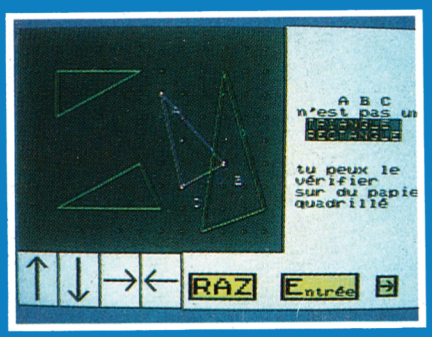
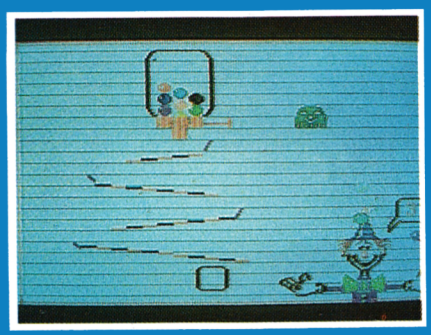
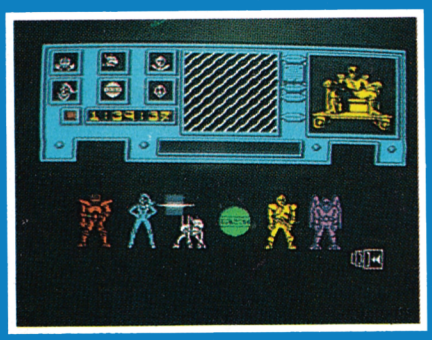
AMSTRAD EDUCATIVO

N.º 7 - 295 Ptas.







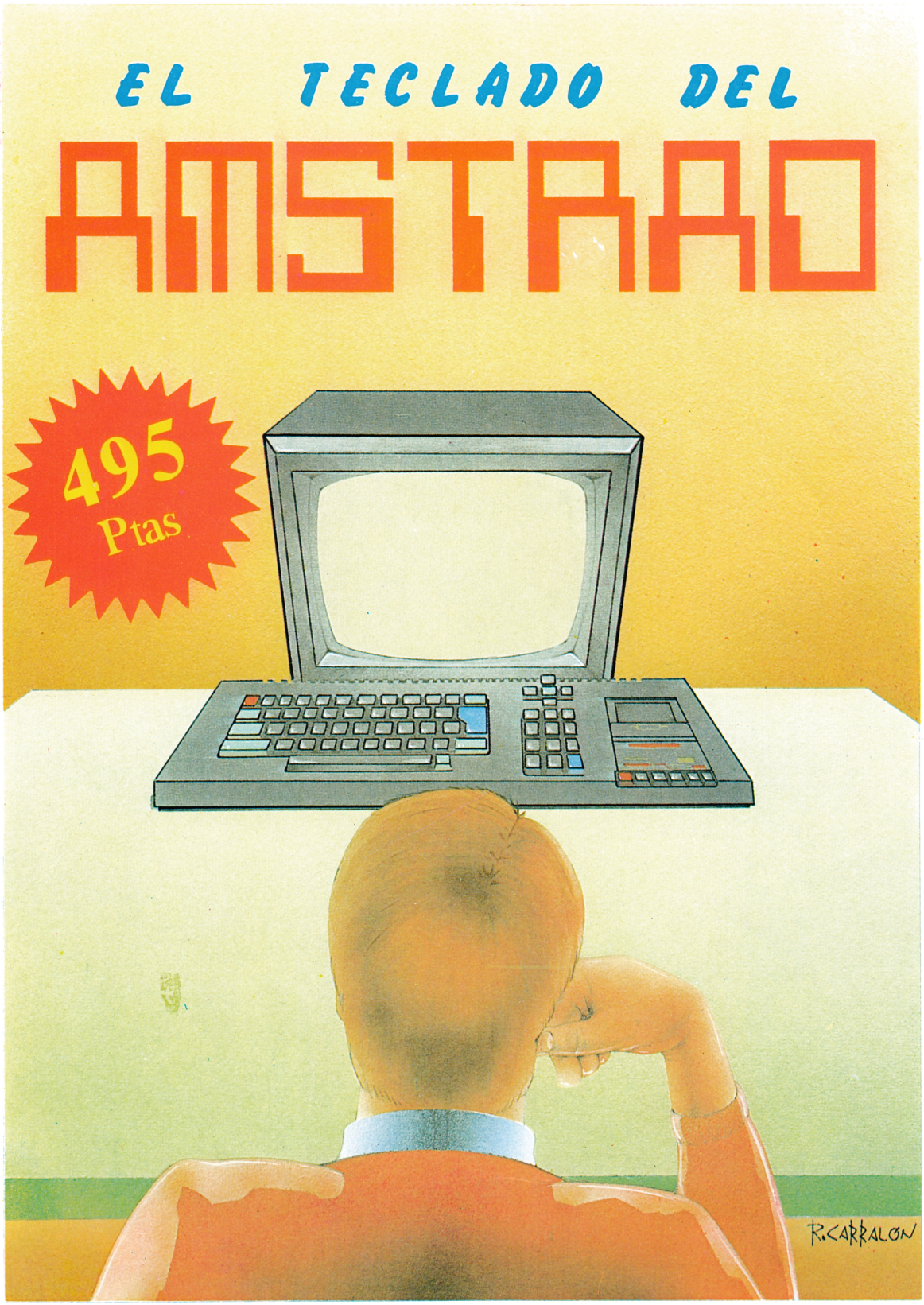
FICHAS DEL AMSTRAD

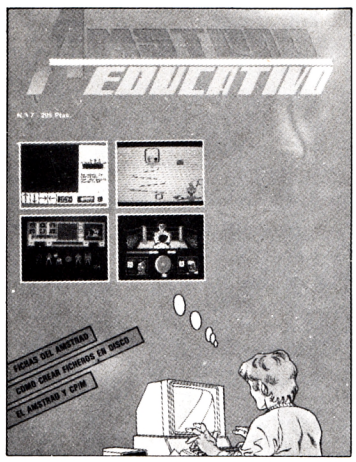
COMO CREAR FICHEROS EN DISCO

EL AMSTRAD Y CP/M

¡ya está a la venta!

EL TECLADO DEL AMSTRAD





Editorial

Los ordenadores están alcanzando tales cotas de difusión que permite incluirlos como un instrumento más de trabajo.
Cada vez, hay también, más periféricos y programas que nos ayudan a sacar más rendimiento de ellos.
Sin embargo, el medio más usual de comunicación entre el usuario y la máquina sigue siendo el teclado, que aunque poco a poco se ha ido ampliando con nuevas variantes —teclas de función, edición, de movimiento de cursor, etc.) es fundamental.
Por esto, en cada número, te invitamos a que practiques y teclees tus programas, hagas experimentos y vayas conociéndole cada día un poco más.
Hasta el próximo mes.

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

SYSGEN (SYSTEM GENERATION: GENERACION DE SISTEMA)

por A. Bellido

Función asignada: Copiar las pistas reservadas al DOS de un disco a otro, generando, de esta forma, un nuevo disco del sistema.

Comentarios: En reiteradas ocasiones se ha dicho que los DOS es un conjunto de programas que este conjunto de programas está dividido en tres subconjuntos —BIOS BDOS, CCP— y que este bloque está contenido, al menos en las dos pistas más externas de cualquier disco del sistema.

También sabemos que SYSGEN es una orden transitoria de CP/M que requiere, para ser ejecutada, la carga del programa SYSGEN.COM en la RAM, con lo cual provoca una copia de las pistas del sistema en el TPA y, posteriormente de éste a un nuevo disco que se convertirá, por ende, en un disco del sistema al recibir en sus pistas externas el bloque BIOS-BDOS-CCP.

Este proceso no afecta para nada a la pista catálogo y al directorio correspondiente, con lo cual si el disco estaba vacío antes, seguirá vacío después, pero con la diferencia de que podremos inicializar el sistema a través del disco, y que, tras el SYSGEN, lo hemos convertido en un disco de sistema.

Forma de obtenerla: Instalar en la unidad de disco por defecto un disco que contenga el programa SYSGEN.COM teclear SYSGEN y, finalmente pulsar <cr>. A partir de este momento CP/M irá mostrando el camino a seguir de una forma similar a ésta:

Ejemplo: Supongamos que nuestro ordenador sólo dispone de una unidad de disco. En este caso el proceso sería:

1.º Instalar en la unidad disponible un disco con SYSGEN.COM y escribir tras la impronta:

A > SYSGEN

Al pulsar <cr> aparacerá:

SYSGEN VERSION 2.0
INSERT SOURCE DISC INTO DRIVE A THEN PRESS ANY KEY:

Esto significa que el operador debe sacar el disco actual y colocar en su lugar uno que contenga las pistas reservadas a GP/M, a no ser, claro está, que el disco inicial sea propiamente un disco del sistema.

2.º Una vez efectuado el cambio de disco, pulsar una tecla, con lo cual un nuevo mensaje de este tipo.
INSERT DESTINATION DISC INTO DRIVE A THEN PRESS ANY KEY:

nos alerta para un nuevo cambio: Ahora debemos sustituir el disco actual por aquel otro que deseamos reconfigurar como disco del sistema.

3.º Al pulsar una tecla cualquiera se producirá la copia de la configuración del DOS actual en el nuevo disco y, terminada ésta, la pantalla nos preguntará:
DO YOU WISH TO RECONFIGURE ANOTHER DISC (Y/N)?:

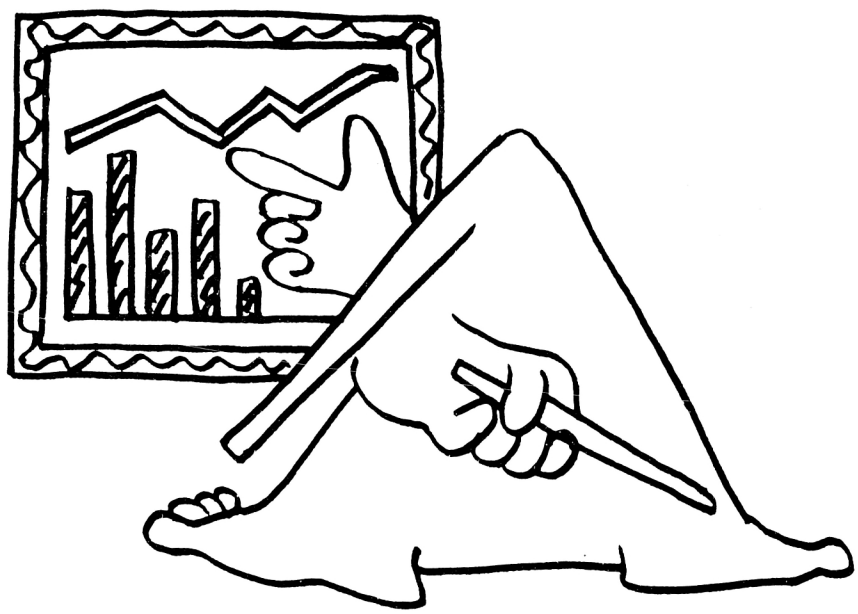
O algo similar, en el sentido de que si queremos reconfigurar otro disco hay que pulsar la tecla Y, en otro caso la N.
Si elegimos la Y se repite todo lo anterior. Si es la N se nos requiere para instalar un disco del sistema:

INSERT A CP/M SYSTEM DISS INTO DRIVE A THEN PRESS ANY KEY:

En este momento, al pulsar cualquier tecla, todo concluirá al aparecer la impronta (A >) indicativa de que el sistema está a nivel operador.

En cualquier instante se puede cancelar el proceso salir de él pulsando C.

Si el ordenador dispusiera de dos unidades de disco los mensajes descrito estarían con una configuración.



Errores posibles:

1.º El más probable se deriva de olvidar colocar el disco que contiene el programa SYSGEN.COM en la unidad de disco, con lo cual al dar la orden SYSGEN, la pantalla mostrará:

A > SYSGEN
SYSGEN?

Este mensaje de CP/M indica al operador que no en-

cuentra el programa correspondiente en la unidad por defecto.

2.º Otro fallo más delicado en su tratamiento se produce cuando en un ordenador con más de una unidad de disco se indica una unidad que no está conectada. Este sería el caso de un computador con dos unidades —A y B— al que se le pide que haga la copia en el disco instalado en la unidad C. CP/M reconoce que tal unidad existe, pero no sabe si físicamente está conectada. Si el operador detecta el error contenido antes que CP/M, debe pulsar C para cancelar la orden y repetir el proceso. Si CP/M interceptará error primero, probablemente el computador se bloquea.

EJERCICIOS CON SYSGEN

1.º ¿Qué función cubre SYSGEN?

Respuesta: Carga el DOS (BIOS/BDOS/CCP) en el TPA de aquí lo transfiere a las 2 pistas externas de un disco convirtiéndolo así en un disco del sistema.

2.º ¿En qué forma se altera un disco conteniendo algún tipo de información tras una orden SYSGEN en la que esté involucrado?

Respuesta: Las pistas externas contendrán, tras SYSGEN, los programas del DOS (BIOS-BDOS-CCP).

3.º ¿Queda modificada la pista catálogo?

Respuesta: No. El directorio no se altera.

4.º ¿Se puede utilizar un disco debidamente ''formateado'' para contener datos, sin que contenga el DOS en sus pistas externas?

Respuesta: Sí.

5.º ¿Qué se puede y qué no se puede hacer con un disco como el descrito en la cuestión anterior?

