```

Si ejecutamos este procedimiento observaremos como la pantalla permanece en rojo hasta que ''de golpe'' presenta todo el dibujo.

Por otra parte, queda por ver toda una serie de procesos que se pueden hacer por el hecho de que la composición de los colores se haga precisamente mediante una lista. Volveremos sobre ello en su momento.

VI.2. ESTADOS DE LA PLUMA.

De lo visto hasta ahora ya resulta la posibilidad de mover la tortuga por la pantalla sin que deje rastro en ella, es decir, sin que realice dibujo alguno. En efecto, como hemos visto, basta cargarla precisamente con el color del fondo. Si se hace con la pluma 0 el dibujo no será posible verlo jamás; si se hace con otra pluma, entonces el dibujo será recuperable. Pero lo que, desde luego, sucederá es que la propia tortuga permanecerá invisible a nuestra vista en todo este proceso.

Como muchas veces lo que querremos será que simplemente no use su pluma para pintar, pero sin dejar de ser visible la propia tortuga, disponemos de unas órdenes que actuan sobre la pluma que tenga incorporada la tortuga, levantándola (para que no pinte) o bajándola (para que vuelva a pintar). La orden que levanta la pluma es:

pu

que es abreviatura de PenUp. Después de ejecutada esta orden, la tortuga permanecerá visible o no (dependiendo del color de la pluma incorporada), pero no realizará dibujo alguno en su trayectoria, hasta que se vuelva a poner la pluma en posición. Esto se consigue mediante la orden:

pd

que es abreviatura de PenDown. Cuando se carga el Dr. Logo, la pluma esta en esta posición como ya sabemos.

Nosotros ya sabemos borrar la pantalla de gráficos, a la vez que situamos la tortuga en el centro de la pantalla y mirando hacia arriba. Sólo tenemos que usar la orden ''cs''. Pero puede suceder que lo que queramos sea borrar algunas líneas de nuestros gráficos, o sencillamente que la tortuga, en vez de actuar como si tuviera instalada una pluma, actue como si tuviera incorporada una goma de borrar. Evidentemente, de alguna manera podríamos obtener este resultado sin más que asignarle la pluma 0 (color del fondo). Pero entonces no vemos a la tortuga en su camino. Estaríamos limitados en los procedimientos que podríamos escribir. Para evitarlo, Dr. Logo dispone de una nueva orden:

pe

que, cuando es ejecutada, instala en la tortuga una goma de borrar. El resultado es el mismo que si hubieramos instalado la pluma 0 pero con la diferencia de que en ningún momento dejaremos de ver a la tortuga (a no ser, claro está, que otros factores así lo hagan). Pero debemos tomar la ejecución de la orde ''pe'' como independiente de los colores. Nuestro consejo es sencillamente que pensemos en la goma de borrar.

Todavía tenemos más posibilidades. En efecto, la orden.

px

tiene un comportamiento curioso. Cuando se ha ejecutado esta orden, si la tortuga se mueve por puntos no pintados, entonces se comporta como si estuviera en posición ''pd'' y pinta normalmente. Pero cuando

pasa por puntos pintados de cierto color, entonces los deja pintados del color complementario o inverso del que tenían.

Es necesario advertir aquí, que la orden ''cs'' sitúa la pluma en posición ''pd'', cualquiera que fuera su situación anterior. Tal vez en este momento el estudioso de Logo se pregunte hasta que punto es esto una grave limitación. Por fortuna en Logo todo es conseguible (bueno, casi todo). Cuando hayamos profundizado más en Dr. Logo veremos de que forma tan sencilla podremos nosotros escribir utilidades que hagan lo que nosotros decidamos; para poder usarlas como mejor nos parezca, por ejemplo una orden ''hermana'' de ''cs'' pero que no altere el estado de la pluma; o una orden parecida que situe a la tortuga tan en el centro como cs pero sin afectar a los gráficos, etc., etc. Ya veremos todas estas cosas.

VI. 3. ESTADOS DE LA TORTUGA Y DE LA PANTALLA

Bueno, poco a poco se van acumulando nuevas instrucciones y propiedades de tipo estático que la tortuga y la pantalla puede o no cumplir en un momento determinado. Surgen enseguida de modo natural dos preguntas a la vista de las últimas instrucciones estudiadas:

1) ¿Cómo podemos saber en un momento determinado cual es el estado de la pluma? ¿Cómo podemos saber las coordenadas de la tortuga?

2) ¿Es posible que las respuestas obtenidas a la pregunta anterior sean manejadas de modo no interactivo por un programa? Es decir: si la respuesta a la pregunta anterior es una nueva orden que presenta en pantalla estos datos, ¿hay alguna manera de que esos datos no se pretenden al operador, pasando a ser datos que pueda manejar un procedimiento?

