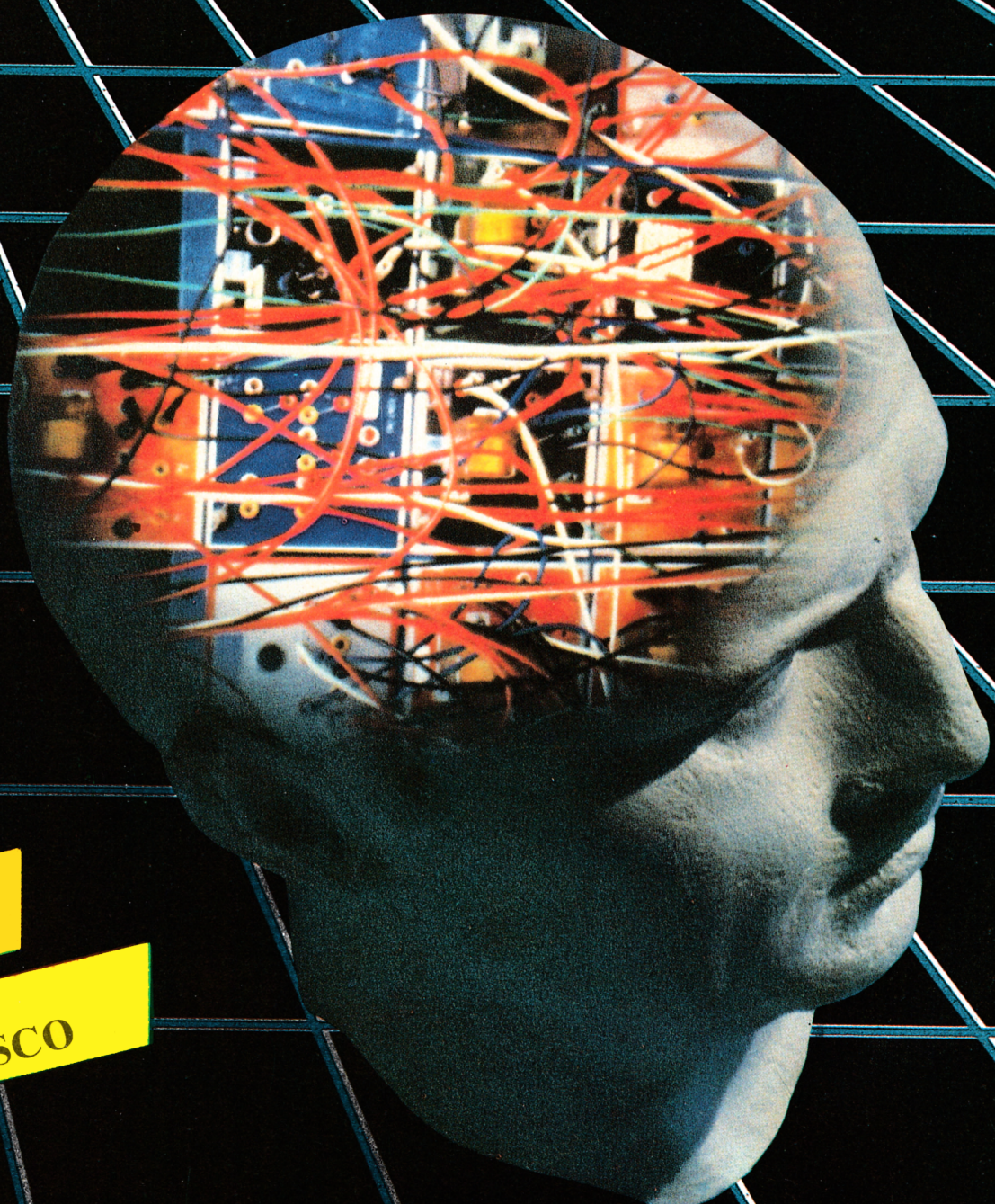
AMSTRAD EDUCATIVO

N.º 8 - 295 Ptas.