Respuesta: Se puede leer y escribir información, pero no se puede utilizar para inicializar el sistema.

6.º ¿Qué sucede si aplicamos la orden SYSGEN sobre un disco del sistema?

Respuesta: El disco del sistema ya tiene grabado en sus pistas externas los programas del DOS; por tanto, un SYSGEN volvería a grabar lo mismo encima otra vez. Conclusión: Nada varía.

7.º Interpretar ésta orden: B:SYSGEN.

Respuesta: Estamos llamando a SYSGEN. COM desde el disco instalado en la unidad B.

8.º Interpretar las dos líneas siguientes:

A > B:SYSGEN
SYSGEN?
A >

COMO CREAR FICHEROS EN DISCO

por A. Bellido

FICHEROS SECUENCIALES. INSTRUCCIONES

Antes de pasar a confeccionar un fichero secuencial, es coveniente conocer las instrucciones que permiten la manipulación de este tipo de ficheros, y sus implicaciones.

OPENOUT ''nombre''

Esta orden abre (OPEN) un fichero secuencial de nombre génerico ''**nombre**'' que permite la salida (OUR) de información hacia el disco.

OPENIN''nombre''

Esta orden abre (OPEN) un fichero secuencial de nombre genérico ''**nombre**'' que permite la entrada (IN) de información procedente del disco.

NOTA 1: Para transferir información entre la RAM y un fichero en disco se debe comenzar por abrir la comunicación mediante la instrucción OPEN:
Si la información va a salir de la RAM se utilizará el indicativo OPENOUT (por **output:** Salida hacia el disco, equivalente a la expresión ''escribir en el fichero''.
Si la información va a entrar en la RAM se utilizará el indicativo OPENIN (por **input:** Entrada desde el disco, equivalente a la expresión ''leer en el fichero''.

NOTA 2: Al ser ejecutada una instrucción OPEN queda establecido en la RAM un **buffer** del registro del fichero en cuestión, y el tipo de acceso al mismo.

NOTA 3: Con OPEN se establece una asociación entre el nombre del fichero, el número de canal a través del cual va a circular la información y el **buffer** correspondiente, de tal forma que las posturas de emisor y receptor las adoptarán bien el fichero o bien el **buffer, siendo el canall asociado al único posible.**

NOTA 4: Todo intento de abrir un fichero ya abierto termina con un mensaje del tipo ''Fichero ya abierto'', en el sentido de que el fichero ya fue abierto.

NOTA 5: Si, por el contrario, el fichero no fue abierto y se le solicita en cualquier forma, se obtiene un mensaje similar a: ''Fichero equivocado'', indicativo de esta situación.

Ejemplo: Abrir, en modo directo, un fichero secuencial denominado PRUEBA para permitir el envío de información hacia el fichero.
Las premisas impuestas exigen teclear la siguiente orden:

OPENOUT ''PRUEBA''

Al pulsar RETURN el fichero PRUEBA ha sido abierto y está a la espera de recibir información.

CLOSEIN

Con esta orden se cierra cualquier fichero abierto por medio de OPENIN.

CLOSEOUT

Con esta orden se cierra cualquier fichero abierto por medio de OPENOUT.

NOTA 1: *Al ser ejecutada una instrucción CLOSE se cancelan las asociaciones establecidas entre los canales, los* **buffers** y los nombres de ficheros correspondientes.

NOTA 2: *Los contenidos de los* **buffers** abiertos para salida de información que graban en sus ficheros correspondientes en el momento de ejecutarse el CLOSE oportuno.

PRINT , expresiones

Esta instrucción ordena el envío de las **expresiones** a través del canal 9 para su grabación en el fichero asociado a tal canal.

NOTA 1: *Al ser ejecutada una orden PRINT, el envío y grabación de las* **expresiones** *dadas en la instrucción se efectúa a imagen de la impresión que se obtendría en pantalla de esas* **expresiones** *por una instrucción PRINT convencional. Por esta razón, el programador debe elegir entre la coma (,) y el punto y coma (;) en el momento de separar unas* **expresiones** *de otras en la instrucción PRINT.*

NOTA 2: *La orden PRINT sólo puede ser utilizada si el fichero fue abierto en modo OPENOUT (***OPENOUT:** *Salida de información hacia el fichero).*

NOTA 3: *De hecho, la información a grabar en el fichero, no se envía directamente al disco, sino que pasa al* **buffer** asociado, el cual procede a la emisión sólo cuando está lleno o cuando se ejecuta la orden de cierre (CLOSE) del fichero, como ya se indicó.

Ejemplo: Supuesto abierto el fichero PRUEBA en modo directo mediante la orden:

OPENOUT''PRUEBA''

Vamos a probar los diferentes efectos que se pueden obtener mediante la grabación de información gracias a PRINT.

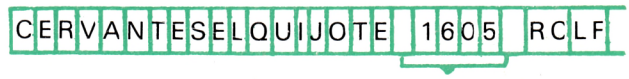
Todas las órdenes se dan en modo directo.

Alternativa 1

Orden de grabación de varias expresiones en un solo registro separadas por puntos y comas:

PRINT 9 ''CERVANTES'';''EL QUIJOTE''; 1605

Esta instrucción provocará em su momento, la grabación de contenido único.



RC.-Retorno de Carro (código ASCII 13)
LF.-Línea siguiente (código ASCII=10)

NOTA: *El contenido de este registro se considera a todos los efectos como una única cadena de caracteres)*
(Ver ''FICHEROS: DECISIONES PREVIAS'')

Alternativa 2:

Orden de grabación de varias expresiones en un sólo registro por comas:

PRINT 9, ''CERVANTES'', ''EL QUIJOTE'',1605

Esta instrucción provocará, en su momento, la grabación de un registro de contenido único:

En el supuesto claro está, de que las zonas de tabulación sean como la indicadas.

NOTA: *El contenido de este registro se considera a todos los efectos como una única cadena de caracteres.*

En ambas alternativas, los números quedan grabados como los caracteres correspondientes a los dígitos que los integran precedidos y sucedidos por sendos espacios (código ASCII 32)

Alternativa 3:

Orden de grabación de varias expresiones en un sólo registro separadas por el **separador** '',''

PRINT 9 ''CERVANTES'';'';''EL QUIJOTE;'';''1605

Esta instrucción provocará, en su momento, la grabación de un registro de contenido múltiple:

NOTA: *El contenido de este registro se considera a todos los efectos como dos cadenas de caracteres: ''CERVANTES'' y ''EL QUIJOTE''. Con respecto a 1605 puede ser asociado a una cadena de caracteres compuesta por los caracteres 1,6,0 y 5, o por el contrario al número indicado.*

Alternativa 4:

Orden de grabación de varios números en un solo registro, separados por punto y comas:

PRINT 9, 1605, 1615

Esta instrucción provocará, en su momento, la grabación de un registro de contenido.

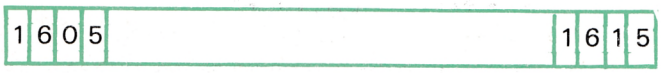
NOTA *El contenido de este registro puede ser asociado a dos números independientes: el 1605 y el A1615. O por el contrario en una sola cadena compuesta por los caracteres, 1,6,05, espacio, espacio, 1,6,1 y 5.*

Alternativa 5;

Orden de grabación de varios números en un solo registro separados por comas:

PRINT9,1605,1615

Esta instrucción provocará, en su momento, la grabación de un registro de contenido.



14 posiciones de caracter correspondientes a la 1.ª zona de tabulación.

NOTA: *El contenido de este registro puede ser asociado a dos números independientes: el 1605 y el 1615. O por el contrario a una sola cadena compuesta por los caracteres 1,6,0,5, seguidos de nueve espacios y, a continuación, 1,6,1 y 5.*

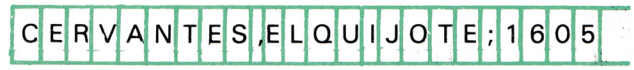
Todo esto claro está en el supuesto de que las zonas de tabulación sean de catorce posiciones de caracter.

Alternativa 6:

Orden de grabación de una expresión en un solo registro contenido dentro de si comas, punto y comas, punto y comas y espacios iniciales:

PRINT9 CHN$(34)''CERVANTES, EL QUIJOTE;1605''

Esta instrucción provocará, en su momento, la grabación de un registro de contenido:



NOTA: *El contenido de este registro se considera a todos los efectos como una cadena que incluye estre sus caracteres una coma y un punto en los lugares donde aparecen.*

Todas las cadenas de caracteres y números que han aparecido hasta el momento podrían haber estado representadas por las oportunas variables alfanuméricas.

Visto el modo de proceder para escribir información en un fichero, pasemos a leerla. Para recuperar la información grabada en un fichero secuencial hay que recurrir a la instrucción INPUT , tras abrir la comunicación con el indicativo OPEINN.

INPUT 9 variables

Esta instrucción ordena la recuperación de información grabada en un fichero secuencial asociándola a las **variables.**

Las **variables** deben ir separadas por comas (,).

NOTA 1: *Las **variables** especificadas en este orden deben ser del mismo tipo que los datos que les van a ser asignados tras la lectura del fichero. Por ejemplo: Si el dato a recuperar es una cadena de caracteres, la variable ha de ser alfanumérica.*

NOTA 2: *El formato con que será recuperada la información contenida en un fichero secuencial mediante INPUT depende del criterio que se haya seguido al grabrarla, según se mostró en las diferentes alternativas ofrecidas por el PRINT.*

NOTA 3: *Al recuperar datos numéricos, se ignoran los espacios que preceden a los números comenzando la lectura en el primer caracter que encuentra y finalizando al encontrar otro espacio, un caracter RC o una coma.*

NOTA 4: *Al recuperar datos alfanuméricos, se ignoran los espacios que preceden al primer caracter, comenzando en éste la lectura y finalizando al encontrar un RC, una coma (como caracter independiente) o tras 255 caracteres.*

Ejemplos: Para seguir la serie de ejemplos que se ofrecen a continuación el lector deberá abrir el fichero ''PRUEBA'' en modo OPENOUT, grabar la información

que se indica, cerrar el fichero, volverlo a abrir en mod OPENIN, recuperarla y, finalmente, ordenar la impresión en pantalla del contenido de las variables utilizadas. Y todo ello en modo directo.

La secuencia de aternativas que se ofrecen aquí corresponden a la seguida al estudiar PRINT.

Alternativa 1:

Grabación y recuperación de varias expresiones situadas en un sólo registro separadas en puntos y comas:

```
OPENOUT ''PRUEBA''
PRINT   9, ''CERVANTES'';''EL QUIJOTE''; 1605.
CLOSE
OPENIN ''PRUEBA''
INPUT   9, AS
PRINT A$
CLOSE
```

El resultado de estos mandatos es la cadena:

```
CERVANTES EL QUIJOTE 1605
```

Lo cual es lógico si se considera que la información contenida en el registro comienza con un caracter, el C, y concluye al encontrar un RC sin que aparezca ninguna coma como caracter independiente.

Alternativa 2:

Grabación y recuperación de varias expresiones situadas en un solo registro separadas por comas:

```
OPENOUT ''PRUEBA''
PRINT 9, ''CERVANTES'', ''EL QUIJOTE'', 1605.
CLOSE
INPUT   9,                                  A$'
CLOSE
```

El resultado de estos mandatos es la cadena:

```
CERVANTES          EL QUIJOTE          1605
```

Que se corresponde con la imagen grabada según muestra el esquema de la alternativa 2 de PRINT.

Alternativa 3:

Grabación y recuperación de varias expresiones situadas en un sólo registro separadas por el **separador** ,.

```
OPENOUT ''PRUEBA''
PRINT   9,''CERVANTES'';'',''''EL QUIJOTE'',',1605
CLOSE
OPEN ''PRUEBA''
INPUT A$, B$, C$
CLOSE
```

En esta ocasión, por aparecer en el registro comas (,) grabadas como caracteres independientes, la información no corresponde a una sola caena de caracteres, razón por la cual la recuperación requiere más de una variable tras la orden INPUT.

Por razones que se deducen de las notas 2, 3, y 4 expuestas más arriba, la expresión 1605 purde ser considerada como un número o una cadena. Esto se puede comprobar mediante la repetición de los tres últimos mandatos de la secuencia anterior cambiando la variable C$ por la numérica C, de esta forma:

```
INPUT   9,A$,B$,C
PRINT                                   A$,B$,C'
CLOSE
```

En este punto es conveniente observar que si no se abre el fichero nuevamente y se intenta el INPUT , A$,B$,C se obtiene un mensaje del tipo ''Input pass end'' indicativo de que se ha superado el último registro y con él el fin del fichero, lo cual es lógico si se considera que por estar trabajando en un fichero secuencial la lectura de registros se produce uno tras otros, a no ser que se tenga la precaución de cerrarlo y abrirlo nuevamente con lo cual la secuencia de lectura comienza por el primero otra vez.

Alternativa 4;

Grabación y recuperación de varios números en un sólo registro separados por punto y comas:

```
OPENOUT ''PRUEBA''
PRINT   9, 1605; 1615
CLOSE
OPEN''I''   *1, ''PRUEBA''
INPUT*1,N,N
PRINT N,N
CLOSE
```

La sentencia INPUT utilizada aquí ordenada de los números 1605 y 1615 conforme a lo indicado en la nota 3. Pero también se podría haber pensado en ordenar la lectura de la cadena 1605 y 1615 mediante la orden INPUT 1, A$, según lo que se deduce.