Lógicamente ambas preguntas encuentran respuesta afirmativa. Algunas veces este tipo de instrucciones se llaman en otros contextos variables del sistema. Son muy parecidas a funciones en el sentido de que su ejecución no provoca ninguna acción sobre dato alguno, sino que da un resultado. La única diferencia con las operaciones estriba en que no reciben ningún argumento. En el caso concreto que nos ocupa tal vez sea un tanto inexacta la afirmación anterior, dado que se debe pensar que Dr. Logo se basa en el estado de la tortuga y de la pantalla para obtener el resultado proporcionado por estas primitivas. Concretamente la orden que nos informa sobre la tortuga es:

tf

Cuando se ejecuta esta orden (abreviatura de Turtle Facts), se produce como resultado una lista formada por 6 palabras:

— La primera palabra es un número: la coordenada **x** de la tortuga en el momento de ejecutar la orden.

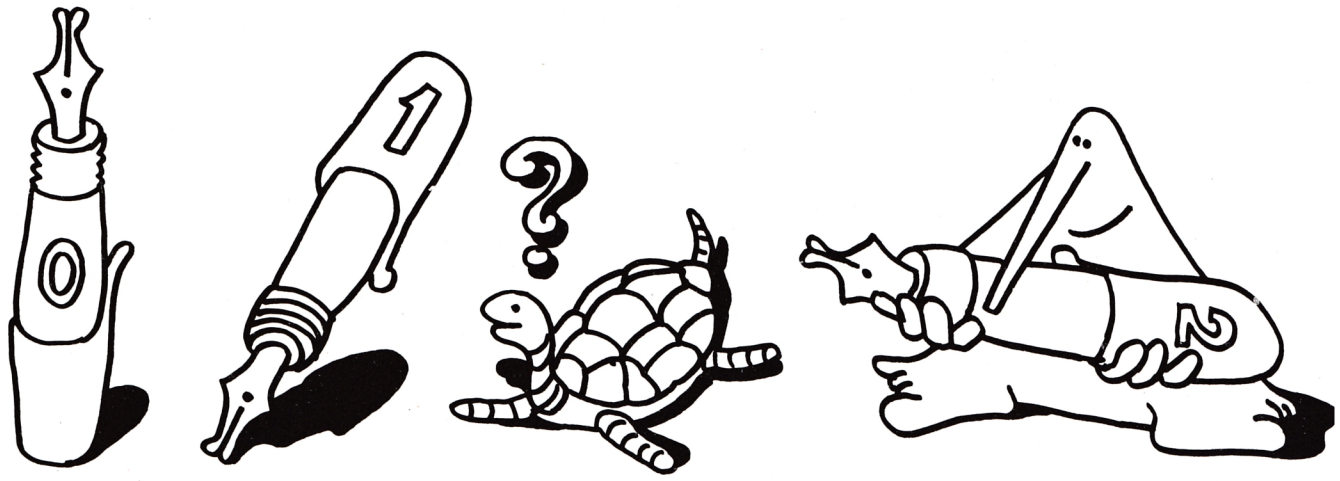
— La segunda es de nuevo un número: la coordenada **y** de la tortuga. Recordemos que la presencia de decimales en estos números indica fracciones de pasos, cuyo funcionamiento ya se estudió antes. Más adelante volveremos con detalle sobre el movimiento de la tortuga con coordenadas (movimiento absoluto).

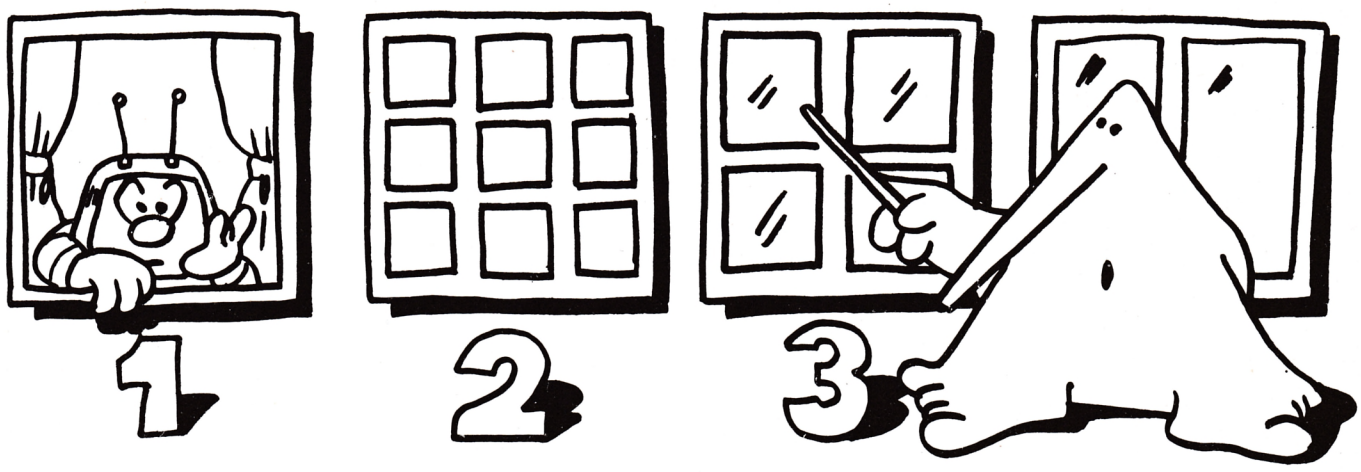
— La tercera palabra es de nuevo un número: indica el rumbo u orientación actual de la tortuga. Aunque volveremos sobre esto cuando estudiemos el movimiento absoluto, podemos indicar aquí que el rumbo es el número de grados que la tortuga se encuentra girada respecto de la orientación 0 que es la que tiene la tortuga cuando mira exactamente hacia arriba en la pantalla.