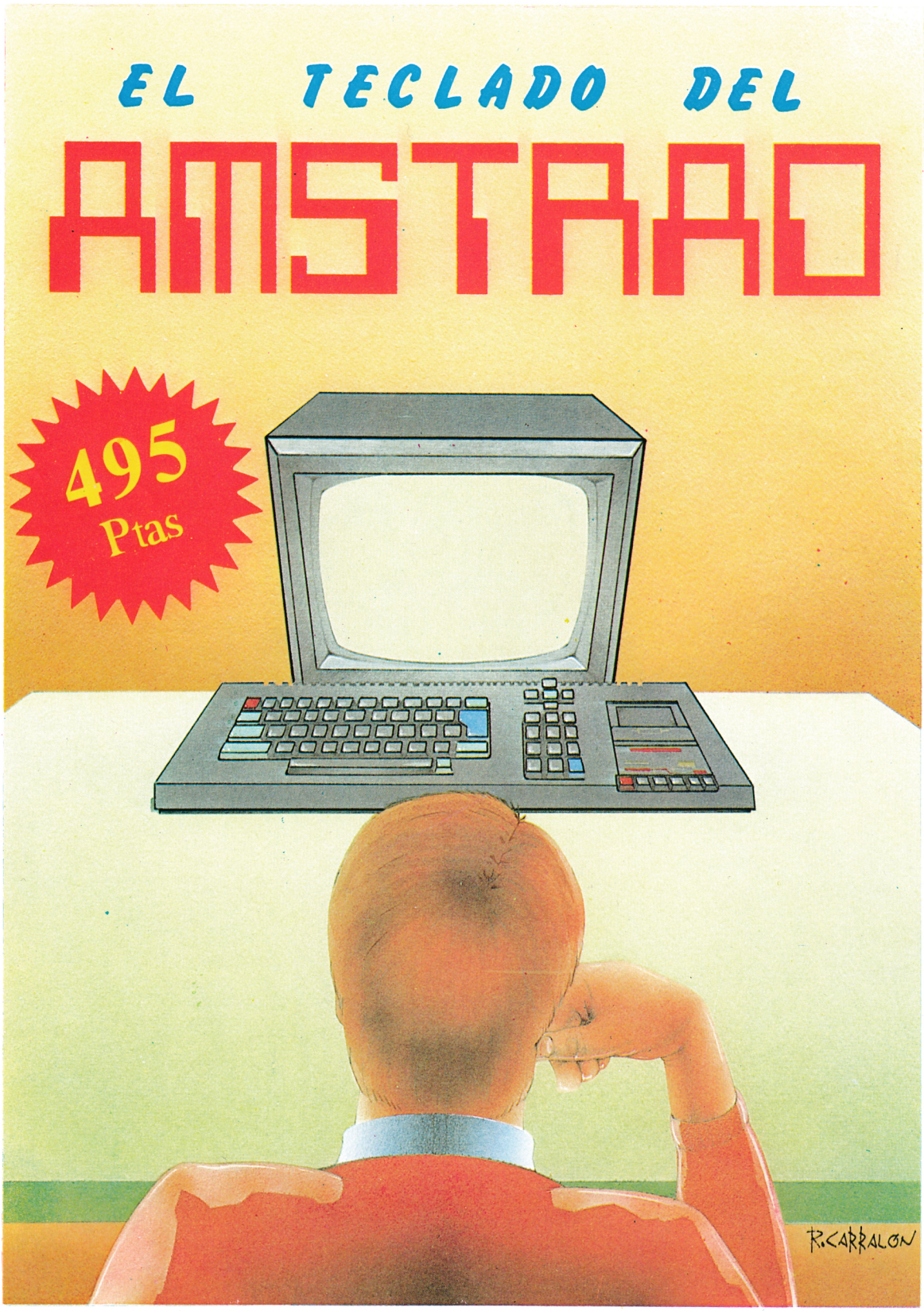
EL AMSTRAD Y EL CP/M

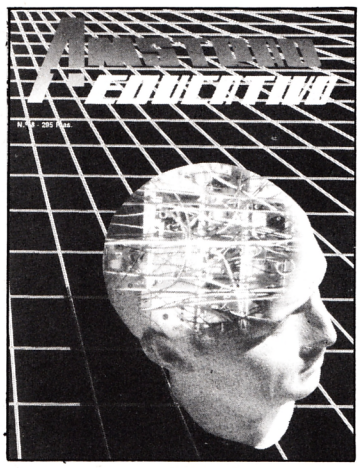
FICHAS DEL AMSTRAD

COMO CREAR FICHEROS EN DISCO

¡ya está a la venta!

EL TECLADO DEL AMSTRAD





editorial

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.

STAT

por A. Bellido

Función asignada: Mostrar o cambiar el estado de discos, ficheros y dispositivos periféricos, de acuerdo con el siguiente esquema:

1.º Mostrar:
   1. Un resumen informativo sobre el almacenamiento en disco de un fichero o grupo de ficheros.
   2. El estado actual de todos o uno de los discos instalados en todas o una de las unidades de disco que esten en línea.
   3. Las características generales de todos o uno de los discos instalados en todas o una de las unidades de discos que estén en línea.
   4. La asignación actual de los dispositivos periféricos.

2.º Cambiar:
   1. El estado de un fichero o grupo de ficheros.
   2. El directorio de un disco.
   3. La asignación actual de los dispositivos periféricos.

Comentarios: El programa STAT.COM es muy flexible como se deduce de la variedad de funciones que tiene encomendadas, lo que exigirá al operador conocer las subórdenes —operandos— que deben ser añadidos a STAT para completar adecuadamente una línea de orden.

En las explicaciones que siguen se da por supuesto que al referirnos a un grupo de ficheros nombrados con ambigüedad (afn), podemos, si conviene, referirnos a un sólo (ufn).

En este orden de cosas, cuando aparezca la expresión "R/W" se está haciendo referencia a READ/WRITE en el sentido de permitir la lectura y escritura en un disco o fichero. R/O (READ/ONLY) viene a significar "SOLO LECTURA" en un disco o fichero. El término SYS alude a ficheros cuyo nombre ha sido excluido, o se excluye, del listado del DIRectorio. Por el contrario, DIR vuelve a introducir un fichero en el listado del directorio siempre, claro está, que haya sido excluido previamente. Para nombrar genéricamente una cualquiera de las unidades de disco, recurriremos a "d".

Estos conceptos quedarán suficientemente claros a lo largo del presente epígrafe. La orden STAT actúa sobre los ficheros asociados al área de usuario activa.

El tema de las asignaciones de los dispositivos periféricos a los dispositivos lógicos que CP/M controla, se estudia en el apéndice correspondiente, para evitar, de esta forma, una discontinuidad en el planteamiento didáctico impuesto a este trabajo.

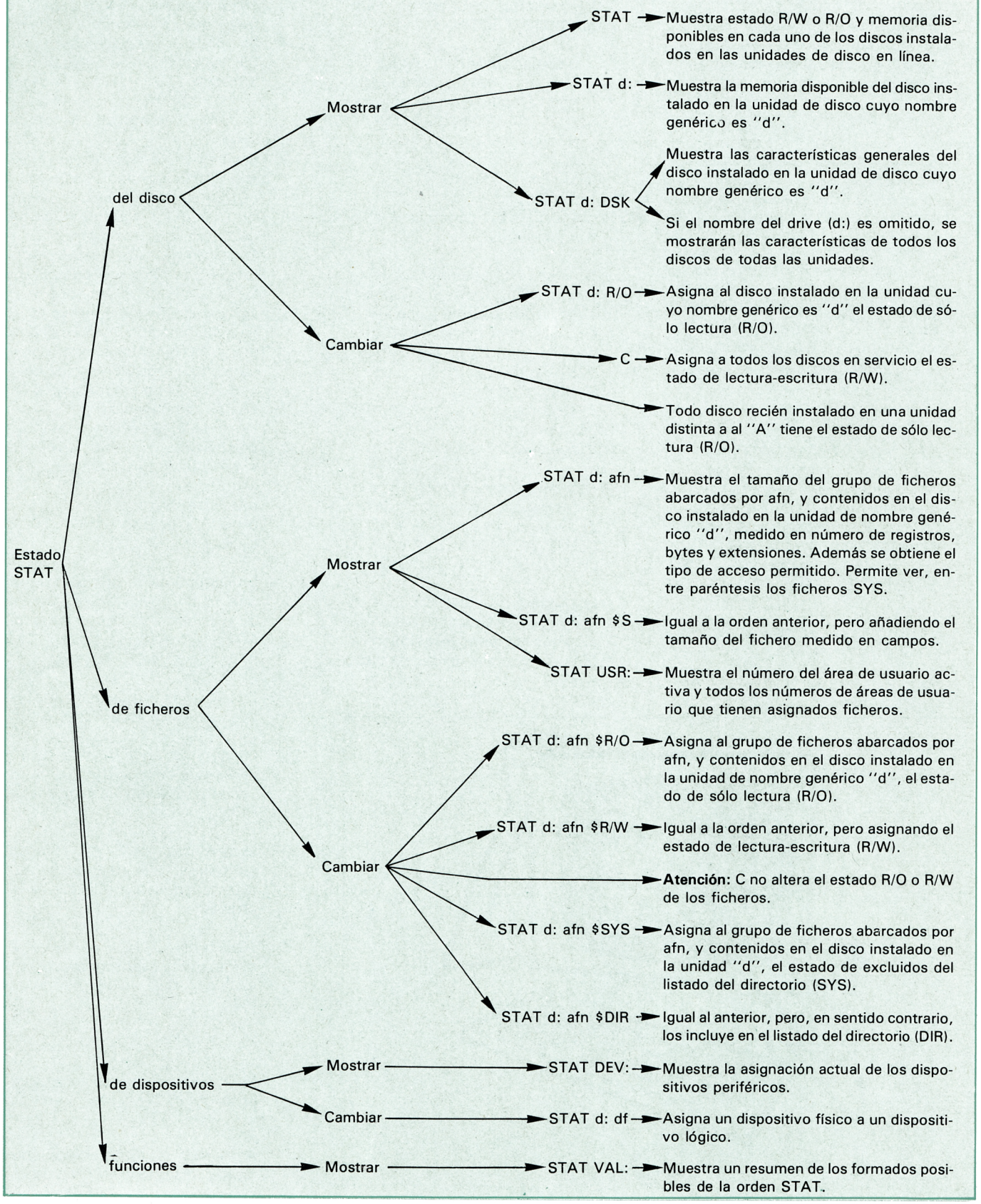
De momento, veamos un esquema del conjunto de órdenes que pueden ser dadas con STAT.