Alternativa 5:

Grabación y recuperación de una expresión en un solo registro dentro de sí comas y punto y comas:

```
OPENOUT ''PRUEBA''
PRINT   9,CHRS (34);''CERVANTES,EL QUIJO-
TE;1605''CHR$(34)
CLOSE
INPUT   9,A$
CLOSE
```

Estos mandos concluyen con la impresión en pantalla de:

```
CERVANTES, EL QUIJOTE;1605
```

Como es lógico, en función de lo dicho en la nota 5.

WRITE 9, expresiones

Esta instrucción ordena el envío de las **expresiones** a través del canal **n** para su grabación en el fichero asociado a tal canal.

El valor de **n** debe ser el mismo dado en la orden OPEN que abrió el fichero involucrado.

NOTA 1: *Las diferencias existentes entre WRITE y PRINT son:*

- *WRITE inserta el caracter coma entre las* **expresiones** *de forma que son consideradas como expresiones independientes.*
- *WRITE incorpora comillas iniciales y finales a las expresiones alfanuméricas.*
- *WRITE no coloca un espacio en blanco delante de los números.*

Ejemplo: Grabación y recuperación de varias expresiones por medio de WRITE y INPUT

```
OPENOUT ''PRUEBA''
WRITE  9, 1605, ''EL QUIJOTE''
CLOSE
1605 EL QUIJOTE
```

LINE INPUT, 9 variable alfanumérica

Esta instrucción ordena la recuperación de información grabada en un fichero secuencial correspondiente a un registro completo (hasta encontrar un RC) asociándola a la **variable alfanumérica.**

Ejemplo: Admitiendo que el primer registro del fichero PRUEBA contenga: CERVANTES, EL QUIJOTE; 1605, se puede ordenar la recuperación de esta información.

```
OPENIN ''PRUEBA''
LINE INPUT   9, A$
CLOSE
PRINT A$
```

Lo cual dará esta impresión en pantalla:
CERVANTES, EL QUIJOTE, 1605.
Así: FILES''d: siendo **d** el nombre de la unidad elegida.

EOF

Esta función devuelve —1 (cierto) si se ha alcanzado el fin del fichero **(end of file)**
De otra forma devuelve 0 (falso)

NOTA 1: *La función EOF solo tiene sentido para los ficheros secuenciales abiertos en modo lectura (1).*

Tras esta revisión general de las órdenes principales que afectan a los ficheros secuenciales, pasemos a utilizarlas.

REM | REM aclaraciones | REMEMORAR

INTERPRETACION

Significa *rememorar, recordar* y permite introducir aclaraciones y observaciones que pueden ser de interés en una posterior revisión o lectura del listado del programa.

La sentencia REM no implica ningún tipo de proceso.

POSIBILIDADES

Como se desprende de la nomenclatura utilizada — aclaraciones— el texto que se utilice para las aclaraciones y observaciones es opcional. Esto quiere decir que si tecleamos REM en una línea de programa —bien a su comienzo bien tras el separador (:)—, ésta quedará impresa para servir de indicador nemotécnico. En ese sentido, es frecuente utilizar el separador (:) como medio de fragmentar un listado en ''trozos'' físicos para distinguir de un vistazo los diferentes elementos de un programa.

En todo caso, la sentencia REM se mantiene en el Estado, tal como se ha escrito, sin ser procesada, continuando la ejecución del programa en la primera sentencia de la siguiente línea del programa.

FORMA DE TECLEAR LA INSTRUCCION

Teclear **REM** y las aclaraciones, seguido de **ENTER**.

EJEMPLO

Diferentes formas de introducir aclaraciones y observaciones en un listado:

PROGRAMA	COMENTARIO
10 REM Comienzo del programas.	**REM** con aclaraciones.
20 INPUT A	
30 REM	**REM** en solitario.
40 INPUT B	
50	Separadores para fragmentar el listado.
60 REM Suma	**REM** con aclaraciones.
70 LET C = A + B: REM contador	**REM** tras un separador.
80 REM Fin	**REM** antes de un separador.
90	Separadores en forma de línea.

STOP/CONT | STOP/CONT | PARAR/CONTINUAR

INTERPRETACIÓN

Cuando la lectura del programa llega a la instrucción **STOP**, finaliza la ejecución de mismo, pasando el computador a la situación de disponible.

Si el programa tiene más instrucciones después de **STOP**, para proseguir su ejecución es necesario teclear **CONT** seguido de ENTER.

POSIBLIDADES

Cuando el programa detiene su ejecución, como consecuencia de un **STOP**, siempre existe la posibilidad

de continuarla si se teclea —como se ha dicho más arriba—, antes que cualquier otra cosa, el comando **CONT** seguido de **ENTER**.

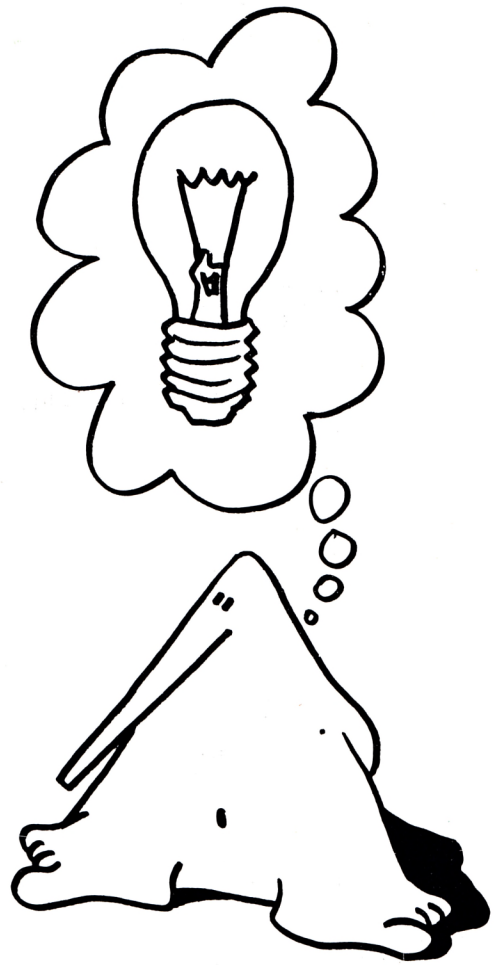
FORMAS DE TECLEAR LAS INSTRUCCIONES

Teclear **STOP**, seguido de **ENTER**, para *detener* la ejecución del programa.

Teclear **CONT**, seguido de **ENTER**, para *continuar* la ejecución del programa detenido por un **STOP**.

EJEMPLO

PROGRAMAS	COMENTARIOS
10 INPUT A **20 PRINT A** **30 INPUT B** **40 PRINT B** **50 PRINT A+B** **60 STOP** **70 PRINT (A+B)*2** **80 STOP**	Una vez introducidos por el teclado los dos valores (números) que se solicitan en las variables de las líneas 10 y 30, el programa detendrá su ejecución en la línea 60 imprimiendo el resultado de la suma. Si tecleamos **CONT/ENTER**, seguirá la ejecución del programa imprimiendo el resultado de la operación de la lína 80, donde hay otro **STOP**.



RECUERDE

- *Las posibilidades de utilización de los comandos PRINT, LET e INPUT.*
- *Cómo se utiliza el comando LIST para listar en la pantalla el programa que el ordenador tiene en su memoria.*
- *La conveniencia de utilizar el separador (:) para fragmentar un programa en sus diferentes partes.*
- *Que el comando REM no afecta el programa, ya que el ordenador no lo tiene en cuenta en la ejecución.*
- *Que el comando REM sirve para introducir aclaraciones en un programa.*
- *Que las sentencias REM de un programa sólo pueden verse al hacer un listado del mismo en la pantalla.*
- *Que el comando STOP sirve para finalizar la ejecución de un programa o de una parte del mismo.*
- *Que el comando CONT sirve para proseguir la ejecución de un programa.*

EJERCICIOS:

Practicar sobre los programas realizados anteriormente:

1. Añadir aclaraciones y observaciones mediante el comando REM.
2. Utilizar separadores para fragmentar los listados.
3. Añadir sentencias STOP para separar las partes de un programa. Por ejemplo, en aquellos que utilizan sucesivamente varias operaciones combinadas: costes, ahorro, áreas, volúmenes, etc.

GO TO | GO TO número de línea | IR A

INTERPRETACION

Se ordena un salto incondicional a un número de línea.

POSIBILIDADES

La secuencia de lectura de un programa, por parte del ordenador, sigue un orden creciente, yendo del número de línea más pequeño al mayor, de una forma sucesiva. Pero cuando a lo largo de esta lectura se encuentra con la orden GO TO, inmediatamente, el orden normal de la lectura se transfiere a la línea cuyo número está indicado en el número de línea.

FORMA DE TECLEAR LA INSTRUCCION

Teclear **GO TO y el número de línea,** seguido de **ENTER,** para ordenar que la secuencia de lectura del programa *siga en la línea del número dado.*

EJEMPLO

1. Hagamos utilizable indefinidamente el programa que determinaba el importe de una conferencia.

PROGRAMA	COMENTARIO
10 INPUT "¿Números de paso?": N **20 INPUT "¿Valor de un paso?":P** **30 PINTO "El importe de su conferencia es "N * P** **40 GO TO 10**	Al añadir la línea 40, obligamos seguir la lectura del programa a partir de la línea 10 tantas veces como queremos.

RECUERDE

- *Que el comando GO TO número de línea, produce un salto incondicional en la ejecución del programa a la línea cuyo número hemos tecleado. En decir, altera el orden en la secuencia de lectura de un programa.*
- *Que este salto puede ser hacia una línea anterior o posterior que aquélla donde está el comando GO TO. Es decir, a una línea de mayor o menor número que la de GO TO.*
- *Las variables situadas en líneas de programa posterios a un GO TO No son memorizadas hasta que la secuencia de lectura no obligue a pasar por esas líneas.*
- *Que se puede emplear el comando GO TO para hacer utilizable indefinidamente un programa.*

EJERCICIOS

Utilizando los comandos conocidos (PRINT, LET, INPUT y GO TO):

1. Escribir programas para resolver problemas referidos al propio entorno en os que intervengan las operaciones aritméticas:

   1.1. Numeración:

   Contar y descontar de 2 en 2, de 3 en 3, etc., desde un número dado.

   1.2. Cálculo:

   Confeccionar tablas de sumar, restar, multiplicar y dividir.

   1.3. Costes:

1.3.1. Calcular el valor de N unidades de una mercancía sabiendo el precio P de una unidad.

1.3.2. Calcular el coste total de N unidades de una mercancía, al precio P por unidad y de M unidades de otra, el precio O por unidad.

1.3.3. 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.

1.4. Ahorro:

1.4.1. Calcular el dinero que le sobra a una persona si tiene X y gasta Y.

1.4.2. Calcular el dinero que le sobra a una persona si tiene X y gasta Y en diversiones y Z en libros (*).

1.4.3. 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 transporte y diversiones.

1.5. Regla de interés:

1.5.1. Hallar el interés I producido por un capital C, colocado al rédito R por ciento, duante el tiempo T.

1.5.2. Hallar el capital C, el rédito R ó el tiempo T, conocidos los otros datos (*).

1.6. Regla de descuento:
recorre el espacio E en el tiempo T.

1.7 Areas:

1.7.1. Hallar áreas de polígonos y figuras circulares.

1.7.2. Hallar áreas laterales y totales de poliedros y cuerpos redondos.

1.7.3. Hallar la base, la altura o la apotema de un polígono conociendo su área y otros datos (*)

1.8. Volúmenes:

1.8.1. Hallar volúmenes de poliedros y cuerpos redondos.

1.8.2. Hallar una dimensión de un poliedro o cuerpo redondo conocidos el volumen y otras dimensiones (*)

1.9 Móviles:

1.9.1. Hallar la velocidad V de un móvil que recorre el espacio E en el tiempo t.

1.9.2. 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 combina das variables numéricas y de caracteres:

2.1. Calcular la edad de una persona sabiendo el año en que nació y el año actual.

2.2. Calcular los días transcurridos desde el comienzo del curso escolar y los días que falta para que termine (*)

2.3. Calcular las horas de clase a lo largo de todo el curso escolar.(*)

3. Idear sus propios programas utilizando los comandos conocidos: PRINT, LET, INPUT y GO TO.

NOTA: Recuerde que debe comparar estos programas con los del capítulo anterior. Podrá observar la ventaja de introducir el comando GO TO.