— La cuarta palabra indica el estado actual de la tortuga. Como hemos visto ya puede ser una de estas cuatro: pd, pu, pe y px. Hemos de notar que, aunque en Dr. Logo estas órdenes se escriben en minúscula, el resultado producido por la orden tf en la cuarta palabra, aparece escrito en mayúsculas.

— La quinta palabra es de nuevo un número. Indica el número de la pluma asignada a la tortuga. Como sabemos puede ser 0, 1, 2 ó 3.

— Finalmente, la sexta palabra es una de las siguientes: TRUE (cierto) si la tortuga esta en situación de visible (se ha ejecutado st o bien no se ha ejecutado ht), o FALSE (falso) si la tortuga no esta visible (se ha ejecutado ht).





Es muy importante señalar aquí que, como el resultado proporcionado por esta operación es una lista, es posible obtener de ella todos sus elementos de forma individual por procedimientos escritos al efecto si es necesario. Sólo cuando hayamos estudiado el manejo de listas será posible ver como se pueden manejar los datos así obtenidos. Podemos intuir aquí que será posible escribir procedimientos que seleccionen, con posibilidades de proceso posterior, los distintos objetos-Logo de una lista.

De manera completamente paralela a la anterior existe una operación que proporciona resultados acerca de la pantalla. La instrucción es:

sf

que es abreviatura de Screen Facts. El resultado que proporciona es de nuevo una lista. En este caso la lista está formada por 5 palabras.

— La primera indica el número de pluma asignada al manejo del color del fondo. Según ya se ha indicado siempre es igual 0.

— La segunda palabra indica el estado de la pantalla. Puede ser, como se indicó en su momento, una de las siguientes: SS (pantalla mixta), FS (sólo gráficos) y TS (sólo texto). Como sucedía antes con la orden tf, las órdenes anteriores se escriben en minúsculas pero la orden sf las presenta en mayúsculas.

— La tercera palabra indica el número de líneas de texto que tiene asignada la pantalla mixta. Recordemos que la orden setsplit instala un número de líneas de texto para el caso de que se ejecute después la orden ss. Por defecto, el número asignado es 5.

— La cuarta ventana proporciona un dato que todavía no hemos estudiado. El estado de la "ventana". Volveremos sobre ello.

— La quinta palabra no tiene un uso específico. Siempre es un 1.

Todo lo dicho para la orden tf respecto a la posibilidad de manejar los elementos de la lista resultado por procedimientos es aplicable a la orden sf en su totalidad.

EJEMPLO VI.5.