Formas de obtenerlas: Instalar un disco que contengan el programa STAT.COM. La sintaxis que requiere una línea de orden que comience por STAT, está sometida a las siguientes reglas generales:

1.º Después de teclear STAT, se debe dejar un espacio antes de añadir cualquier operando.
2. Es conveniente aceptar como norma no dejar espacios antes y después de los dos puntos (:).
3.º Antes del símbolo $ dejar un espacio, después no.
4. Cuando sea permitido varias subórdenes dentro de una línea de orden STAT se deberán separar por una coma (,).

La línea de orden se da por concluida y se ejecuta al pulsar <cr>.

ESQUEMA DE ORDENES Y FUNCIONES POSIBLES CON STAT



Ejemplos:

1.° ¿Qué orden CP/M se ha de dar para obtener el estado de todos los discos instalados en las unidades de disco conectadas, así como la capacidad de almacenamiento disponible en cada uno de ellos?

A > STAT

Al pulsar <er> obtendremos, por ejemplo:

A: R/W, Space: 18 K
B: R/O, Space: 112 K

Con lo que deducimos que el disco de la unidad A permite la lectura y escritura y aún restan por ocupar 18 K, mientras que el disco de la unidad B es de sólo lectura, y permite almacenar 112 K más.

2.° ¿Qué orden CP/M se ha de dar para averiguar la capacidad de almacenamiento disponible en el disco instalado en una unidad de disco predeterminada (la B, por ejemplo)?

A >STAT B:

Al pulsar <cr> obtendremos, por ejemplo:

BYTES REMAINING ON B: 18 K

Este mensaje expresa la capacidad de almacenamiento disponible en el disco instalado en la unidad B: 18 K bytes.

3.° ¿Qué orden CP/M se ha de dar para conocer todas las características que afectan a un disco instalado en una unidad de disco especificada (la A, por ejemplo)?

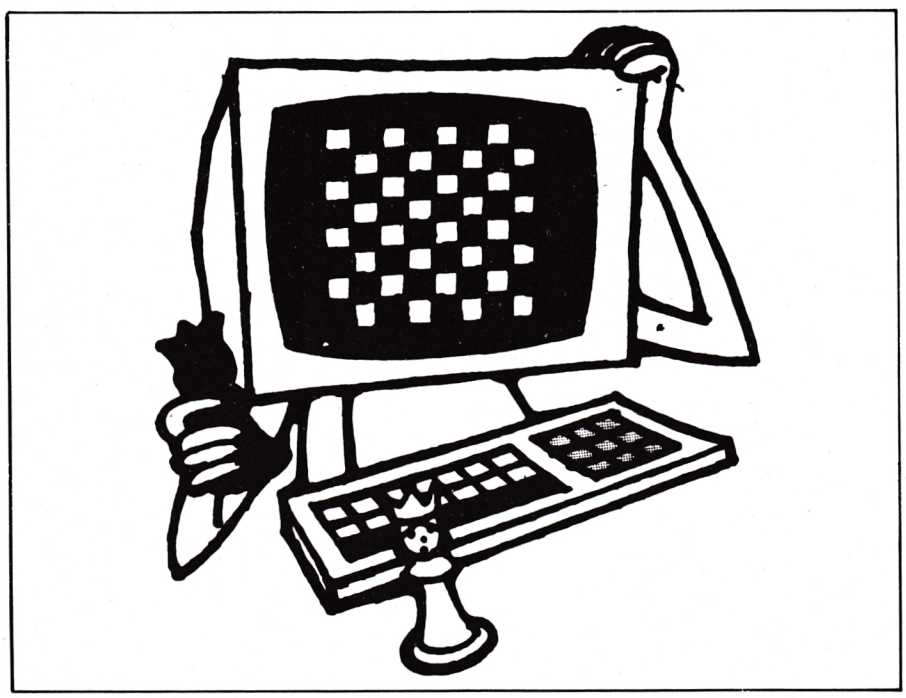
A >STAT A: DSK:

Al pulsar <cr> aparecerá el siguiente listado:

A: Drive Characteristics
1368: 128 Byte Record Capacity
171: Kilobyte Drive Capacity
64: 32 Byte Directory Entries
64: Checked Directory Entries
128: Records/Extend
8: Records/Extend
8: Records/Block
36: Sectors/Track
2: Reserved Tracks

Para interpretar correctamente las líneas que siguen, en caso de duda, repasar el epígrafe "REFERENTE AL ALMACENAMIENTO EN DISCO DE UN FICHERO".

La primera línea del listado anterior "A: Drive Characteristics" indica que los datos que siguen están referidos a la unidad de disco A.



La segunda expresa la capacidad de almacenamiento total del disco medido en unidades de 128 bytes que es la longitud de un registro, de donde podemos inferior que la capacidad en bytes será de 1368 * 128 = 175104 bytes < > 172 K bytes, que es justamente lo que muestra la tercera línea.
El cuarto renglón manifiesta el número máximo de ficheros que pueden ser catalogados (en la pista catálogo) a razón de 32 bytes de referencia para cada uno.
El quinto muestra cuantas referencias de ficheros comprueba CP/M en la pista catálogo para cerciorarse de si un nuevo disco ha sido instalado en la unidad de disco. Normalmente toda la pista catálogo, razón por la cual coincide con el número máximo de ficheros catalogables.
La sexta línea declara la cantidad de registros que conforman una extensión, lo que equivale a 128 registros/extensión * 128 bytes/registro = 16384 bytes/extensión < > 16 K bytes/extensión.

En la séptima se precisa que un fichero no puede ocupar menos de 8 ficheros o, lo que es igual, 1024 bytes (8 registros * 128 bytes/registro).

En la octava vemos que cada pista está dividida en 36 "sectores".

Finalmente, en la novena, se observa que existen dos pistas reservadas que son las dedicadas al DOS.

Esta función de STAT se usará normalmente al comenzar a manejar una nueva configuración de ordenador.

4. ¿Qué orden CP/M se ha de dar para impedir la escritura en el disco instalado en la unidad por defecto?

A < STATA: = R/O

Para volver al estado de lectura-escritura (R/W) basta con dar un C.