SOLUCIONES

1.1.

```
  1 REM Ficha: '' GO TO '' Ej. 1.1.
 10 PRINT ''CONTAR DE N EN N''
 20 PRINT ''Comienzo: '';
 30 INPUT C
 40 PRINT C
 50 PRINT ''Número a sumar: '';
 60 INPUT N
 70 PRINT N
 80 PRINT C+N
 90 LET C=C+N
100 GO TO 80
```

1.2.

```
  1 REM Ficha: ''GO TO '' Ej.1.2.''
 10 PRINT ''TABLA DE MULTIPLICAR''
 20 LET A=1
 30 PRINT ''Número: ''
 40 INPUT N
 50 PRINT N; por '';
 60 PRINT A;'' = '';A*N
 70 LET A=A+1
 80 GO TO 50
```

1.3.1.

```
 1 REM Ficha; " GO TO " Ej. 1.3.1."
 5 PRINT "UNIDADES POR PRECIO"
10 PRINT "Unidades: ";
20 INPUT N;
30 PRINT N
40 PRINT "Precio Unidad: ";
50 INPUT P
60 PRINT P
70 PRINT "TOTAL: ";N * P
80 PRINT
90 GO TO 10
```

1.3.2.

```
 1 REM ficha: "GO TO " Ej. 1.3.2."
 5 PRINT "CONJUNTO DE UNIDADES POR
PRECIO"
10 PRINT "Unidades A: ";
20 INPUT N;
30 PRINT N
40 PRINT "Precio Unidad A: ";
50 INPUT P
60 PRINT P
70 PRINT "COSTE UNIDADES 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 UNIDADES B: ";N1 * P1
150 PRINT
160 PRINT "COSTE TOTAL:
";N * P + N1 * P1
170 PRINT
180 GO TO 10
```

1.3.3.

```
 1 REM Ficha: "GO TO " Ej. 1.3.3."
 5 PRINT "CALCULO DE SALDOS DE
COMPRAS"
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 UNIDADES 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 UNIDADES 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
220 PRINT
230 GO TO 10
```

1.4.1.

```
 1 REM Ficha: "GO TO "Ej. 1.4.1"
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-6
80 GO TO 10
```

1.4.3.

```
 1 REM Ficha: "GO TO" Ej. 1.4.3"
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 alimentación: ";
80 INPUT X
90 PRINT X
100 PRINT "Gastado en vestuario: ";
110 INPUT Y
120 PRINT Y
130 PRINT "Gastado en Transportes y diver-
siones: ";
140 INPUT Z
150 PRINT Z
160 PRINT "Número de meses: ";
170 INPUT M
180 PRINT M
190 PRINT "AHORRO: ";-V * M-X * M-
Z * M + S * M
200 PRINT
210 GO TO 10
```

1.5.1.

```
 1 REM Ficha: "GO TO " Ej. 1.5.1.
10 PRINT "REGLA DE INTERES"
20 PRINT "Capital: ";
30 INPUT C
40 PRINT C
50 PRINT "% anual: ";
60 INPUT R
70 PRINT R
80 PRINT "Tiempo en días: ";
90 INPUT T
100 PRINT T
110 PRINT "Interés: ";C * R * T/36000
120 PRINT
130 GO TO 20
```

1.7.1.

```
 1 REM Ficha: "GO TO " Ej.1.7.1."
 2 PRINT "AREA DE POLIGONO REG.
INSCRITO"
```

```
 5 PRINT ''Descomponer en triángulos''
10 PRINT ''Longitud del lado: '';
20 INPUT L
30 PRINT L
40 PRINT ''Longitud de lqa apotema: '';
50 INPUT A
60 PRINT A
70 PRINT ''Número de lados: '';
80 INPUT N
90 PRINT N
100 PRINT ''AREA: '';L * A/2 * N
110 PRINT
120 GO TO 10
```

```
 1 REM Ficha:'' GO TO '' Ej. 1.7.1.''
 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
80 GO TO 10
```

1.7.2.

```
 1 REM Ficha:'' GO TO '' Ej. 1.7.2.''
 2 PRINT ''VOLUMEN DE PIRAMIDES''
10 PRINT ''Area de la base: '';
20 INPUT S
30 PRINT S
40 PRINT ''Altura: '';
50 INPUT A
60 PRINT A
70 PRINT ''VOLUMEN: '';S * A/3
80 PRINT
90 GO TO 10
```

```
 1 REM Ficha: '' GO TO '' Ej. 1.7.2''
 2 PRINT ''AREA DE CILINDROS''
10 PRINT ''Longitud del radio: '';
20 INPUT R
30 PRINT R
40 PRINT ''Longitud de altura: '';
50 INPUT A
60 PRINT A
70 PRINT ''AREA LATERAL DEL CILINDRO:
'';2 * PI * R * A
80 PRINT
90 GO TO 10
```

```
 1 REM Ficha: ''GO TO ''Ejj.1.7.2''
 2 PRINT ''AREA DE CILINDROS''
10 PRINT ''Longitud del radio: '';
20 INPUT R
30 PRINT R
40 PRINT ''Longitud de altura: '';
50 INPUT A
60 PRINT A
70 PRINT ''AREA TOTAL DEL CILINDRO:
'';2*PI*R*A+4*PI*R
80 PRINT
90 GO TO 10
```

```
 1 REM Ficha:'' GO TO ''Ej.1.7.2''
 2 PRINT ''SUPERFICIE DE LA ESFERA''
10 PRINT ''Longitud del diámetro: '';
20 INPUT D
30 PRINT D
40 PRINT ''SUPERFICIE: ''PI*D2
50 PRINT
60 GO TO 10
```

1.8.1

```
 1 REM Ficha: ''GO TO ''Ej.1.8.1''
 2 PRINT ''VOLUMEN DE PRISMAS''
10 PRINT ''Area de la base: '';
20 INPUT S
30 PRINT S
40 PRINT ''Altura de la figura: '';
50 INPUT A
60 PRINT A
70 PRINT ''VOLUMEN: ''; S*A
80 PRINT
90 GO TO 10
```

```
 1 REM Ficha:'' GO TO '' Ej. 1.8.1''
 2 PRINT ''VOLUMEN DE LA ESFERA''
10 PRINT ''Longitud del diámetro: '';
20 INPUT D
30 PRINT D
40 PRINT ''VOLUMEN: ''PI/6*D3
50 PRINT
60 GO TO 10
```

1.9.1

```
 1 REM Ficha: ''GO TO '' Ej. 1.9.1.''
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
```

2.1.

```
 1 REM Ficha: '' GO TO '' Ej. 2.1.''
10 PRINT ''EDAD APROXIMADA DE UNA
PERSONA''
20 PRINT ''Año de nacimiento: '';
30 INPUT A
40 PRINT A
50 PRINT ''Año actual: '';
60 INPUT A1
70 PRINT A1
80 PRINT ''Edad: '';A1—A;'' años''
90 PRINT
100 GO TO 20
```

IF/THEN | IF expresión THEN sentencia | SI.../ENTONCES

INTERPRETACION

SI el resultado de la expresión es *cierto* (o *falso*) **ENTONCES** obedece la orden dada por sentencia.

POSIBLIDADES

La expresión siempre estará formada por operadores de relación u operadores de relación y operadores lógicos.

La sentencia puede ser cualquiera de las que figuran en el ESQUEMA DE COMANDOS BASIC.

FORMA DE TECLEAR LA INSTRUCCION

Teclear zz**IF**, la expresión **THEN** y la sentencia, seguido de **ENTER.**

* **Recordar:**

- **Que la expresión** siempre debe contener operadores de relación u operadores lógicos y de relación.
- Que **la sentencia** puede ser cualquiera del BASIC (PRINT, GO, TO, etc.).

EJEMPLO

1. Para ser candidata a Miss Universo se debe medir más de 1,70 metros y pesar menos de 70 kilogramos.
   ¿Podría hacer un programa en el que dado el nombre, la estatura y el peso de cada aspirante se obtuviera la respuesta de si es o no admitida?

SOLUCION

PROGRAMA	COMENTARIOS
10 INPUT ''Nombre ''; N$	El programa se interrumpirá durante su ejecución hasta que se introduzca la cadena de caracters que represente el nombre.
20 INPUT ''¿Talla? ''; T	Se producirá una nueva interrupción hasta que se introduzca el valor numérico correspondiente a la talla.
30 INPUT ''¿Peso? ''; P	Se producirá otra interrupciión hasta que se teclee el valor numérico del peso.
40 IF T<1.7 THEN GOTO 110	Si el valor dado a T es menor que 1.7 ENTONCES, y sólo entonces, el programa salta a la línea 110.
50 IF P >60 THEN GO TO 110	Si el valor dado a la variable P es mayor que 60 ENTONCES, y sólo entonces, el programa sala a la línea 110.
60 PRINT NS, ''admitida''	Si el ordenador llega a leer esta instrucción es, evidentemente, porque en la línea 40 el valor de T era mayor que 1.7 y en la línea 50, con lo cual la candidata supera las condiciones impuestas de talla y peso. Por tanto, se ordena la impresión de su nombre y el mensaje *admitida.*
70 INPUT ''¿Otra candidata?'', O$	Con esta interrupción, el programa queda a la espera de una caena de caracteres, cuya función se ve en la línea siguiente.
80 IF OS = ''NO'' THEN STOP	Si la cadena asignada a la variable OS es igual a NO, entonces el programa se detiene, ya que así lo ordena la sentencia STOP.
90 CLS	Esta sentencia —que se explicará en otro momento— ordena un borrado de pantalla, con el que desaparecen todas las impresiones previas.
100 GO TO 10	Para que el ordenador llegue a leer esta instrucción habrá sido necesario que el valor dado a O$ haya sido distinto a NO y, de esta forma, habremos mandado la lectura del programa a la línea 10, comenzando de nuevo el proceso.
110 PRINT N$, ''rechazada	Esta línea de programa sólo será leída si alguna de las líneas 40 ó 50 lo permiten.
120 GO TO 70	En esta línea se provoca un salto incondicional a la línea 70 para saber si el proceso debe repetirse.

RECUERDE

— **Que exiten símbolos específicos para establecer una relación enntre los términos de una expresión:**
  a) **Los operadores de relación, que sólo operan en proposiciones entre dos operadores:** =, <>, <, >, <=, >=.
  b) *Los operadores lógicos, que actúan entre operadores en los que, a su vez, intervienen operadores de relación. Son operadores lógicos AND, OR y NOT.*
— *Que el comando IF usado con THEN establece una determinada condición dentro del programa.*
— *Que cuando la condición establecida por IF se cumpel, ejecuta la instrucción siguie a THEN.*
— *Que cuando la condición no se cumple, el programa continúa ejecutando la línea siguiente a la del comando IF.*
— *Que la expresión debe estar formada por operadores de relación u operadores de relación y operadores lógicos.*
— *Que el comando IF puede combinarse también con otros comandos BASIC.*

EJERCICIOS

1. Escribir un programa que diga si es o no correcta la suma (resta, multiplicación o división) de dos números cualesquiera.
2. Escribir un programa para contar (o descontar)) de **n** en **n** desde un número **A** (*).
3. Escribir un programa para realizar operaciones combinadas (*).
   Por ejemplo:
   A + B — C
   (A + B)*C
   (A + B — C)/D
4. Escribir un pograma para resolver problemas de este tipo:
   Un individuo tiene **P** pesetas y quiere comprar **U**, unidades de una mercancía, **X** pesetas la unidad, y **V** unidades de otra mercancía, a **Y** pesetas. ¿Cuánto le costará cada mercancía, cuánto importará y cuánto le sobrará?.
5. Escribir programas que comparen entre sí:
   — el valor de varios números.
   — el valor de varias letras.
   — el valor de varias palabras.
6. Escribir el programa que, al introducir las variables **b** (base) y **h** (altura), nos diga si es o no correcto el cálculo que hagamos del área **S** de un paralelogramo (*).
7. Escribir el programa que, al introducir las variables **b** (base) y **h** (altura), nos diga si es o no correcto el cálculo que hagamos del área **S** de un triángulo (*).
8. Escribir el programa que, al introducir las variables **lb** (lado de la base), **ab** (apotema de la base) y **H** (altura del prisma), nos diga sí son o no correctos los cálculos que hagamos del área total **S1**, el área total **St** y el volumen **V** de un prisma (*).
9. Escribir el programa que, al introducir la variable **r** (radio), nos diga si son o no correctos los cálculos que hagamos del área **S** y el volumen **V** de una esfera (*).
10. Escribir el programa que nos diga si es correcta o no la relación que establezcamos entre cada país y su capital.
    Países: España, Portugal, Francia, Italia, Albania, Grecia.
    Capitales: Madrid, Lisboa, París, Roma, Tirana, Atenas.
11. Escribir el programa que nos diga si es correcto o no la realización que establezcamos entre cada órgano del aparato digestivo humano con el acto que realiza (*).
    Organos: boca, dientes, glándulas salivales, faringe, esófago, estómago, intestino delgado, intestino grueso.
    Actos: ingestión, masticación, insalivación, deglución, progresión, quimificación, quilificación, absorcion, defecación.
12. Escribir el programa para calcular la nota final **NF** de un alumno **X**, cuyas evaluaciones a lo largo del curso han sido **N1, N2** y **N3.**
    Clasificaciones posibles:
    I: insuficiente <5
    SF: suficiente = 5
    B: bien = 6
    NT: notable = 7 y 8
    SB: sobresaliente = 9 y 10
13. Idear sus propios programas utilizando conocidos: PRINT, LET, INPUT, GO TO e IF/THEN GO TO.