Inmediatamente después de cargar el lenguaje se obtiene:

```
tf
[0 0 0 PD 1 TRUE]
sf
[0 TS 5 WINDOW 1]
```

Ejecutemos algunas órdenes y veamos después los resultados proporcionados por estas operaciones:

```
fd 100
rt 90
pu
setpc 3
ht
tf'
[0 100.0003431 90 PU 3 FALSE]
setsplit 2
fs
sf
```

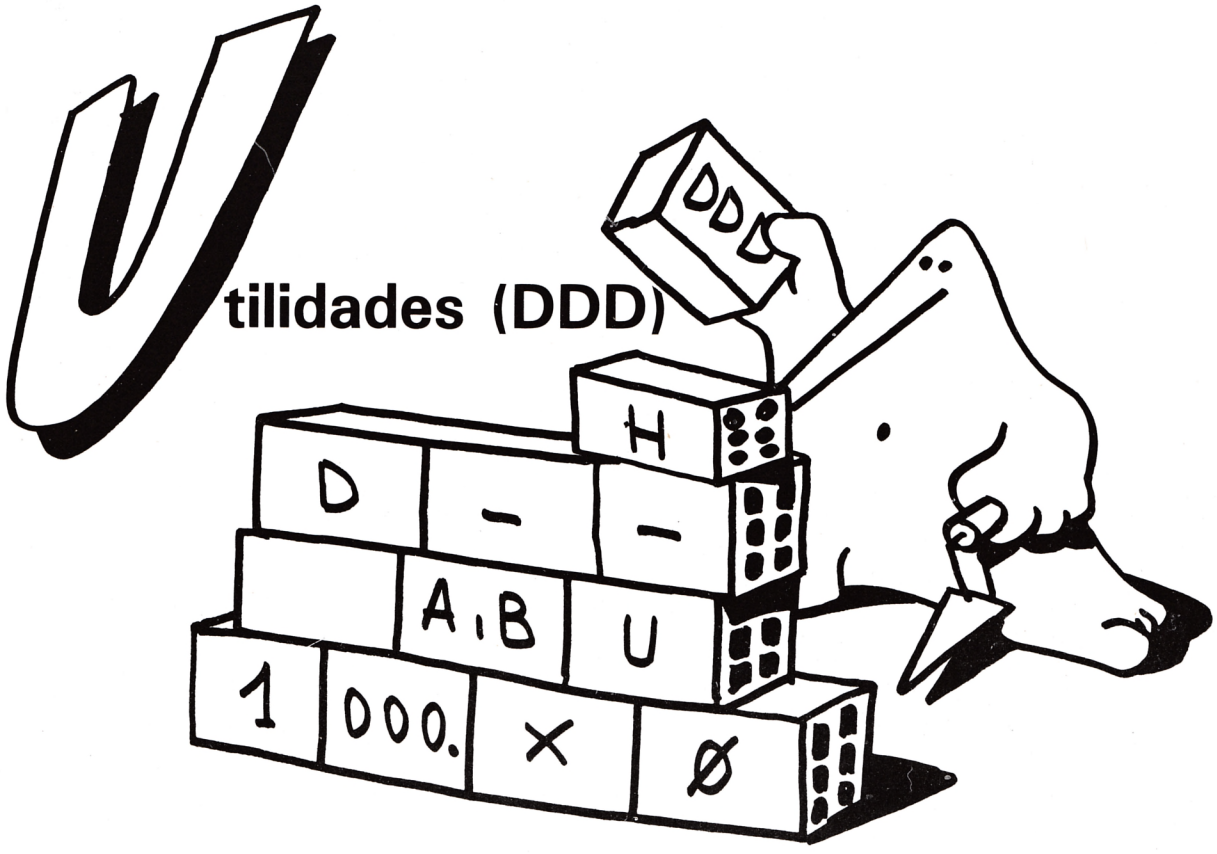
(lógicamente, en este momento no puede verse el resultado pues la orden fs instala la pantalla completa. De modo que sólo podemos ver el resultado cuando la segunda palabra no marque FS. Bueno, si se han seguido todos los pasos anteriores ahora podríamos almacenar la lista anterior en alguna variable y luego imprimirla. Más adelante estudiaremos con detalle todos los conceptos aquí involucrados. Por ahora sólo hemos de preocuparnos de la lista resultado de sf).

```
make "a sf
ts
show: a
[O FS 2 WINDOW 1]
```

Este capítulo ha presentado nuevas ampliaciones sobre el lenguaje. Recordemos que, durante una sesión de Dr. Logo, la mayor parte de las órdenes producen cambios estáticos en la situación. Esto significa que la situación establecida permanecerá hasta que de modo expreso otra orden la cambie. Las dos últimas órdenes vistas pueden contribuir a que nuestro control de la situación sea mayor.

PROGRAMA comentado en LOGO

Utilidades (DDD)



En alguna ocasión hemos hecho ya referencia a uno de los hechos más trascendentes del lenguaje Logo: la posibilidad de que el programador se vaya construyendo poco a poco programas y procedimientos que pueda luego usar en muy distintos contextos. Es la forma de inventar y definir nuevas órdenes (instrucciones) del lenguaje. Esto, además de permitir el ahorro de tiempo, acostumbra al no iniciado a trabajar modularmente y, por supuesto, permite acrecentar la creatividad del estudioso.

El lenguaje aporta inicialmente, como es sabido, un gran conjunto de primitivos con los que se pueden construir todos los que hagan falta. Así, si en un momento determinado, el programador hecha de menos alguna instrucción, todo lo que tiene que hacer es poner a funcionar su creatividad e inventiva para escribir los procedimientos que necesite. Este tipo de procedimientos suelen llamarse **utilidades** y deberían conservarse a mano en el disco para que puedan cargarse y usarse cuando sean necesarios.

En esta sección solemos utilizar instrucciones y técnicas no explicadas todavía en el texto. Pretenden servir a los que ya conocen buena parte del lenguaje como ejemplos concretos; a los que van siguiendo el desarrollo de la teoría les invitamos a que los lean, intenten ver por donde ''van las cosas'', y se preparen a volver sobre ellos cuando sus conocimientos se lo permitan. Hacemos notar que las utilidades aquí presentadas, en esta ocasión, llevan una calificación de **tres D**: su construcción es un tanto avanzada.