Al instalar un disco en cualquier unidad distinta a la omitida, es considerada por CP/M como R/O.

5.° ¿Qué orden CP/M se ha de dar para conocer el estado y tamaño de todos los ficheros del tipo. BAS contenidos en el disco instalado en la unidad por defecto?

A ⩾ STAT * .BAS

Al pulsar <cr> se obtendría un listado de este tipo:

```
Recs  Bytes  Ext  Acc
158   20 K   2    R/W  A: DISC .BAS
 11    2 K   1    R/W  A: EX 1 .BAS
  3    1 K   1    R/W  A: EX 2 .BAS

Bytes Remaining On A: 18 K
```

De donde extraemos, por ejemplo, que el fichero EX 2.BAS permite la lectura y escritura (R/W), ocupando 3 registros, 1 K byte y 1 extensión. Adicionalmente indica que aún quedan 18 K bytes disponibles.

6.° ¿Qué orden CP/M se ha de dar para proteger contra la escritura a todos los ficheros de nombre BASIC contenidos en el disco instalado en la unidad por defecto?

A > STAT BASIC.* $R/O

Al ejecutar la orden este listado indicará la nueva situacion.

```
BASIC. 9 set to R/O
BASIC. 1 set to R/O
BASIC. 2 set to R/O
BASIC. 6 set to R/O
BASIC. 4 set to R/O
BASIC. 7 set to R/O
BASIC. 8 set to R/O
```

Para volver a la situación R/W, la orden será

A > STAT BASIC.* $R/W

Con lo cual el listado cambiará a:

```
BASIC. 9 set to R/W
BASIC. 1 set to R/W
BASIC. 2 set to R/W
BASIC. 6 set to R/W
BASIC. 4 set to R/W
BASIC. 7 set to R/W
BASIC. 8 set to R/W
```

7.° ¿Qué órdenes CP/M hay que dar para eliminar del listado del catálogo del disco instalado en a unidad por defecto, todos los ficheros de nombre BASIC y, posteriormente, volver a introducir?

1.—
A > STAT BASIC.* $SYS
Al pulsar <cr> aparecerá:

```
BASIC. 9 set to SYS
BASIC. 1 set to SYS
BASIC. 2 set to SYS
BASIC. 6 set to SYS
BASIC. 4 set to SYS
BASIC. 7 set to SYS
BASIC. 8 set to SYS
```

Si en estas condiciones se pidiera un DIR, este conjunto de programas no aparecería en el directorio.

Para devolverlos a su estado inicial, se ordenará:

A > STAT BASIC.* $DIR

Con lo cual el listado cambiará a:

```
BASIC. 9 set to DIR
BASIC. 1 set to DIR
BASIC. 2 set to DIR
BASIC. 6 set to DIR
BASIC. 4 set to DIR
BASIC. 7 set to DIR
BASIC. 8 set to DIR
```

A partir de este momento, estos programas volverían a estar incluidos en el DIRectorio del disco.

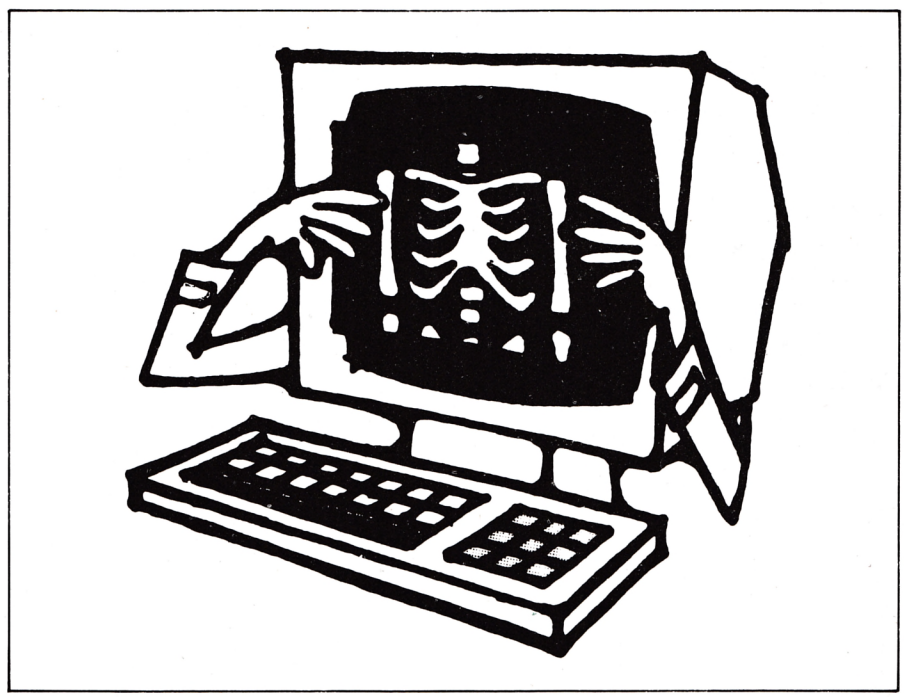
8.° ¿Qué orden CP/M se ha de dar, para conocer la actual asignación de dispositivos periféricos a los lógicos de CP/M?

a > STAT DEV:

Al ejecutar la orden, obtendríamos una situación que podría corresponder a la siguiente:

```
CON: IS CRT:

RDR: IS TTY:
PUN: IS TTY:
LST: IS LPT:
```



La interpretación de este listado exige la lectura del apéndice dispositivos, no obstante, en resumen, viene a decir que la consola del operdor (CON) tiene asignado un monitor de video (CRT), que como lectora de información se utiliza una terminal similar a un teletipo, que como salida de información tiene un dispositivo igual al anterior y para listar las salidas de información se utiliza una impresora.

9.° ¿Cómo se podría obtener un resumen de todas las funciones de la orden STAT?

A > STAT VAL:

Mostraría:

Temp R/O Disk: d = R/O
Set Indicator: d; filename, typ $R/O $R/ W $ SYS $DIR
Disk Status: DS: d: DSK:
User Status: USR:
lobyte Assign:
CON: = TTY: CRT: BAT: UCI:
RDR = TTY: PTR: UR1: UR2:
PUN = TTY: PTP: UP1: UP2:
LST: = TTY: CRT: LPT: UL1:

Cuya interpretación es la siguiente:

1.ª línea: Temp R/O Disk: d: = R/O.

Indica que, para una asignación temporal de estado de sólo lectura (R/O) a una unidad de disco, se deberá añadir a su nombre los símbolos que aparecen (: = R/O).

2.ª línea: Set Indicator: d: filename. tip $R/O $R/W $SYS $DIR.

Indica el conjunto de operadores que se pueden añadir a STAT según las alteraciones en el estado de los ficheros que se pretendan conseguir, y que han sido expuestos en este epígrafe.

3.ª línea: Disk Status: DSK: d: DSK.
Aquí se resumen las posibilidades que tiene el operador para averiguar las características generales que corresponden al disco instalado en la unidad d: o de todas las unidades si se omite el nombre de la unidad.