SOLUCIONES

1.
```
 1 REM Ficha: '' IF THEN '' Ej.1
10 PRINT ''COMPROBRACION SUMA''
20 PRINT ''TOTAL:'';
30 INPUNT T
40                PRINT                T'
50 PRINT ''Sumando'';
60 PRINT N
80 PRINT ''Sumando'';
90 INPUT N1
100 PRINT N1,
110 IF N+N1 <> THE PRINT ''Incorrecta''
120 IF N+N1 = T THEN PRINT ''Correcta''
130 PRINT
140 GO TO 20.
```

4.
```
 1 REM Ficha: ''IF /THEN ''Ej.4''
10 PRINT ''Dinero que tiene: '';
20 INPUT P: PRINT P
30 PRINT ''Unidades '' ''A'' '' '';
40 INPUT U: PRINT U' ''Precio: '';
```

```
 50 INPUT X: PRINT X' ''Unidades '' ''B'' '' '';
 60 INPUT V: PRINT V' ''Precio: '';
 70 INPUT Y: PRINT Y
 80 PRINT ''COSTO MERCANCIA '' ''A'' '': U*X
 90 PRINT ''COSTO MERCANCIA '' ''B'' '': V*Y
100 PRINT ''COSTO TOTAL: ''; U*X+V*Y
110 LET D=U*-X+V*-Y+P: PRINT
120 IF D>O THEN PRINT ''SOBRAN: '';D
130 IF D<O THEN PRINT ''FALTAN: '';D
140 PRINT
150 GO TO 10.
```

5.

```
 1. REM Ficha: ''IF / THEN ''Ej.5''
 10 PRINT ''COMPARA Y ORDENA NUMEROS''
 20 PRINT ''Escriba: '';
 30 INPUT A;'' y '';B;'' y '';C
 40 PRINT A;'' y '';B;'' y '';C
 50 IF A>B THEN LET N=A: LET A=B: LET B=N
 60 IF A>C THEN LET N=A: LET A=C: LET
C=N
 70 IF B>C THEN LET N=B: LET B=C: LET C=N
 80 PRINT A'B'C
 90 PRINT
100 GO TO 20
```

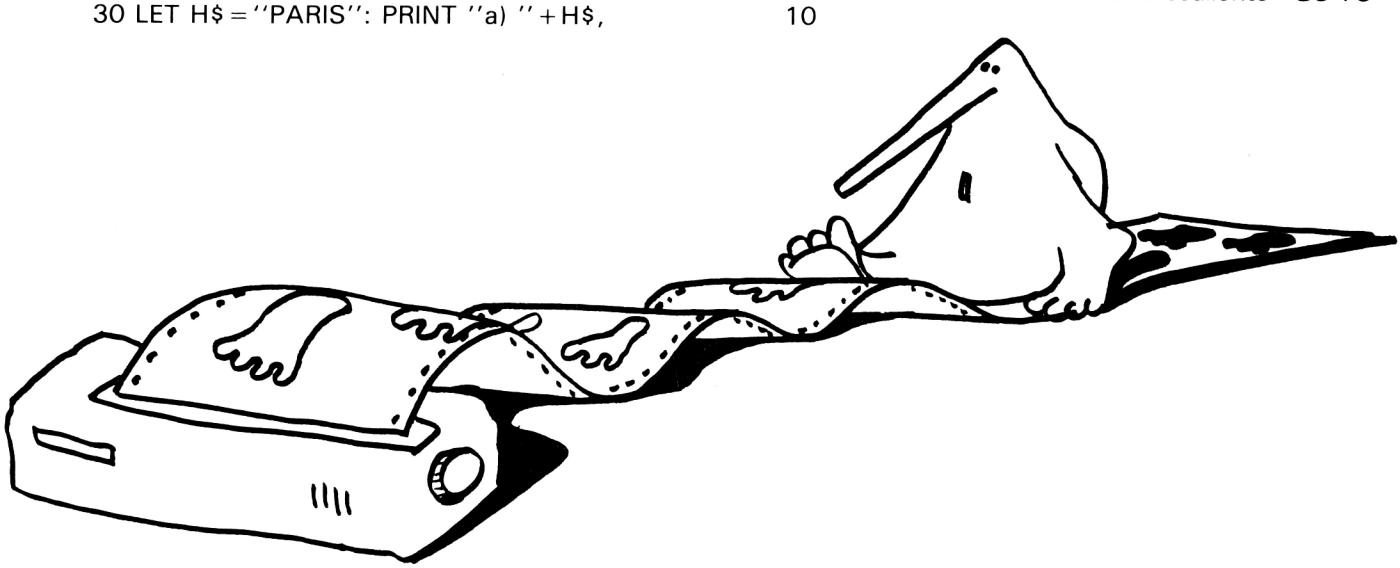
```
 1 REM Ficha: '' IF / THEN '' EJ. 5''
 10 PRINT ''COMPARA LETRAS Y LETRAS''
 20 PRINT ''Escriba: '';
 30 INPUT A$;B$;C$
 40 PRINT A$;'' '';B$;'' '';C$
 50 IF A$>B$ THEN LET N$=A$: LET A$=B$:
LET B$=N$
 60 IF A$>C$ THEN LET N$=A$: LET A$=C$:
LET C$=N$
 70 IF B$>C$ THEN LET N$=B$: LET B$=C$:
LET C$=N$
 80 PRINT A$'';B$;'' ''C$
 90 PRINT
100 GO TO 20
```

10.

```
 1 REM Ficha: '' IF / THEN ''Ej. 10''
 10 PRINT ''RELACION ENTRE NACIONES '' ' ''
Y SUS CAPITALES. '' ''.
 20 LET A$=''ESPAÑA'': PRINT ''1 ''+A$,
 30 LET H$=''PARIS'': PRINT ''a) ''+H$,
 40 LET B$=''PORTUGAL'': PRINT ''2 ''+B$,
 50 LET I$=''ATENAS'': PRINT ''b) ''+I$,
 60 LET C$=''FRANCIA'': PRINT ''3 ''+C$,
 70 LET J$=''TIRANA'': PRINT ''c) ''+J$,
 80 LET D$=''ITALIA'': PRINT ''4) ''+D$,
 90 LET K$=''MADRID'': PRINT ''d) ''K$,
100 LET E$=''ALBANIA'': PRINT ''5 ''+E$,
110 LET L$=''LISBOA'': PRINT ''e) ''+L$,
120 LET F$=''GRECIA'': PRINT ''6 ''+F$,
130 LET M$=''ROMA'': PRINT ''f) ''+M$,
140 PRINT '''Nación:'', ''Capital:'' ''
150 INPUT N$
160 IF N$=''1'' THEN PRINT A$,K$
170 IF N$=''2'' THEN PRINT B$,L$
180 IF N$=''3'' THEN PRINT C$,H$
190 IF N$=''4'' THEN PRINT D$,M$
200 IF N$=''5'' THEN PRINT E$,J$
210 IF N$=''6'' THEN PRINT F$,I$
220 IF N$=''a'' THEN PRINT C$,H$
230 IF N$=''b'' THEN PRINT F$,I$
240 IF N$=''c'' THEN PRINT E$,J$
250 IF N$=''d'' THEN PRINT A$,K$
260 IF N$=''e'' THEN PRINT B$,L$
270 IF N$=''f'' THEN PRINT F$,I$
280 GO TO 150
```

12.

```
 1 REM Ficha: ''IF / THEN '' Ej. 12''
 10 PRINT '' 1a.Evaluación: '';
 20 INPUNT N1
 30                PRINT                N1'
 40 PRINT ''2a. Evaluación: '';
 50 INPUT N2
 60 PRINT N2
 70 PRINT N3
 80 INPUT N3
 90 PRINT N3''
100 LET NF=(N1+N2+N3)/3: IF NF>10 THEN
GO TO 10
110 PRINT ''NOTA FINA: '';NF,
120 IF NF>THEN PRINT ''Insuficiente'' GO TO 10
130 IF NF > THEN PRINT ''Suficiente'' GO TO 10
140 IF NF > THEN PRINT ''Bien'' GO TO 10
150 IF NF > THEN PRINT ''Notable'' GO TO 10
160 IF NF > THEN PRINT ''Sobresaliente'' GO TO
10
```



FICHAS DEL AMSTRAD

GUIA DE PALABRAS BASIC

GUIA DE PALABRAS BASIC

POKE

POKE ¡expresión1!, ¡expresión2!
Escribe el valor de **expresión1** en la dirección de memoria por el valor dado por **expresión2.**

```
10 INPUT ''Valor a introducir'';V
20 INPUT ''Dirección de memoria'';d
30 POKE V, d
40 GOTO 10
```

POS (0)

POS (0)
Retoma la posición actual horizontal del cursor en la pantalla de textos.

```
10 PRINT ''POSICION ACUTAL HORIZONTAL''
20 PRINT POS ( 0)
```

PRINT

PRINT †lista de expresiones†

```
10 A$ = ''UNO''
20 PRINT A$, ''DOS'', ''TRES''
```

PRINT SPC

PRINT †SPC(expresión)†Lista de expresiones†
Coloca el número de espacios dado por **expresión** inmediantamente antes de imprimir la **lista de expresiones.**

```
10 A = 10:A$ = ''PRUEBA''
20 PRINT SPC(A)A$
30 PRINT SPC(15)''OTRA PRUEBA''
```

PRINT TAB

PRINT TAB (expresión) †lista de expresiones†
Desplaza el cursor desde el margen izquierdo de la pantalla tantas posiciones de caracter como indica **expresión** antes de comenzar a imprimir la **lista de expresiones.**

```
10 A = 10:A$ = ''PRUEBA''
20 PRINT TAB (A)A$
30 PRINT TAB (15) ''OTRA PRUEBA''
```

PRINT USING

PRINT†USING''plantilla''; expresión†
Permite la impresión de la **expresión**de acuerdo con el formato dado por la **''plantilla''.**
Caracteres que conforman las plantillas:

	Cada uno de estos caracteres representan una posición de dígito;
.	Este caracter fija la posición del punto decimal:
.,	La coma antes del punto divide el número de gruposs de tres dígitos:
* *	Mediante estos caracteres se ordena la impresión de asteriscos en aquellas zonas de la pantalla que queden ocupadas por espacios:
+	Este símbolo ordena la impresión del signo del número a imprimir antes o después del número, según parezca el + al principio o al final de la plantilla:
—	Este símbolo ordena la impresión del signo — al final de los números negativos:
↑↑↑	Estos símbolos determinan la representación exponencial:
!	Este símbolo es de aplicación en las plantillas para cadenas de caracteres y obliga a la impresión del primer caracter exclusivamente: !
	Estos símbolos más de los espacios incluidos entre ambos determinan los caracteres de la cadena que serán impresos:
&	Este símbolo ordena la impresión de la cadena completa.

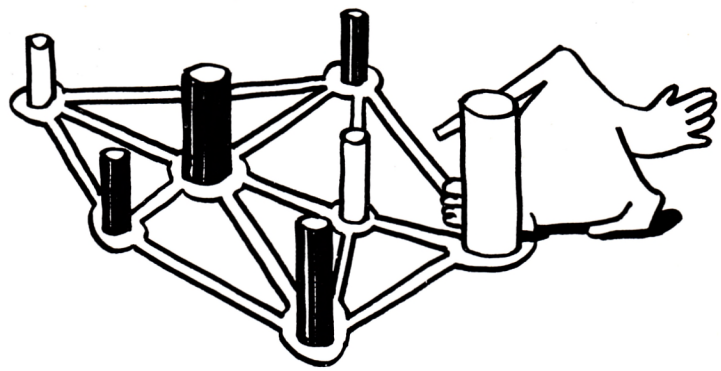
RAD

RAD
Ajusta los cálculos a radianes.

PROGRAMA comentado en

BASIC

DIBUJO EN FUNCIONES EXPLICITAS



Tratando de profundizar en el BASIC del Amstrad, exponemos este programa que sin ser un juego, nos ofrece la posibilidd de pasar ratos entrentenidos y a la vez instructivos.

Se trata de un sencillo programa que nos permitirá representar gráficamente, en unos ejes de coordenadas X-Y, funciones matemáticas (en forma explícita).

El funcionamiento es muy simple, una vez cargado el programa, introducimos en la línea 60 la función a representar en la siguiente forma:

Por ejemplo sea la función y = sen x, la línea 60 deberá contener dicha función, expresada en BASIC, y dentro de DEF FN (Orden Define Funcion).

```
60 DEF FN Y(X) = SIN(X)
```

Una vez introducida la línea, en la forma expuesta, ejecutar el programa con RUN e introducir los valores mínimos y máximos que ha de tomar la X, teniendo en cuenta que el programa sólo admite valores comprendidos en el intervalo (−16, +16).

A continuación se representan los ejes de coordenadas y seguidamente la función por puntos. Una vez concluída dicha representación y presionado una tecla, puede volverse a estudiar la misma función variando los valores mínimos y máximos de X.

Si se desea cambiar de función, interrumpir el programa (ESC 2 veces) e introducir la nueva en la línea 60 tal y como se indicó anteriormente.

Comentarios al programa

10-20	Título del programa y nombre utilizado para grabar.
30	Pantalla de 80 caracteres. Creación de una ventana para mensajes, desde la columna 20 a la 55, en la fila 24 (Penúltima de la pantalla). Los colores de tinta y papel se invierten.
40	Mensaje en pantalla.
50	Mensaje en ventana.
60	Definición de la función mediante DEF FN.
70	Pausa indefinida, hasta que no se presione una tecla.
80	Origen de coordenadas a 0,0 (abajo y a la izquierda).
90-100	Entrada de datos por ventana. Borrado de pantalla.
120	Control de los valores introducidos, en el intervalo previsto.
130	Bucle de dibujo del eje X.
140	Dibujo del eje punto a punto.
150	Escala de dibujo (1 Ud = 20 Pixels).
160	Dibujo de los trazos a intervalos de 20 pixels, cada trado indica una unidad.
170	Fin del bucle de dibujo del eje X.
180	Sitúa el sentido del eje X (−/+).
190-240	Bucle para el eje Y, igual al utilizado para el X.
250	DIBUJO DE LA FUNCION.
260	Nuevo origen para situarlo en el cruce de los ejes dibujados anteriormente.
270	Bucle de dibujo.