Los nombres asignados a estos procedimientos se han escrito en mayúsculas para mayor comodidad. En tal caso deberán escribirse de esta forma para ejecutarlos.

1) No dispone Dr. Logo de un primitivo que situe a la tortuga en el centro de la pantalla y mirando hacia arriba pero sin afectar a los gráficos. Todos los demás parámetros de tortuga y pantalla serán dejados en la misma situación que tenían.

```
to HOME
pu
(* levanta la pluma de la tortuga *)
setpos [0 0]
(* situa a la tortuga en el punto de coordenadas (0,0) *)
seth 0
(* hace que la tortuga mire hacia arriba *)
end
```

2) Como es lógico se pueden agregar las ordenes que se deseen si se prefiere que la utilidad vaya un poco más allá. Por ejemplo, el procedimiento anterior transporta a la tortuga sin que esta pinte. Pero entonces la

pluma permanece levantada tras la utilidad. Si preferimos dejar la pluma baja podemos definir así el procedimiento:

```
to HOME.2
pu
setpos [0 0]
seth 0
pd
end
```

3) Claro que lo mejor sería trasladar a la tortuga sin que pinte pero dejar a la pluma en la misma situación que tenía. Para conseguir esto el procedimiento resulta un poco más sofisticado. Lo usaremos como ejemplo cuando veamos la teoría correspondiente.

```
to HOME.3
  (local ''pluma pluma 2)
(* impide que los valores de estas variables interfieran con otras del programa en que se use la utilidad que, eventualmente, se llamaran de la misma forma *).
make ''pluma item 4 tf
(* la lista resultado de la operación tf presenta en el cuarto item el estado de la tortuga; este estado se asigna a la variable ''pluma *)
make ''pluma 2 word (char ascil first: pluma)) + 32) char (ascil (bf: pluma )) + 32
(* esta instrucción se debe a que el resultado de la operación tf está en mayúsculas, siendo las órdenes a la tortuga en minúsculas. La diferencia entre los códigos ASCII de uno y otro tipo de letras es de 32. Las instrucciones involucradas permiten pasar una palabra como por ejemplo ''PD'' que es el contenido de la variable ''pluma, a la forma ''pd'' como contenido de ''pluma 2 *)
pu
setpos [0 0]
seth 0
run (list: pluma 2)
(* esta es una orden avanzada de Logo. Permite ejecutar como instrucciones todas las palabras de una lista. En este caso, dada la palabra contenida en la variable ''pluma 2 (a la que nos referimos como: pluma 2) se forma una lista constituida sólo por esa palabra y después se ejecuta. De esta forma, esta instrucción puede ejecutar distintas cosas según el contenido de la variable ''pluma 2 *).
end
```

4) Siguendo con este tipo de utilidades, nos encontramos con que el primitivo ''cs'', además de borrar los gráficos, situa la pluma en el estado ''pd'', como hemos visto. El siguiente procedimiento sólo se diferencia del primitivo ''cs'' en que deja la pluma de la tortuga en la misma situación que tenía:

```
to CS.2
(local ''pluma'' pluma 2).
make ''pluma item 4 tf
make ''pluma 2 word char (ascil (first: pluma) + 32 char (ascil (bf: pluma)) + 32
cs
run (list: pluma 2)
end
```

Como vemos la idea general es la misma que la del procedimiento anterior.

5) Siempre se puede consultar la lista de números que forman las plumas, inicialmente, si lo que se desea es retornar a sus valores iniciales. Después de esto se pueden ejecutar las órdenes precisas para restaurar los valores. Bueno, puede ser de gran valía disponer de una utilidad que lleve a cabo todas estas acciones por nosotros. El siguiente procedimiento es análogo a CS.2 pero de forma que restaura la asignación de plumas y de colores a sus valores iniciales.