4.ª línea: User Status: USR:

Muestra la suborden que pemite a STAT informar sobre los ficheros en las áreas de usuario.
El resto de las líneas sintetiza las posibilidades de asignación de dispositivos periféricos a los lógicos: CON, RDR, PUN LST.

Errores posibles:

1.º El más usual procede de faltas de sintaxis del tipo, por ejemplo, de dejar un espacio entre el nombre de la unidad de disco y los dos puntos.

Este sería el caso en

A > B:

Con lo que CP/M buscaría un fichero denominado B y, al no encontrarlo, enviaría un mensaje similar a FILE NOT FOUND o NO FILE.

2.º Otro error puede provenir de no contener el disco de la unidad activa el programa STAT.COM, de tal forma que al dar la orden

A > STAT

se obtendría

STAT?

Indicativo de que tal fichero no se encuentra en el disco.

3.º Utilizar STAT referido a una unidad de disco más allá de las que estén en línea CP/M detectaría la situación de la siguiente forma:

A > STAT D:
BDOS ERR ON D: SELECT

El operador debería tratar de cancelar pulsando C.

COMO CREAR FICHEROS EN DISCO

por A. Bellido

FICHEROS SECUENCIALES. PRIMER PASO

En el epígrafe "OPCIONES DE USO DE UN FICHERO" se resumían las alternativas que un fichero puede ofrecer a un usuario: Altas, bajas, modificaciones, consultas y listados más claro está, la propia creación del fichero. Esto se puede resumir en la siguiente pantalla de opciones:

```
*** OPCIONES ***
1... FICHERO NUEVO
2... ALTAS
3... BAJAS
4... MODIFICACIONES
5... CONSULTAS
6... LISTADOS
```

Pulse el número de la opción elegida.

A continuación se exponen las diferentes secciones de un programa capaz de respaldar a cada una de las opciones ofrecidas en el anterior **menú** incluida la propia pantalla de **OPCIONES**.

En primer lugar vamos a centrar nuestra atención en desarrollar un pequeño listado que haga operativa la anterior oferta.

```
10 REM *************FICHERO SECUENCIAL*
***************
20 CLS
30 + + + + + + + + + + + + + + + + + + + + + + +
40 REM *** PANTALLA DE OPCIONES ***
50 PRINT "*** OPCIONES *** "
60 PRINT "1... FICHERO NUEVO"
70 PRINT "2... ALTAS"
80 PRINT "3... BAJAS"
90 PRINT "4... CONSULTAS"
100 PRINT "5... MODIFICACIONES"
110 PRINT "6... LISTADOS"
120 PRINT "7... RETORNO AL BASIC"
130 PRINT
140 INPUT "Pulse el número de la opción elegida"; 0
150 ON O GOSUB 1000, 2000, 3000, 5000, 6.000,
7.000
160 GOTO 20
170'
```

Según este listado, la rutina para crear un nuevo fichero debe estar situada a partir de la línea 1.000, y desde esta línea en adelante vamos a escribir un programa que nos permita generar uno secuencial para controlar una biblioteca en base a los títulos y autores de los libros que lo integran.

CREACION

```
1000 >>>>>>>>>>>>>>>>>>>>>>>>>
1010 REM *** NUEVO FICHERO ***
1020 CLS
1030 CR = 1
1040 OPENOUT "SECUENCI.ALF"
1050 PRINT //9, CR
1060 CLOSEOUT
1070 RETURN
1080 <<<<<<<<<<<<<<<<<<<<<<<<<
```

Aunque todas las instrucciones que figuran en este listado nos son conocidas, no está de más revisar su significado.

En la línea 1040, se ordena abrir un fichero secuencial de nombre "SECUENCI.ALF" que permita la salida de información hacia él a través del canal 1. (Modo escritura).

En la línea 1050, se ordena la grabación o escritura del contenido de la variable CR en el primer registro del fichero "SECUENCI.ALF".

En 1060 se cierra el fichero previamente abierto.
Tras la ejecución de esta rutina el fichero en cuestión está creado, y contiene en su primer y único registro un 1. Este número corresponde al número de registros que, a la sazón, integran el fichero.

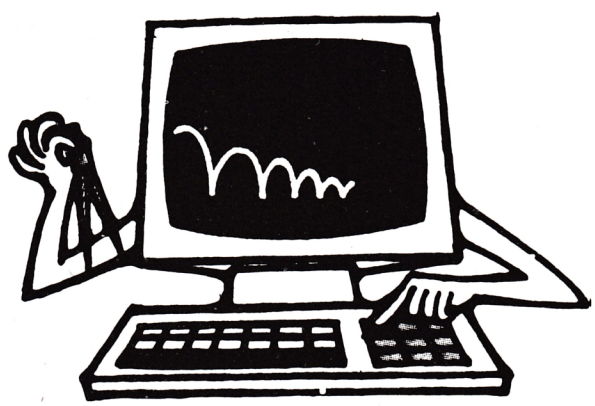
LISTADOS

Conseguir un listado en un fichero secuencial consiste en leer los registros e ir imprimiendo sus contenidos uno tras otro.

```
6000 >>>>>>>>>>>>>>>>>>>>>>>>>
6010 REM *** LISTADOS ***
6020 OPENIN "SECUENCI.ALF"
6030 INPUT # 9, CR
6040 FOR N = 2 TO CR
6050 INPUT # 9, T$, A$
6060 PRINT "FICHA"; N-1
6070 PRINT "TITULO: ";T$
6080 PRINT "AUTOR: ";T$
6090 IF N/7 < > INT (N/7) THEN 6120
6100 PRINT "PULSE UNA TECLA PARA
CONTINUAR"
6110 IF INKEY$ = "" THEN 6110
6120 NEXT N
6130 CLOSEIN
6140 INPUT "MAS LISTADOS: ";0$
6150 IF 0$ = "S" OR 0$ = "s" THEN 6020
6160 RETURN
6170 <<<<<<<<<<<<<<<<<<<<<<<<<
```

RETORNO AL BASIC

```
7000 >>>>>>>>>>>>>>>>>>>>>>>>>
7010 REM *** RETORNO AL BASIC ***
7020 END
7030  <<<<<<<<<<<<<<<<<<<<<<<<<
```



EJERCICIO 1

Comprobar el efecto de un cierre intempestivo en un fichero abierto en modo escritura (''O'')

El método más fácil de ver las consecuencias que produce el caso propuesto consiste en abrir un fichero en modo escritura, introducir ciertos datos, cerrarlo, comprobar el contenido de tal fichero, volverlo a abrir en modo escritura e inmeditamente, cerrarlo. Tras este último paso el fichero ha perdido su contenido.