280 Calcula el valor de Y correspondiente a un determinado valor X, dado por el bucle.
290 Adecua los valores de X e Y a la escala elegida.
300 Calcula si los valores a representar se encuentran dentro de la pantalla.
310 Dibuja la función punto a punto.
320-360 Pausa controlada y vuleta al programa.

Mejoras al programa

Deliberadamente hemos impuesto una restricción, al intervalo de representación de la función, concretamente entre $-16 <= X <= 16$), pero obviamente puede representarse para valores mayores y menores. Utilice el factor de escala y modifique el programa en este sentido.

A continuación se incluye una relación de funciones a representar y los intervalos correspondientes, pero no se limite a ellos, busque nuevas funciones en un libro de matemáticas, o créelas usted mismo haciendo variaciones sobre las propuestas, a fin de estudiar los resultados.

Y = X↑2 + 5 ..... −5 / 5
Y = X↑2 ..... −3 / 3
Y = LONG (X) ..... 0.1 /10
Y = SIN (X) ..... −10 / 10
Y = TAN (X) ..... −5 / 5
Y = SQR (X ↑2+3) * SIN (X) ..... −6 / 6
Y = EXP (X) ..... −1 / 13
Y = X − 5 ..... −5 / 15
Y = COS(EXP(−X/5)) ..... −8 / 8
Y = 5 + 2 * X↑ − x↑4 ..... −3 / 3

Existen un sinfín de mejoras, así que anímese y hágalas, éste es uno de los objetivos que nos proponemos con estos programas, que profundice más en el BASIC, mejorando siempre nuestros programas.

```
10 ' ************    DIBUJO DE FUNCIONES EXPLICITAS   ***********
20 ' ********************      "DIBUFUN"     **********************
30 MODE 2: WINDOW #1,20,55,24,24:PAPER #1,1:PEN #1,0:CLS#1
40 PRINT "SI DESEA CAMBIAR LA FUNCION PROGRAMADA, PULSE <ESC> 2 VECES Y EDITE LA
 LINEA 60  INTRODUZCA  LAS MODIFICACIONES OPORTUNAS Y HAGA RUN NUEVAMENTE"
50 PRINT #1," PRESIONE UNA TECLA PARA CONTINUAR"
60 DEF FN Y(X)=TAN(X)
70 CALL &BB18
80 ORIGIN 0,0
90 INPUT #1,"   VALOR MIN DE X";XMIN
100 INPUT #1,"   VALOR MAX DE X";XMAX
110 CLS
120 IF XMIN<-16 OR XMAX>16 THEN 80
130 FOR X=1 TO 639
140 PLOT X,200
150 ESCALA=X/20
160 IF ESCALA=INT(ESCALA) THEN DRAWR 0,5:DRAWR 0,-10:DRAWR 0,5
170 NEXT X
180 LOCATE 1,14: PRINT "-X": LOCATE 80,14: PRINT "X"
190 FOR Y=1 TO 399
200 PLOT 320,Y
210 ESCALA=Y/20
220 IF ESCALA=INT(ESCALA) THEN DRAWR 5,0:DRAWR -10,0:DRAWR 5,0
230 NEXT Y
240 LOCATE 39,1: PRINT "Y":LOCATE 38,25: PRINT "-Y"
250 'DIBUJO
260 ORIGIN 320,200
270 FOR X=XMIN TO XMAX STEP 0.01
280 Y= FN Y(X)
290 CX=X*20:CY=Y*20
300 IF CY>199 OR CY<-200 THEN 320
310 PLOT CX,CY
320 NEXT X
330 LOCATE 1,25: PRINT "PRESIONE PARA CONTINUAR"
340 CALL &BB06
350 CLS:CLS#1
360  GOTO 80
```

LOGO del Amstrad

Capítulo VII

VARIABLES

Los lectores que hayan seguido los ejemplos y los programas comentados hasta el número presente han tenido la oportunidad de ver cómo el Dr. Logo procesa datos numéricos. Hemos dado muchas órdenes a la tortuga para que haga diversas cosas. Por ejemplo para ordenar que la tortuga avance 50 pasos se defe ejecutar la orden,

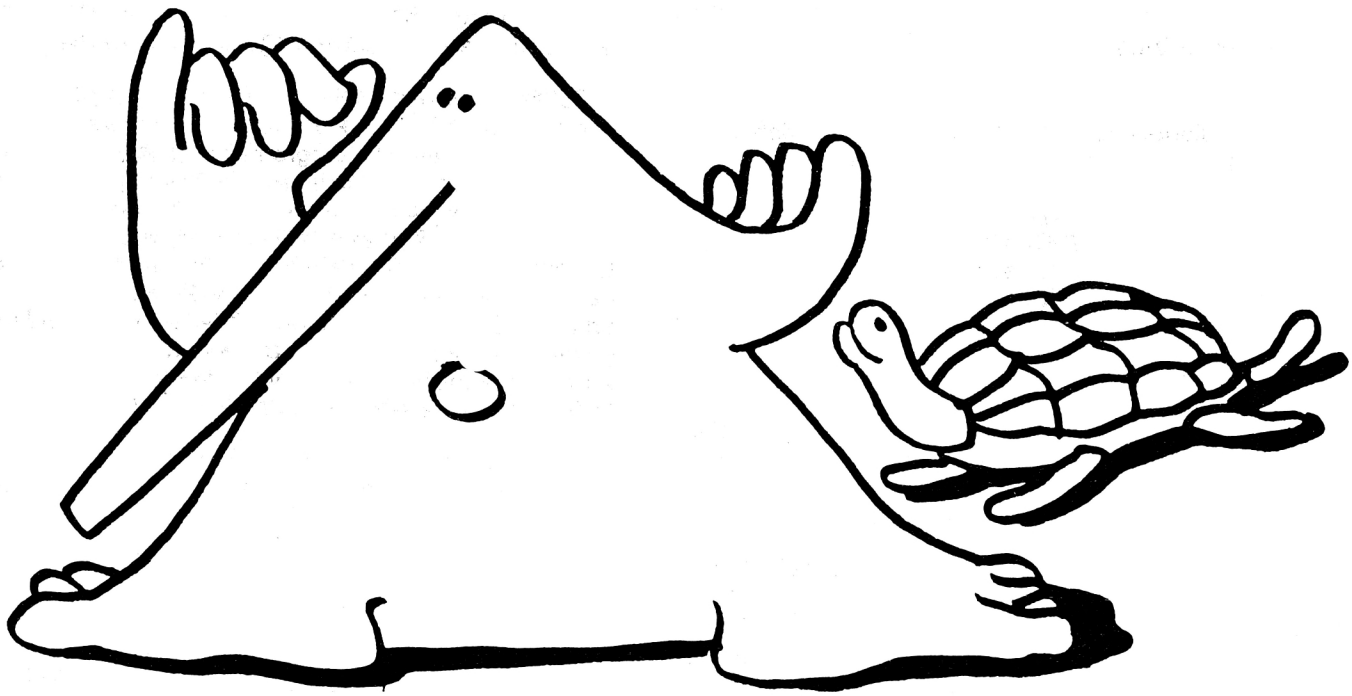
fd 50

perfectamente conocida ya por todos nosotros.

Ante una orden como la anterior se suscitan algunas preguntas en relación con el número 50 de pasos que la tortuga debe recorrer. Por ejemplo es lícita la siguiente reflexión: ¿qué sucede si yo lo que necesito es que la tortuga avance (o retroceda o gire, etcétera), un número de pasos que desconozco cuando escribo el programa, porque depende del propio desarrollo de éste? Efectivamente, la orden **fd 50** hace avanzar invariablemente a la tortuga 50.

Otra pregunta que puede suscitarse ante la orden anterior se refiere a la propia naturaleza del número (50) que le pasa como **dato**. Además de las palabras que forman las órdenes de Dr. Logo, ¿éste sólo puede manejar números? Es decir ¿existen otros tipos de datos?

Apresurémonos a contestar siquiera sea superficialmente a la segunda pregunta, sobre la cual volveremos en otro capítulo con mucho más detalle. Naturalmente no siempre será nuestra intención procesar números cuando escribamos programas en Logo. Y podremos escribir programas que procesen información no numérica como palabras, frases, etcétera. Pero aquí sólo estamos interesados en resaltar el hecho de que existen distintos tipos de datos que pueden manejarse con un ordenador, por ejemplo, trabajando en Logo. Hay otros lenguajes que ponen mucho énfasis precisamente en este hecho a la hora de escribir los programas. Por ahora notemos sólo que el concepto de **tipo de dato** existe y es importante. Un tipo de dato determina el conjunto de valores que el dato puede tomar. Por ejemplo, si lo que pretendemos es ordenar a la tortuga un desplazamiento, es obvio que el tipo de dato que le suministremos como argumento a la correspondiente orden habrá de ser numérico.



VII.1. CONCEPTO DE VARIABLE

Buena parte del resto del capítulo está destinado a contestar a la primea de las preguntas planteadas. Buena parte sí pero no todo el capítulo porque veremos además la riqueza que aporta Dr. Logo en el tratamiento de este tema muy superior en algunos aspectos al suministrado por otros lenguajes de programación.

Hasta ahora siempre que hemos dado una orden a la tortuga por ejemplo, lo hemos hecho mediante un número. Ese número no cambia a lo largo del desarrollo del programa por más veces que la ejecución de la instrucción se repita. Así, ''fd 50'' hace avanzar 50 pasos cada vez a la tortuga. Cuando un dato de un programa no cambia (no **puede** cambiar), a lo largo de la ejecución del mismo, se dice que este **dato** es **constante.**

Si existen datos que no pueden cambiar y hablamos de ellos debe ser porque existirán datos que pueden cambiar a lo largo de la ejecución del programa. En efecto, cuando un dato va a cambiar (**puede** cambiar) durante la ejecución de un programa se dice que estamos ante un **dato variable.**

Debe pues haber algún modo de representar los datos variable. En efecto, la solución sería no escribir directamente el propio dato en el programa (en cuyo caso **no se puede cambiar y sería un dato constante)** sino algún **símbolo** que lo represente a todos los efectos. A este tipo de símbolos se les llama **variables**. Así pues variable es un símbolo que representa a algún dato que puede variar a lo largo de la ejecución del programa. Como se formen esos símbolos en un determinado lenguaje es otro problema. Nosotros sólo estamos interesados en cómo se forman las variables en Logo.

Una **variable** en Logo consiste sencillamente en una palabra. Cualquier palabra puede ser el símbolo usado por **Dr. Logo** para representar a un dato variable. Por ejemplo, la palabra PASOS podría ser el nombre usado para ''almacenar'' el número de pasos que debe avanzar la tortuga.

Hemos usado la palabra **almacenar** a propósito en este punto. Normalmente el dato representado por una variable se dice que está almacenado en ella recordando que, en realidad, una variable representa el nombre de alguna posición de memoria en la que se ''almacena'' el dato.

Esencialmente el concepto de variable está descrito. Pero hay mucho más que decir sobre ellas en una primera aproximación.

Por ejemplo, y esto es sólo una sugerencia, aunque Dr. Logo acepte casi cualquier palabra como nombre de una variable, nuestro consejo como el de la inmensa mayoría de los autores, es que se utilicen nombres lo más representativos posible. Por ejemplo, la palabra

$

podría ser el nombre de una variable. Sin embargo, resulta bien poco representativa el dato que almacena (salvo, tal vez, que el dato fuera un número de dólares, cosa no probable). Otro tanto puede decirse de la palabra **2** o de la palabra **zyftr**. Es mucho mejor acostumbrarse desde el principio a usar palabras con algún significado. Por otra parte Dr. Logo no impone ninguna limitación pero otros lenguajes, sí. Así, tenemos en otros lenguajes la imposibilidad de comenzar el nombre de una variable con algún carácter no alfabético y además el número máximo de caracteres que pueden formar el nombre suele estar también limitado.

EJEMPLO VII.1.

Los siguientes son nombres posibles de variables en Dr. Logo:

pasos
PASOS
distancia1
dist.1
$.h
34

A propósito: ¿Las variables llamadas ''pasos'' y ''PASOS'' son la misma o son distintas para Dr. Logo? Son distintas. A todos los efectos Dr. Logo distingue entre las letras mayúsculas y las minúsculas.

Bueno, ya sabemos qué cosa es una variable, y qué clase de símbolos podemos usar para utilizarlas con eficacia. Pero nos esperan todavía algunos problemas por resolver antes de poder utilizarlas. Nuestras siguientes líneas van a ir encaminadas a resolver la siguiente pregunta: ¿Cómo se informa a Dr. Logo de que una cierta variable representa un determinado dato desde un punto determinado del programa y ''hasta nueva orden''? ¿Cómo se hace para utilizar el contenido de una variable (es decir, cómo se hace para que Dr. logo sepa distinguir entre el nombre de la variable y el dato por ella representado)?