```
to CS.3
(local ''pluma'' pluma 2)
make ''pluma item 4 tf
make ''pluma 2 word char (ascil(first: pluma)) + 32 char (ascil (bf: pluma)) + 32
cs
run (list: pluma 2)
setpal 0 [0 0 1]
setpal 1 [2 2 0]
setpal 2 [0 2 2]
setpal 3 [2 0 0]
setpc 1
end
```

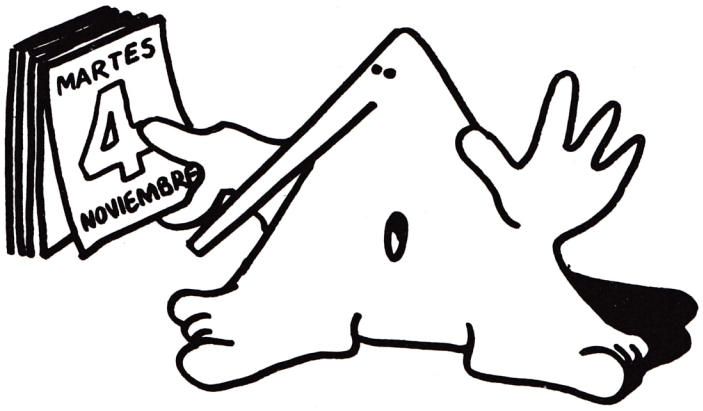
6) De modo enteramente análogo, es posible que en lugar de los colores que propone Dr. Logo como los iniciales, prefiramos asignar inicialmente otros. Basta construir la correspondiente utilidad. Por ejemplo,

```
to CS.4
(local ''pluma'' pluma 2)
make ''pluma item 4 tf
make ''pluma 2 word char (ascii (first: pluma)) + 32 char (ascil (bf: pluma)) + 32
cs
run (list: pluma 2
setpal 0 [0 2 1]
setpal 1 [2 0 0]
setpal 2 [0 2 2]
setpal 3 [0 0 2]
setpc 1
end
```

7) Finalmente y siguiendo con este tipo de utilidades, vamos a ver un procedimiento que además de las acciones realizadas por el último, lleva un input local que permite decidir el número de líneas de texto con que ha de quedar la pantalla mixta.

```
to CS.5: n
(local ''pluma ''pluma 2)
(* no ha de declararse ''n como local porque es ya un input local *)
(f 25<:n [pr [valor inválido] stop]
(* hay que rechazar los valores no válidos *)
setsplit: n
(* asignación de número de filas en pantalla mixta *)
make ''pluma item 4 tf
make ''pluma 2 word char (ascii (first: pluma)) + 32 char (ascii (bf: pluma)) + 32
cs
run (list: pluma 2)
setpal 0 [0 2 1]
setpal 1 [2 0 0]
setpal 2 [0 2 2]
setpal 3 [0 0 2]
setpc 1
end
```


Programa Calendario !



por Víctor J. Campo López

Este programa es válido para el Amstrad CPC 464, 664 y 6128 en su versión 1 y 2, de pantalla e impresora, y para los modelos 664 y 6128 en su versión 3 de copy de pantalla.

Su manejo no puede ser más simple, basta con introducir el año (4 cifras) y el programa nos irá mostrando el calendario, mes a mes.

No debe preocuparse en que día de la semana comienza el año, ni si se trata de un bisiesto, ni nada por el estilo, el programa lo calcula por usted.

VERSION 1.— Esta versión es únicamente de pantalla y es válida para los tres modelos de Amstrad mencionados. Le será muy fácil, si lo desea, cambiar los colores utilizados, e incluso si su monitor es verde, puede prescindir de ellos y obtendrá mejor imagen.

VERSION 2.— Si añade a la versión 1 las líneas de pgm. que le indicamos a continuación, obtendrá simultáneamente el calendario impreso en papel.

```
161 PRINT #8,CHR$(27);"W";CHR$(1)
171 PRINT #8,CHR$(27);"A";CHR$(20)
181 PRINT #8,TAB(3)A$(K);TAB(33)A
202 PRINT #8," L   M   X   J   V   S   D"
212 PRINT #8,"============================"
232 PRINT #8,TAB(D*5);J;
291 PRINT #8: PRINT #8,CHR$(27);"A";CHR$(35)
292 PRINT #8
```

Los códigos de control utilizados pertenecen a la impresora Seikosha 1000A, si dispone de otro tipo y no son compatibles, utilice los correspondientes, teniendo en cuenta que:

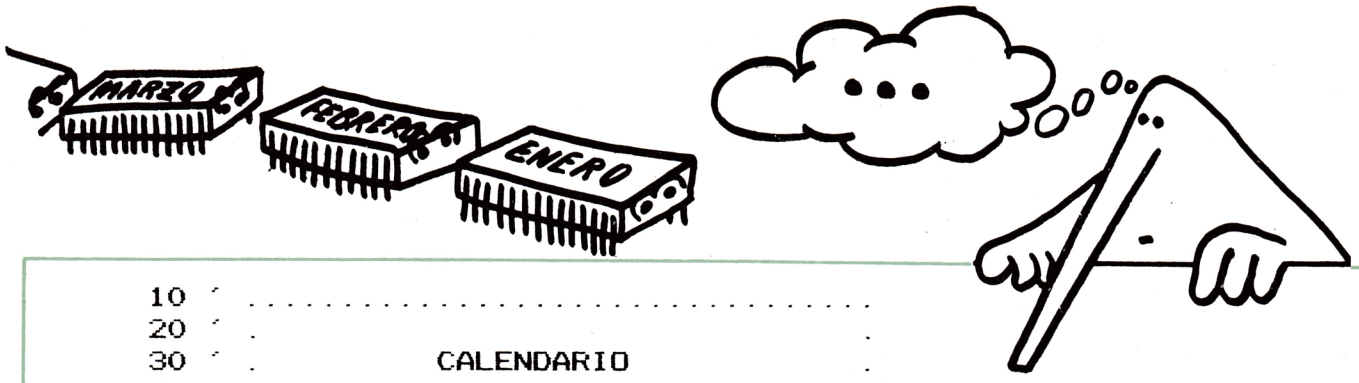
La línea 161 imprime en caracteres expandidos.