OPENOUT, ''PRUEBA'' PRINT #9, ''PRUEBA UNO'' CLOSEOUT	Introducción de datos
OPENIN, ''PRUEBA'' INPUT #9, A$ CLOSEIN PRINT A$	Comprobación
OPENOUT, ''PRUEBA'' CLOSEOUT	Cierre del fichero antes de escribir el primer registro.
OPENIN, ''PRUEBA'' INPUT #9, A$ Input past end	El mensaje ''Input past end'' indica que se ha alcanzado el fin del fichero, y como sólo había un dato en él esto significa que se ha perdido toda la información de los registros posteriores al cierre.

EJERCICIO 5

Transferir el contenido de un fichero secuencial a una matriz.

En lo que sigue se admite que estamos trabajando sobre el fichero SECUENCI.ALF en el que su primer registro está reservado para el contador de registros ocupados (CR).

```
10 >>>>>>>>>>>>>>>>>>>>>>>>>>>
20 REM ***
30 OPENIN ''SECUENCI.ALF''
40 INPUT # 9, CR
50 DIM T$ (CR-2): DIM A$(CR-2)
60 FOR N = 0 TO CR-2
70 INPUT # 9, T$ (N), A$(N)
80 NEXT N
90 CLOSEIN
```

El límite superior del bucle (CR-2) viene impuesto por el hecho de partir los subíndices de los elementos de la matriz de ∅ y por estar reservado el primer registro para el contador CR.

EJERCICIO 6

Clasificación alfabética de un fichero secuencial.

Para lograr este objetivo hay que comenzar por transferir el contenido del fichero a una matriz, ordenar alfabéticamente la matriz y, finalmente, transferir la matriz ya clasificada al fichero.

```
10 >>>>>>>>>>>>>>>>>>>>>>>>>>>
20 REM *** CLASIFICACION DE UN FICHERO SE-
CUENCIAL ***
30 OPENIN ''SECUENCI.ALF''
40 INPUT # 9, CR
50 DIM T$ (CR-2): DIM A$ (CR-2)
60 FOR N = 0 TO CR-2
70 INPUT # 9, T$ (N), A$(N)
80 NEXT N
90 CLOSEIN
100 D = INT [(CR-1) /2]
110 WHILE D> = 1
120 CC = 0
130 WHILE CC< = (CR-1)-D-1
140 VA = CC
150 IF T$ (VA) >T$ (VA + D) THEN SWAP T$(VA),
T$(VA + D): SWAP A$(VA), A$(VA + D): VA = VA-
D: IF VA> = 1 THEN 150
160 CC = CC + 1
170 WEND
180 D = INT (D/2)
190 WEND
191 PRINT
200 OPENOUT, ''SECUENCI.CLA''
210 PRINT //9, CR
220 FOR N = 0 TO CR-2
230 PRINT // 9, T$ (N); '',''; A$ (N)
```

El listado siguiente permite comprobar el efecto de la clasificación:

```
270:
300 OPENIN "SECUENCI.CLA"
310 INPUT # 9, CR
320 FOR X = 2 TO CR
330 INPUT # 9, T$, A$
340 PRINT T$, A$
350 NEXT X
360 CLOSEIN
```

EJERCICIO 7

Consultas a un fichero clasificado alfabéticamente.

La principal ventaja que ofrece un fichero clasificado alfabéticamente se deriva del hecho de permitir una búsqueda dicotómica. Para lograr esto, tendremos que empezar por transferir el fichero ya clasificado, p.e.: SECUENCIA.CLA del ejercicio anterior, a una matriz. A partir de aquí, se procede conforme a lo estudiado en el epígrafe dedicado a "LOCALIZACION".

```
10 >>>>>>>>>>>>>>>>>>>>>>>>>>>>
20 REM *** BUSQUEDA DICOTOMICA ***
30 OPENIN, "SECUENCI.CLA"
40 INPUT # 9, CR
50 DIM T$ (CR-2): DIM A$ (CR-2)
60 FOR N = 0 TO CR-2
70 INPUT # 9, T$ (N), A$(N)
80 NEXT N
90 CLOSEIN
100 INPUT "TITULO A LOCALIZAR": ";TI$
110 PT = 0: OUT = CR-2
120 IF TI$<T$(PT) OR TI$>T$(UT) THEN 190
130 IF TI$ = T$(PT) THEN L = PT: GOTO 200
140 IF TI$ = T$(UT) THEN L = UT: GOTO 200
150 MITAD = INT (UT + PT) / 2)
160 IF TI$ = T$ (MITAD) THEN L = MITAD: GOTO
200
170 IF TI$>T$ (MITAD) THEN PT = MITAD ELSE UT
= MITAD
180 IF UT-PT>1 THEN 150
190 PRINT "TITULO INEXISTENTE": END
200 PRINT "TITULO: ";T$(L)
210 PRINT "AUTOR: "A$(L)
220 END
```

FOR... TO/NEXT FOR... TO...STEP/NEXT	PARA... HASTA/PROXIMO PARA...HASTA...SALTO/PROXIMO

FOR variable = { expresión numérica 1 } TO { expresión numérica 2 } /NEXT variable

INTERPRETACION

La variable va cambiando de valor —de unidad en unidad— partiendo del valor dado por la expresión numérica 1, y hasta alcanzar el valor dado por la expresión numérica 2.

El cambio de valor de la variable se producirá —normalmente, de unidad en unidad— cuando en la sencuencia normal de lectura se encuentre la sentencia.

NEXT variable

Con ella, el programa cambia su sencuencia de lectura a la línea donde se encuentra la sentencia inicial FOR... TO.

POSIBILIDADES

El cambio de valor de la variable puede ser distinto de la unidad. Para ello, bastará con añadir la sentencia **STEP seguida de la expresión numérica 3** a la sentencia FOR... TO, donde la expresión numérica 3 determinará el salto que ha de dar la variable al llegar a NEXT, quedando la estructura de la sentencia así:

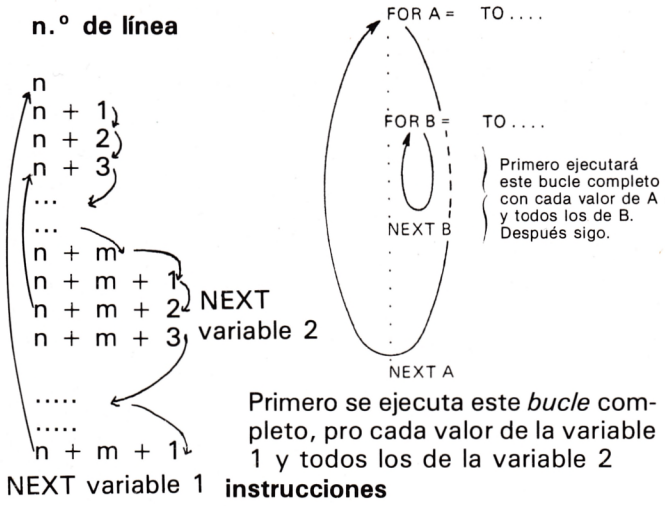
En esta sencuencia, la variable que sigue a FOR sólo puede ser una letra.