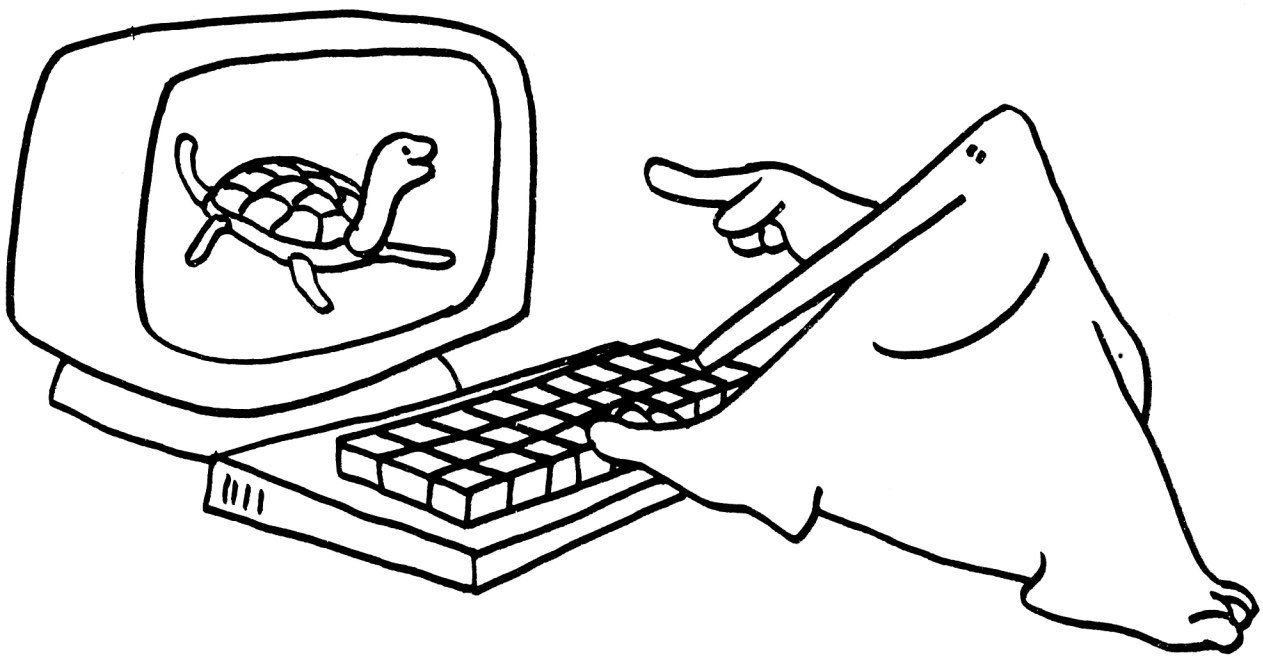
Dado que es completamente seguro que muchos de nuestros lectores tienen conocimientos del lenguaje BASIC vamos a hacer algunas comparaciones con él para aclarar algo más los conceptos que, sin embargo, serán expuestos de modo que los lectores que no estén en ése caso podrán seguirlos con igual precisión.

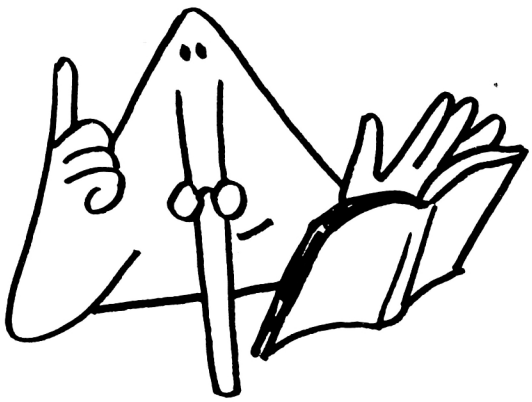
Sabemos ya por capítulos anteriores que para invocar la ejecución de un procedimiento, basta escribir en el punto adecuado del programa su nombre. Como los nombres de los procedimientos cumplen iguales requistos que los de las variables, algo habrá que hacer para informar a Dr. Logo de qué cosa queremos hacer exactamente. Recordemos en este punto que el concepto de **palabra** es fundamental en Logo; todo son palabras en Logo, unas representan órdenes primitivas del lenguaje, otras son ellas mismas los datos del programa, otras son nombres de procedimientos y ahora tenemos que las variables se referencian también con palabras.

Si el número de pasos que queremos que avance la tortuga está contenido en la variable llamada, por ejemplo, DATOS y nosotros escribimos la orden a que nos referíamos al principio, pero transformada en la forma:

fd PASOS

entonces al sabio Dr. Logo se le prestará una horrorosa ambigüedad ¿Será PASOS un dato numérico? (no parece a primera vista pero ahora veremos que la pregunta tiene sentido); ¿será PASOS un nombre de variable?; ¿será PASOS un procedimiento que proporciona el resultado que se debe suministrar a la orden ''fd''? Dr. Logo no tienen datos para resolver el problema. La tercera de las preguntas puede no haberla entendido con exactitud el lector. Más adelante, en este mismo capítulo encontrará la solución a sus dudas. La segunda pregunta tiene un significado obvio. Respecto de la primera hemos de aclarar lo siguiente: es claro que PASOS no es un dato numérico. Pero según se dijo an-





tes cualquier palabra es un nombre legal de variable. Así que, por ejemplo, 50 podría ser formalmente un nombre de variable. Entonces la orden ''fd 50'' permite la duda de Dr. Logo entre las 3 posibilidades indicadas. Pasemos pues, a resolver el problema.

Hay dos cosas que se pueden hacer con una variable. **Asignarle** un valor y **utilizar** el valor almacenado en expresiones, fórmulas, etcétera. Todos los lenguajes de programación suministran instrucciones de asignación. Una **instrucción de asignación** permite asignar a una variable un valor, es decir, informar al ordenador de cuál es el dato que representa la variable. En Dr. Logo esta instrucción es:

make ''nombre dato

Como vemos, el nombre de la variable va **inmediatamente** precedido por el carácter comillas: ''. Esta es la forma en que se informa a Logo de que nos estamos refiriendo a la variable así llamada y no a alguna otra cosa con ese mismo nombre.

Antes de pasar a ver cómo hacemos referencia al contenido de una variable, vamos a ver algunos ejemplos de cómo asignar palabras, números y listas:

EJEMPLO VII.2.

—Para asignar a la variable llamada ''pasos'' el valor 50 ejecutaríamos:

make ''pasos 50

—En la orden anterior, el dato almacenado en el variable ha sido numérico. Cuando queramos almacenar una palabra, entonces la propia palabra a almacenar también ha de ir precedida de los caracteres '' para indicar a Logo que no nos referimos al resultado que podría producir la ejecución de un procedimiento así llamado. Por ejemplo, para almacenar en la variable llamada ''acción'' la palabra ''avanzar'' se ejecuta la orden:

make ''acción ''avanzar

—Como tantas otras veces, surge una pregunta: ¿Qué pasará si al número 50 se le antepone el caracter ''? (Será tomado como un número o como una palabra y, por tanto un dato alfanumérico? La respuesta está aquí: Es indiferente el situar o no las comillas delante de los datos numéricos; serán siempre bien interpretados. Por ejemplo, la orden.

make ''pasos 50

es equivalente a la orden.

make ''pasos 50

—Como ha quedado ya claro, las cosas son de otro modo con las demás palabras. Así la ejecución de la orden,

make ''acción avanzar

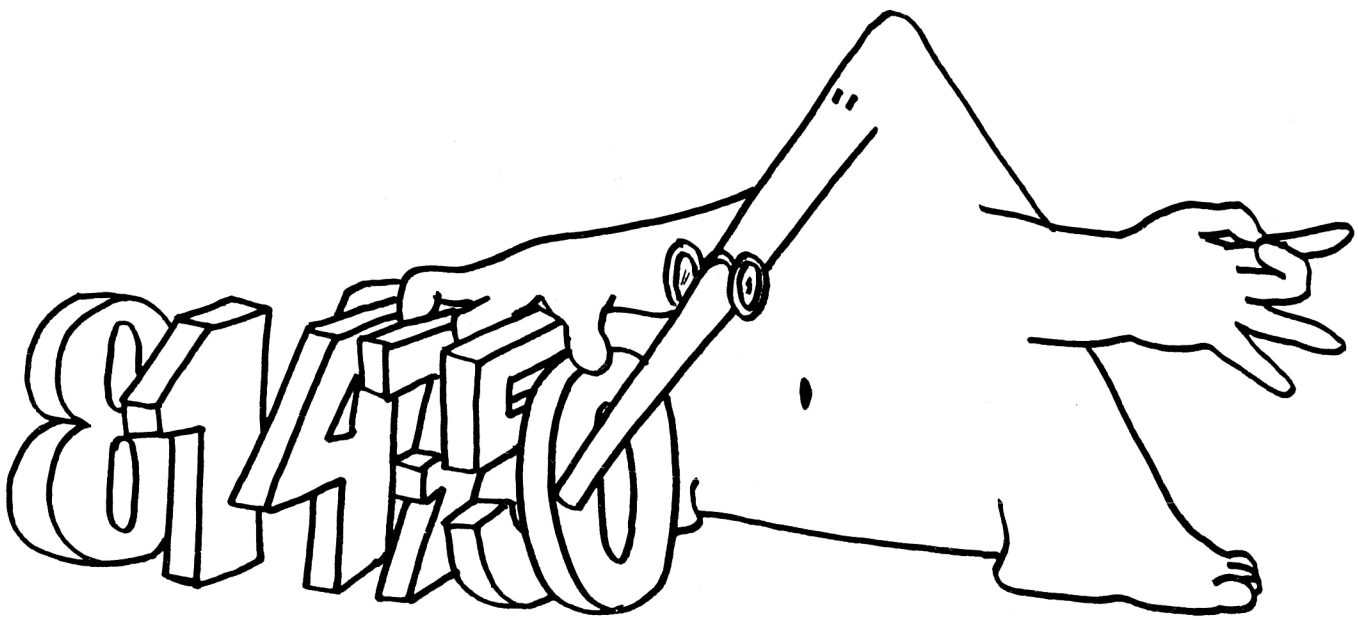
haría interpretar a Dr. Logo que ''avanzar'' es el nombre de un procedimiento que produce algún tipo de resultado, y que ese resultado es el que debe almacenar en la variable ''acción''. Volveremos sobre este punto.

—La forma de asignar una lista a una variable es muy simple. Basta encerrar la lista completa entre corchetes. Cada uno de los elementos de la lista debe describirse en su interior. Por ejemplo:

make ''lista [pasos [50 30 45] giros [90 80 30]]

almacena en la variable ''lista'' una lista que tiene cuatro elementos: la palabra ''pasos'', la lista cuyos elementos son 50 30 y 45 en este orden, la palabra ''giros'' y la lista cuyos elementos son 90, 80 y 30. Estas listas se llaman entonces sublistas de la anterior.

Pasemos pues ya a ver como hacer para referirnos al contenido de una variable. Supongamos para ello, por ejemplo, que hemos asignado el valor 50 a la variable ''pasos'' ejecutando la orden: make ''pasos 50.



Pretendemos ordenar a la tortuga que avance tantos pasos como el número almacenado en la variable ''pasos''. Ya se ha dado cuenta el lector de que la orden no puede ser.

fd pasos

porque de nuevo Dr. Log ''pensaría'' que ha de hacer avanzar a la tortuga tantos pasos como dé el resultado de ejecutar el procedimiento pasos (que, en realidad, no tiene porque existir). De otra parte, la orden tampoco puede ser

fd ''pasos

porque entonces Dr. Logo emitirá el mensaje de error:

fd doesn't like pasos as input

Claro: el único tipo de dato que se puede incluir en la sentencia fd es numérico, y ''pasos es interpretada por Logo como la propia palabra.

Para referirnos al contenido de una variable, se antepone a la palabra que la identica el caracter ''dos puntos''; '':''. Esto hace que Dr. Logo pueda diferenciar entre palabras que son nombres de procedimientos, las propias palabras consideradas como tales, y los contenidos de las palabras que son nombres de variables.

EJEMPLO VII.3.

—Si ejecutamos la secuencia de órdenes;

make ''pasos 70
fd :pasos

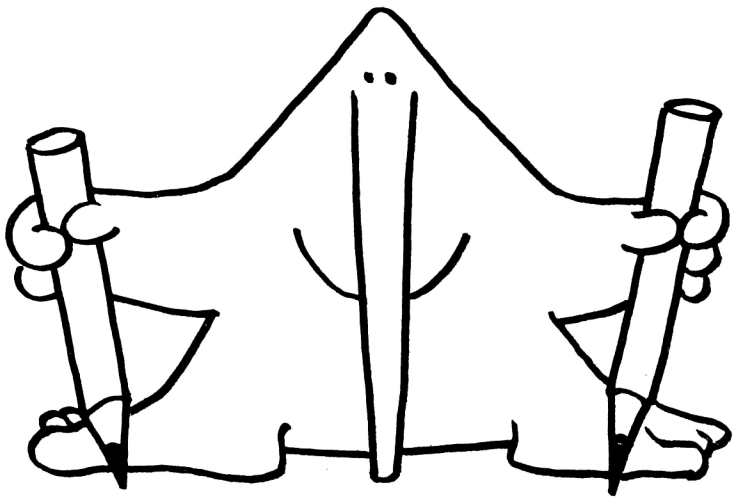
entonces se asigna el valor 70 a la variable llamada ''pasos'' y después la orden de avance hace que la tortuga se desplace hacia adelante el número de pasos que indique el contenido de esa variable. Del mismo modo la orden,

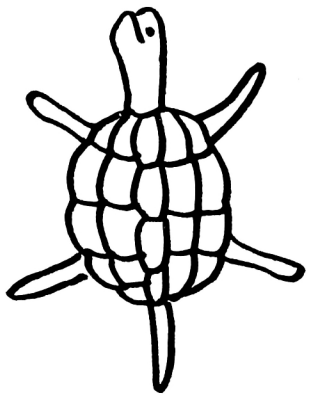
pr :pasos

escribe en pantalla el número 70 (contenido de ''pasos'').