La línea 171 efectua un salto de línea de 20/72 pulgadas.

La línea 291 efectua un salto de línea de 35/72 pulgadas.

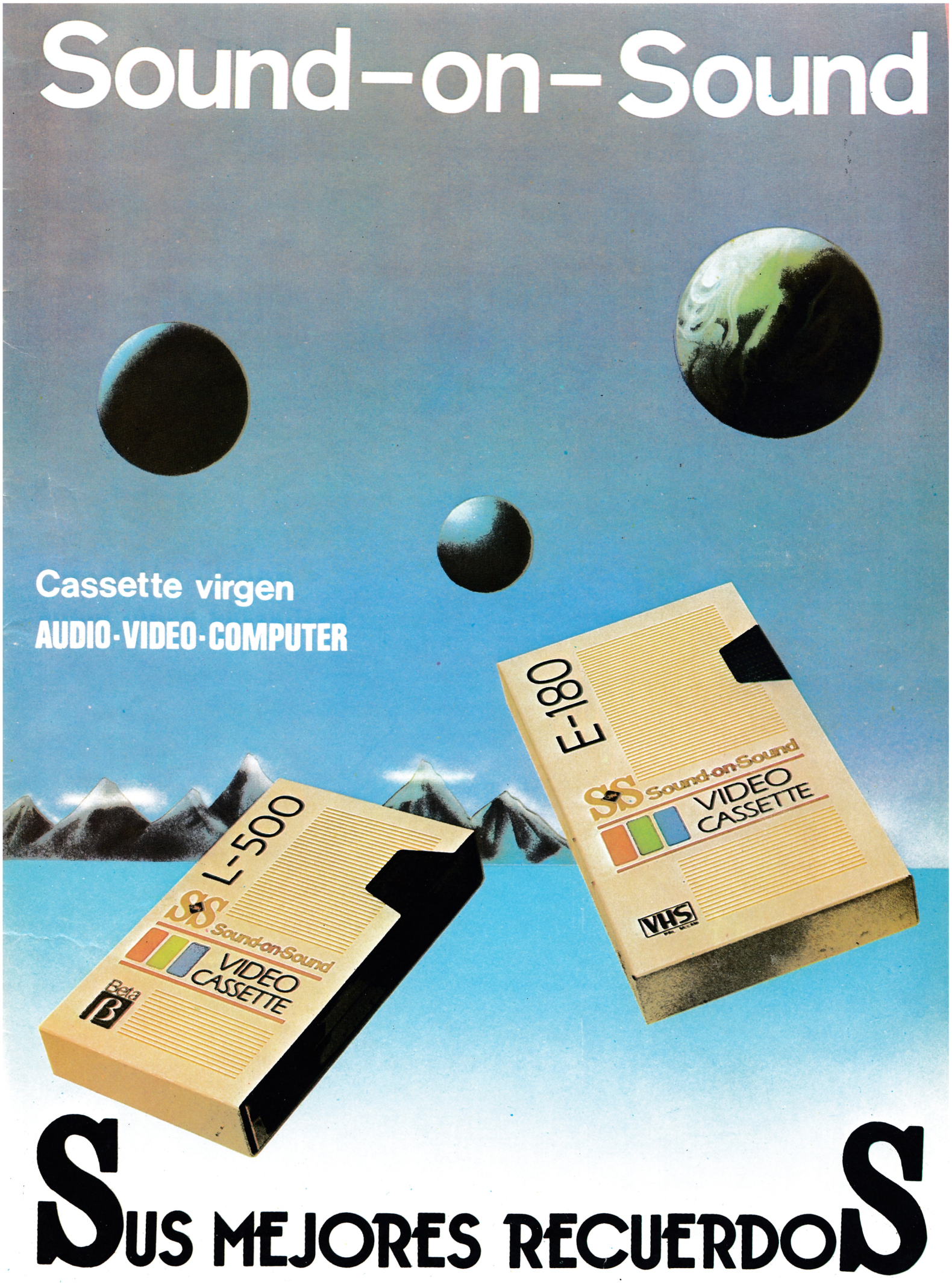
VERSION 3.— Si tiene problemas con su impresora, incluya en el pgm. de la versión 1, las siguientes líneas, que efectuaran un copy de pantalla en la impresora.