Esta sentencia porduce un *bucle de lectura* que se repite, una y otra vez, hasta que variable alcanza el va-

FOR variable = { expres. numér.1 } TO { expres. numér.2 } STEP { expres. numér. 3 } /NEXT variable

lor de la expresión numérica 2, pudiéndose representar esquemáticamente de la siguiente forma:

Bucle FOR... TO/NEXT normal (de unidad en unidad)



Primero se ejecuta este *bucle* completo, pro cada valor de la variable 1 y todos los de la variable 2

instrucciones

FOR variable = { **expres. numér. 1** } **TO** { **expres. numér. 2** }

FOR variable 2 = { **expres. numér. 3** } **TO** { **expres. numér. 4** }

La secuencia normal de lecturas se restablece cuando la variable 1 se hace igual a la expresión numérica 2. En ese momento, la siguiente línea a leer es aquélla que sigue a la sentencia NEXT.

FORMA DE TECLEAR LAS INSTRUCCIONES

Teclear FOR, el nombre de la variable numérica, el signo igual (=), la expresión numérica 1, TO y la expresión numérica 2, seguido de ENTER, para que el valor de la variable cambie *de unidad en unidad.*

Teclear FOR, el nombre de la variable numérica, el signo igual (=) la expresión numérica 1, TO, la expresión numérica 2, STEP y la expresión numérica 3, seguido de ENTER, para que el valor de la variable cambie *de n en n unidades* indicadas en la expresión numérica 3.

Secuecia de instrucciones del bucle.

Teclear NEXT y el nombre de la variable numérica, seguido de ENTER.

EJEMPLO

1. Confeccionar un programa que, una vez ejecutado, imprima en pantalla los resultados de la tabla de multiplicar por 2 desde ∅ hasta ∅

PROGRAMA	COMENTARIOS
10 FOR Y = ∅TO 10	En este programa nombramos la variable con la letra Y; la expresión numérica 1 la hacemos igual a ∅ y la expresión numérica 2 la hacemos igual a 10.
20 LET A = 2* Y	En la línea 20, damos a la variable A el valor que resulte de multiplicar por 2 el valor actual de la variable Y —que tomará sucesivamente los valores 0, 1, 2, 3, 4, 5, 6, 7, 8, 9 y 10.
30 PRINT A	La línea 30 obliga a imprimirse el valor actual de A.
40 NEXT Y	Al llegar a la línea 40, el programa seguirá su lectura en la línea 10, siempre y cuando el valor de 6 no haya llegado a 10. En otro caso, pasaría a la línea 50 y el programa parará.

EJEMPLO

2. Escribir un programa que, una vez ejecutado, imprima en pantalla los números impares del 1 al 50.

PROGRAMA	COMENTARIOS
10 FOR X =1 TO 50 STEP 2 20 PRINT X 30 NEXT X	En este caso, obligamos a saltar a la variable X de dos en dos —gracias a STEP 2—, partiendo del valor 1 y hasta llegar al valor 50.

EJERCICIOS

1. Escribir un programa que, ejecutado, imprima en pantalla los números del 0 al 100.
2. Escribir el programa que, ejecutado, imprime en pantalla diez veces la palabra *"pollo"* en una columna, y otro programa que, al ejecutarlo, imprima en pantalla diez veces seguidas la palabra *"pio"*.
3. Escriba el programa que, ejecutado, imprima en pantalla en diez líneas y dos columnas diez veces la palabra *"pollo"* y diez veces la palabra *"pío"*.
4. Escribir los programas que, una vez ejecutados, impriman en pantalla:
   - 4.1. Diez veces seguidas la palabra *"pollo"* y otras diez veces seguidas la palabra *"pío"*. (*).
   - 4.2. Diez veces en columna la palabra *"pollo"* y otras diez veces en la misma columna la palabra *"pío"* (*).
   - 4.3. Cinco veces la palabra *"pollo"* y cinco veces la palabra *"pío"* en dos columnas (*).
   - 4.4. Diez veces la palabra *"pollo"* en una columna y diez veces la palabra *"pío"* en otra columna, de modo que haya veinte líneas en pantalla en total (*).
5. Escriba el programa que, ejecutado, imprima en pantalla la tabla de sumar 5 desde 0 hasta 10.
6. Escriba el programa que, ejecutado, imprima en pantalla las tablas de sumar 1, 2, 3... 10 desde 0 hasta 10.
7. Escriba el programa que, ejecutado, imprima en pantalla los números pares menores que 100.
8. Escriba el programa que, ejecutado, imprima en pantalla los múltiplos de 5 desde 0 hasta 100.
9. Escribir el programa que, dándole un número N del 0 al 10, nos diga si es correcto o no el producto de N por otro número M también del 0 al 10.
10. Idear sus propios programas utilizando los comandos PRINT, LET, INPUT, FOR... TO/NEXT, FOR... TO... STEP/NEXT, IF y GO TO.

SOLUCIONES

1.
```
 1 REM Ficha:" FOR / TO / NEXT " Ej.1
10 PRINT "NUMEROS"
20 FOR N=1 TO 20

30 PRINT N;"    ";n+20;"    ";N+40;"    ";N+60;"    ";
   N+80
40 NEXT N
```

2.
```
 1 REM Ficha:" FOR / TO : NEXT " Ej.2"
10 FOR N=1 TO 10: PRINT "Pollo": NEXT N
20 FOR N=1 TO 10: PRINT "Pio": NEXT N
```

3.
```
 1 REM Ficha:" FOR / TO : NEXT " Ej.3"
10 FOR N=1 TO 10: PRINT "Pollo","Pio": NEXT N
```

5.
```
 1 REM Ficha:" FOR / TO : NEXT " Ej.5"
10 PRINT "TABLA DE SUMAR 5"
20 FOR N=0 TO 10
30 PRINT N;" mas ";5;" = ";N+5
40 NEXT N
```

6.
```
 1 REM Ficha:" FOR / TO : NEXT " Ej.6"
10 PRINT "TABLA DE SUMAR": LET A=1
20 FOR N=0 TO 10
30 PRINT N;" + ";A;" = ";N+A
40 NEXT N
50 LET A=A+1: IF A>10 THEN  STOP
60 GO TO 20
```

7.
```
 1 REM Ficha:" FOR / TO : NEXT " Ej.7"
10 PRINT "NUMEROS PARES"
20 FOR N=2 TO 98 STEP 2
30 PRINT N;"  ";
40 NEXT N
```

8.
```
 1 REM Ficha:" FOR / TO : NEXT " Ej.8"
10 PRINT "MULTIPLOS DE 5"
20 FOR N=1 TO 20
30 PRINT N*5;" ";
40 NEXT N
```

9
```
 1 REM Ficha:" FOR / TO : NEXT " Ej.9"
10 PRINT "Di un numero del 1 al 10"
20 INPUT N: PRINT N
30 FOR I=1 TO 10
40 PRINT N;"*";I;"=";
50 INPUT M: PRINT M
60 IF M=N*I THEN  GO TO 90
70 PRINT "No, intentalo otra vez"
80 GO TO 40
90 PRINT "SI"
100 NEXT I
110 STOP
```

GUIA DE PALABRAS BASIC

GUIA DE PALABRAS BASIC

RANDOMIZE. ''expresión''

Inicializa el generador de números aleatorios con la *expresión,* con lo que la secuencia de números aleatorios comenzará con distintos valores.