Hemos de aclarar todavía algunos puntos. Por ejemplo. nos podemos preguntar si es posible que Dr. Logo diferencie entre un procedimiento y una variable que tengan el mismo nombre. La respuesta es sí. Así, a la vez que la variable llamada ''pasos'' definida cuando se ejecutó la instrucción;

make ''pasos 50





puede haber un procedimiento llamado pasos, que puede incluso usar a esa variable. Por ejemplo:

```
to pasos
make "pasos 50
fd :pasos
pr :pasos
end
```

hace que todas las acciones se ejecuten normalmente sin interrelación alguna entre la variable y el procedimiento del mismo nombre. Es muy probalble que muchas personas que conozcan algún otro lenguaje esté pensando en la cantidad inmensa de posibilidades que tiene el Logo par amanejar símbolos. Es muy cierto. Apenas hemos empezado.

En efecto, poco a poco se irán presentando ante nuestros ojos cosas cada vez más potentes que puede hacer Dr. Logo. Lo siguiente pretende por un aparte hacer alguna aportación en este sentido; y además hacer alguna última aclaración sobre el concepto de variable tal y como la maneja Dr. Logo. La pregunta que vamos a resolver a continuación es la siguiente: Supongamos que almacenamos en la variable llamada VAR1 la palabra VAR2. Esto se hace con la orden:

```
make "VAR1 "VAR2
```

y supongamos que estamos interesados en almacenar **en el contenido** del contenido de VAR1 (es decir VAR2). ¿Bastará ejecutar la orden,

```
make :VAR1 "VAR3?
```

La respuesta es SI!! En efecto, esta última orden indica a Dr. Logo que ha de almacenar la palabra VAR3 en el contenido de la variable llamada VAR1. Este contenido es VAR2 de modo que el contenido de VAR2 es ahora VAR3.

EJEMPLO VII.4.

Supongamos que ejecutamos,

```
make "VAR1 "VAR2
make "VAR1 "VAR3
```

entonces las siguientes órdenes producen los resultados indicados:

```
pr :VAR1
VAR2
pr :VAR2
VAR3
```

Hemos, sin embargo, de hacer una aclaración. Si algún lector se está preguntando (a buen seguro que más de uno), si Dr. Logo admite referencias a más larga distancia en el sentido anterior, hemos de indicarle que, lamentablemente no. Concretamente nos referimos al siguiente razonamiento: Si el contenido de VAR1 es VAR2 y el contenido de VAR2 es VAR3 entonces el contenido del contenido de VAR1 es VAR3 (hasta aquí todo es correcto) en consecuencia (desde aquí no lo cumple Dr. Logo), podemos referenciar el contenido del contenido de VAR1 desde el propio VAR1. Dr. Logo no hace esto. Concretamente la orden:

```
pr ::VAR1
```

imprime un mensaje de error en lugar de VAR3 que es lo que esperábamos. Dr. Logo ofrece muchas ventajas sobre otras versiones del Logo que hay en el mercado, pero ofrece algún inconveniente respecto de alguna de ellas. Algunas versiones (desde luego el LOGO estandar propuesto por Seymour Papert) incorporan la operación "thing", que permite referencial el contenido del contenido de una variable desde la propia variable.

PROGRAMAS

Presentamos en este número tres programas completamente distintos. Probablemente los lectores que estén siguiendo el texto y los distintos programas que se van presentando hayan observado que muchos de los programas escritos en Logo son extremadamente cortos. Esto es cierto e incluso lo que el hecho de que algunos de los más cortos programas tienen una lógica interna a la vez simple y complicada.

Aunque la teoría correspondiente se desarrollara en un capítulo posterio vamos a presentar en este número, uno de los programas más versátiles y que primero suelen estudiarse en Logo. Nos referimos a uno de los ejemplares de la ''familia de procedimientos'' POLI. Recomendamos al lector que ejecute este programa en el ordenador y disfrute con las figuras obtenidas con la tortuga.

Como un segundo ejemplo de programas sencillos para realizar experimentos y figuras geométricas de gran belleza proponemos un segundo programa más simple, si se quiere, pero también ilustrativo de las técnicas que pueden usarse. Nos referimos al programa LINS.

Por último, se muestran un par de programas de los también considerados clásicos en el estudio de la tortuga y de los lenguajes recursivos. Nos referimos al programa ARBOL que se adentra en los fenómenos de la recursión general. Desde luego cae lejos de la teoría presentada hasta ahora, pero se estudiara con profundidad en su momento. Invitamos al lector a que examine desde ahora cuidadosamente el ''tortuoso'' camino que debe recorrer la tortuga para completar el dibujo.

Se ha usado la orden CS.5 que fue presentada como utilidad en el número 6 de la revista. Los lectores que no tengan a mano este número, o simplemente que así lo prefieran, pueden sustituir esta orden por **cs**, dado que la utilidad anterior sólo modifica los colores del fondo y de la tortuga y establece el número de líneas de la pantalla mixta.

1) POLI

Aunque existen diversas versiones para este procedimiento, quizás la más sencilla sea la siguiente:

```
to poli :lado : ángulo
fd :lado rt : ángulo
poli :lado :ángulo
end
```

Es sorprendente el número de líneas de que consta el programa si lo comparamos con la enorme versatilidad que tiene. Proponemos al lector que disfrute ejecutándolo, al menos, de los siguientes modos:

```
CAS.5 2
poli 40 30
poli 100 144
poli 100 45
poli 100 125
poli 100 150
```

Atención: La ejecución de estos procedimientos no termina a no ser que pulse la tecla **ESC**. Pulsese pues al cabo de algunos segundos cuando ya la figura sea regular. De no proceder así y dejar a Dr. Logo recorrer una vez tras otra la figura otendremos el mensaje de error siguiente:

I'm out of space in poli :lado : angulo

y podremos observar además que el prompt del Logo habrá cambiado del caracter ''?'' al caracter ''!''. Para que las cosas sigan funcionando normalmente debemos ejecutar ahora la orden:

recycle

En su momento veremos al razón de esto. Si en vez de ello pretendemos ejecutar un procedimiento cuando el prompt sigue siendo ! entonces lo más probable es que obtengamos el mensaje de error:

I don't have any LOGO nodes left

2) LINS

Este es un sencillo programa que sólo pretende inspirar al lector para la construcción de bonitas figuras con la tortuga.

```
to cuadrado :1
repeat 4[fd :1 90]
end

to raro
fd 50 cuadrado 20
lt 90 fd 20 rt 90 fd 50
lt 90 cuadrado 50
rt 180 fd 50 cuadrado 20
rt 90 fd 70
end

to lins
repeat 12[raro rt 30]
end
```

3 ARBOL

No se trata, como tantas otras veces en esta sección, de que se entienda el programa. Se trata de ir viendo las muchas posibilidades de la tortuga del Logo. El siguiente es un programa complejo. Ejecutese en el ordenador e inténtese solamente seguir los pasos de la tortuga.

Para ejecutar el procedimiento debe escribirse el nombre de este (árbol) seguido por dos números. El primero de ellos indica la longitud de la rama mayor (por ejemplo 100) y el segundo la longitud de la rama menor (por ejemplo 2).

```
to árbol :1 :t
if :1<:t [stop]
fd :1
lt 45
árbol :1/2 :t
(* llamada recursiva al procedimiento *)
rt 90
árbol :1/2 :t
(* nueva llamada recursiva *)
1t 45
bk :1
end
```

Un ejemplo de llamada a ejecución sería : ''árbol 100 5''

La siguiente versión introduce la posiblidad de decidir cuanto ha de reducirse la longitud de las ramas cada vez; en la versión anterior esta reducción es a la mitad, como vemos. Es indispensable para que todo esto tenga sentido, que el tercer parámetro (el factor de reducción) sea menor que 1. Para ejecutar esta nueva versión se ha de añadir el tercer número; si este número es 5 entonces el árbol obtenido será de las proporciones de la versión anterior. Recomendamos que se sitúe a la tortuga en la parte baja de la pantalla antes de ejecutarlo.

```
to arbolb : 1 :t :r
if :r>1 [pr [rror] stop]
if :1<[stop]
fd :1
lt 45
arbolb: 1* :r :t :r
rt 90
arbolb: 1*r :t :r
lt 45
bk :1
end
```

Para ejecutarlo podríamos situar a la tortuga en la parte baja de la pantalla por ejemplo con las siguientes órdenes:

```
CS. 5 3
pu
bk 40
pd
```

después de lo cual podríamos ejecutar por ejemplo:

arbolb 100 5 .7

El programa técnico del mes

CALCULO DEL DETERMINANTE DE UNA MATRIZ CUADRADA

por Víctor J. Campo López

Iniciamos este mes una serie de tres programas de cálculo matricial, que esperamos sea de gran utilidad tanto para el estudiante como para el profesional, relacionado con este tema.
La serie abarca el cálculo de la matriz inversa, valor del determiante de una matriz cuadrada y resolución de sistemas de ecuaciones lineales.

El primer programa que presentamos es el de ''Cálculo del valor del determinante de una matriz cuadrada''. El procedemiento utilizdo es el triangulación de la matriz, de forma que el valor de dicho determinante, resulta ser el producto de los elementos de la diagonal principal de la matriz triangulada.

Para dar una mayor brevedad al programa, no se ha incluido una rutina de depuración de datos de entrada, de manera, que cualquier error en los mismos, supone el tener que interrumpir el programa, para volver a ejecutar RUN. No obstante, para determinante de orden elevado sería conveniente la inclusión de una rutina de depuración al efecto, por lo que la exponemos a continuación, de foma que el lector la incorpore o no al programa. Los números de línea de esa rutina han sido seleccionados, de forma que sean compatibles con los del programa.

Rutina de depuración de datos de entrada

```
175 GOSUB 1000

1000 ' Rutina de depuracion de datos de entrada  .....................
1005 '
1010 MODE 2 :WINDOW #1,1,80,24,25:PAPER #1,1:PEN #1,0:CLS#1
1020 PRINT "    ";CHR$(24);
1030 FOR C=1 TO N:PRINT C,:NEXT C:PRINT CHR$(24)
1040 FOR F=1 TO N
1050 PRINT CHR$(24);F;CHR$(24);
1060 FOR C=1 TO N
1070 PRINT A(F,C),
1080 NEXT C
1090 PRINT
1100 NEXT F
1110 INPUT #1,"CUANTOS ELEMENTOS ERRONEOS ";MAL
1120 IF MAL=0 THEN 1190
1130 FOR K=1 TO MAL
1140 PRINT  #1,"INTRODUZCA  <Fila,Columna> DEL DATO ERRONEO No.";K;
1150 INPUT #1,"    ";F,C
1160 INPUT #1,"INTRODUZCA NUEVO VALOR ";A(F,C)
1170 NEXT K
1180 GOTO 1010
1190 RETURN
```

```
10 ' ************************************************************************
20 ' *******   CALCULO DEL DETERMINANTE DE UNA MATRIZ CUADRADA   ******
30 ' *******                    "DETERMIN"                        ******
40 ' ************************** VJCL ********************************
50 '
60 MODE 2:INPUT "ORDEN DEL DETERMINANTE ";N
70 DIM A(N,N),Z(N)
80 ' Entrada de datos de la matriz   ..................................
90 '
100 FOR F=1 TO N
110 PRINT "F I L A ";F
120 FOR C=1 TO N
130 PRINT "COLUMNA ";C;
140 INPUT A(F,C)
150 NEXT C
160 PRINT "----------------------------------------"
170 NEXT F
180 ' Validez del determinante (matriz no singular) ..................
190 '
200 FOR C=1 TO N
210 FOR F=C TO N
220 IF A(F,C)<>0 THEN 250
230 NEXT F
240 PRINT CHR$(24);"LA MATRIZ  ES SINGULAR Y POR TANTO EL DETERMINANTE NULO";CHR
$(24):END
250 ' Calculo de la matriz inversa (B)  ..............................
260 '
270 FOR k=1 TO n
280 AUX=A(C,K)
290 A(C,K)=A(F,K)
300 A(F,K)=AUX
310 NEXT K
320 Z(C)=A(C,C)
330 Z=1/A(C,C)
340 FOR K=1 TO N
350 A(C,K)=A(C,K)*Z
360 NEXT K
370 FOR I=1 TO N
380 IF I=C THEN 430
390 Z=-A(I,C)
400 FOR J=1 TO N
410 A(I,J)=A(I,J)+A(C,J)*Z
420 NEXT J
430 NEXT I
440 NEXT C
450 ' Calculo de los productos de la diagonal  .......................
460 '
470 P=1
480 FOR K=1 TO N:P=P*Z(K):NEXT K
490 ' Salida de resultados  ..........................................
500 '
510 PRINT :PRINT CHR$(24);TAB(10)"El valor del determinante es   ";P;CHR$(24)
```

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 Vídeo
- Decoración

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

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. REF. T-XT 08013 Barcelona

Tel.: (93) 245 33 06

Actúe ahora en su propio beneficio y pídanos información.

GRATUITAMENTE

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

_____

Nombre y apellidos _____ Edad _____

Domicilio _____

_____ N.º _____ Piso _____ Pta. _____ Tel. _____

C. Postal _____ Población _____

_____ Provincia _____

Profesión _____

CEAC. Aragón, 472

(Dpto. REF. T-XT) 08013 Barcelona

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