```
 265 GOSUB 8000
8000 REM COPY DE PANTALLA PARA 40 COLUMNAS
8010 WIDTH 40
8020 FOR FIL=6 TO 22
8030 FOR COL=1 TO 40
8040 LOCATE COL,FIL
8050 A$=COPYCHR$(#0)
8060 PRINT  8,A$;
8070 NEXT COL,FIL
8080 RETURN
```




```
10 ' ....................................
20 ' .                                  .
30 ' .          CALENDARIO              .
40 ' .                                  .
50 ' ....................................
60 MODE 0: PRINT "C A L E N D A R I O"
65 INK 1,23:INK 0,1
70  LOCATE 7,9: PRINT "_"
80 LOCATE 6,10: PRINT  "ANO "
85  LOCATE 9,10:INPUT a
90 A1=INT(A/100):A2=A-100*A1
100 IF A2<>0 AND (A2-4*INT(A2/4))=0 THEN N=1: GOTO 130
110 IF A2=0 AND (A1-4*INT(A1/4))=0 THEN N=1: GOTO 130
120 N=2
130 C=INT(365.25*A2)+31+N
140 D=C+3-7*INT((C+2)/7)
150 GOSUB 500
160 MODE 1:INK 0,15:INK 1,0:BORDER 19:CLS
170    FOR K=1 TO 12
180  LOCATE 1,8:PRINT "    ";A$(K);TAB(33)A
190 PRINT
200 PRINT "      L     M     X     J     V     S     D"
210 PRINT "========================================="
220    FOR J=1 TO M(K)
230  PRINT TAB(D*5);J;
240 IF D >=7 THEN D=1: PRINT : GOTO 260
250 D=D+1
260 NEXT J
270 LOCATE 3,25: PRINT CHR$(24);" PRESIONE UNA TECLA PARA CO
    NTINUAR ";CHR$(24)
280 Z$=INKEY$: IF Z$="" THEN 280
290 CLS
300 NEXT K
310 MODE 0:INK 0,1:INK 1,24:BORDER 1
315 BORDER 1
320 LOCATE 1,12: PRINT "PARA UN NUEVO CALEN-
                        DARIO PRESIONE <C>"
330 Z$=INKEY$:IF Z$="" THEN  GOTO 330
340 IF Z$="C" OR Z$="c" THEN RUN ELSE MODE 1:END
500 DIM A$(12),M(12)
510 RESTORE 570
520    FOR K=1 TO 12
530 READ A$(K),M(K)
540  NEXT K
550 IF N=1 THEN M(2)=29
560 RETURN
570 DATA "ENERO",31,"FEBRERO",28,"MARZO",31
580 DATA "ABRIL",30,"MAYO",31,"JUNIO",30
590 DATA "JULIO",31,"AGOSTO",31,"SEPTIEMBRE",30
600 DATA "OCTUBRE",31,"NOVIEMBRE",30,"DICIEMBRE",31
```

Sound-on-Sound



SUS MEJORES RECUERDOS

CURSO DE BASIC + MICROORDENADORES

prácticas con...

Microordenador ZX SPECTRUM

Microordenador COMMODORE

Microordenadores AMSTRAD, MSX, PC

Para saber cómo hablar con los ordenadores

El Curso CEAC a Distancia, BASIC + Microordenadores, le va a introducir paso a paso, con un cuidado método, en uno de los temas más apasionantes de nuestros días: **la programación de ordenadores.**

Al aprender PRACTICANDO desde un principio a programar BASIC, lenguaje diseñado especialmente para dar los primeros pasos en programación, estará sentando las bases para el estudio de cualquier otro lenguaje de alto nivel.

Curso CEAC de BASIC + Microordenadores: un diálogo permanente con el ordenador.

Otros Cursos:
- Introducción a la Informática
- Electrónica (con experimentos)
- Contabilidad
- Fotografía
- Curso de Video
- Decoración

ESTAS ENSEÑANZAS SE AJUSTAN AL ART. 35 DEL DECRETO 707/1976 Y A LA ORDEN MINISTERIAL DE 5/2/1979

REF. T-XT

GRATUITAMENTE

Actúe ahora en su propio beneficio y pídanos información.

Sí, deseo recibir a la mayor brevedad posible información sobre el Curso de: _____

_____

Nombre y apellidos _____ Edad ____

Domicilio _____

_____ Nº ____ Piso ____ Pta. ____ Tel. _____

C. Postal _____ Población _____

_____ Provincia _____

Profesión _____

CEAC. Aragón, 472

(Dpto.) 08013 Barcelona

o llame... (93) 245 33 06 de Barcelona

CEAC

CENTRO DE ENSEÑANZA A DISTANCIA AUTORIZADO POR EL MINISTERIO DE EDUCACION Y CIENCIA N.º 8039185

(BOLETIN OFICIAL DEL ESTADO 3-6-83)

Aragón, 472 (Dpto.) 08013 Barcelona

Tel.: (93) 245 33 06