Si *expresión* se sustituye por TIME, RANDOMIZE produce números lo más aleatorios posibles.

```
10 RANDOMIZE TIME
20 FOR X = I TO 20
30 PRINT RND
40 NEXT X
```

REM ''expresión''

Permite introducir los comentarios dados en *''expresión''* dentro de cualquier línea de programa sin que interrumpa ni intervenga en el proceso general.

```
10 REM ''Estudio de cargas''
```

RESTORE número

Con esta sentencia le permite comenzar la lectura de los constantes desde la primera del DATA situado en el número de línea dada por *número.* Si el número de línea no se especifica, la lectura de constantes se iniciará en la línea del primer DATA del programa.

Después de que un programa lee una sentencia RESTORE, la primera sentencia READ que ejecute el programa accederá a la primera constante, según el criterio expuesto más arriba.

Teclear RESTORE, y el *número de línea* seguido de <cr> para volver a leer las constantes que hay en el DATA de la línea en cuestión. Teclear RESTORE seguido de <cr>, para volver a leer las constantes desde el primer DATA del programa.

```
10 FOR X = 10 TO 5
20 READ V : PRINT V
30 RESTORE
40 NEXT X
50 DATA 3
```

La variable V será leída cinco veces.

```
10 FOR X = 1 TO 10
20 READ U$: PRINT U$
30 RESTORE
40 NEXT X
50 DATA ''Caballo'', ''Jinete''
```

La variable U$ es leída 10 veces en la línea 20 tomando el primer valor del DATA de la línea 100.

PROGRAMA comentado en

BASIC

por A. Larumbe

DESCRIPCION Y FUNCIONAMIENTO DEL PROGRAMA TORRES

1. DESCRIPCION

El programa consiste (a grandes rasgos pues en el punto 2 se explicará completamente en trasladar una torre, construída a base de rectángulos de diferente longitud, en el menor número posible de movimientos.

Los rectángulos que componen cada torre son 4 (hay 3 torres) y se generan en las líneas 70-160. El primero de ellos (el inferior y más grande) posee 10 caracteres de longitud. Esta viene dada por la sentencia de la línea 70 (FOR I = 1 TO 10). El siguiente en tamaño y posición, tiene 8 caracteres que están formados mediante la línea 100 (FOR I = 1 TO 8). Así, sucesivamente con los dos restantes. Uno posee 6 caracteres, y el más peque/o tiene 4.

La variable TORRE(I,J) almacena un uno cuando una torre determinada posee un rectángulo; esto es, por ejemplo, al iniciar el juego, la primera torre tiene todos los rectángulos mientras que las dos restantes no tiene ninguno; por ello, la situación de esa variable será la siguiente:

torre (1,4) = 1
torre (1,6) = 1
torre (1,8) = 1
torre (1,10) = 1
torre (2,4) = 0
...
...
torre (3,4) = 0
...
torre (3,10) = 0

En el momento que hagamos un movimiento a otra torre, la variable TORRE(1,J) correspondiente al rectángulo origen se podrá a 0 indicando que ya no está disponible en esa torre. Sin embargo, dicha variable relativa al rectángulo destino, se pondrá a 1.

Igualmente, hay otra variable arriba(1) que indica el valor del rectángulo que hay en cada torre:

Si es el de 4 caracteres, contendrá un 4. Si es el de 10, un 10 y así sucesivamente.

Las líneas 180-220 declaran las ventanas a utilizar por el programa:

*** Entre las columnas 1-80 y las filas 1-3 se presentará el mensaje indicativo de las torres: ''TORRE 1... TORRE 3''.**

*** La WINDOW**£2, 1,80, 20, 22 es decir, columnas 1 a 80 y filas 20 a 22 se reserva a presentar las órdenes que precisa la ejecución del programa.

* Las tres últimas ventanas (columnas 5-20, 21-36, 37-52 y filas 5-10) se reservan para presentar cada una de las tres torres (líneas 200-220).

En las líneas 230-250 se presenta, por la ventana 1 (LOCATE£1, 25,1) las cabeceras (TORRE 1...TORRE 3).

Las líneas 260-320 presentan por pantalla la torre inicial que habrá que posponer a otro lado. El sentido de la 270 es el siguiente:

La variable long posee la longitud del rectángulo correspondiente en cada momento, y la variable blanco imprimirá un número determinado de blancos para centrar el rectángulo en 8 caracteres dentro de su ventana (blanco = int (8-long/2); fórmula que proporciona el número de blancos necesarios a la izda. del rectángulo para que este quede centrado.

De esta manera imprimimos el número necesario de blancos antes del rectángulo, siendo este el último en aparecer, tras la impresión de los blancos necesarios (líneas 280-300).

Cuando lleguemos a la línea 325, la torre con todos sus rectángulos colocados cada uno en su lugar correspondiente, habrá sido dibujada y el programa estará dispuesto para aceptar nuestros movimientos que se produzcan a partir de esa línea.

En la línea 330 (por la ventana N.º 2) se solicita la entrada de la torre origen de la cual partirá nuestro rectángulo. Este número de torre se almacenara en una variable llamada O. Lógicamente, cualquier valor que no corresponda con alguna de las torres (menor que 1 o mayor que 3), será rechazado. En la línea 350 se solicita la introducción de la torre a la que nosotros queremos llevar el rectángulo. Viene representada por una variable regular O y al igual que antes, cualquier valor inferior a 1 o superior a 3 (IF d<1 OR d>3) será rechazado.

En la línea 370, lo que hacemos es comprobar que en esa torre exista algún rectángulo para mover (ARRIBA (O) < > pues de lo contrario saldría el mensaje de (MOVIMIENTO ILEGAL) y nos solicitaría otros datos. Si existe algo y consiguientemente lo podemos mover, se comprueba (línea 400) que el posible rectángulo que haya en la torre sea más grande que el que vamos a poner nosotros. Caso contrario, iría a la línea 380 mostrando, de nuevo, el mensaje de error.

La línea 410 contiene la variable que va a poseer el número de movimientos que podamos hacer, y será la que se nos muestre al finalizar la ejecución del programa.

En la línea 420 se ejecuta la tarea antes comentada de actualizar las torres "origen" y "destino". Es decir, torre toma el valor 0 pues el rectángulo desaparece de ella; la variable torre destino, toma el valor 1 puesto que hemos traspasado el rectángulo a dicha posición; y ahora el rectángulo de arriba de la torre "destino", será el antiguo de la torre "origen".

Una vez hecho el movimiento valido, se procede a borrar la ventana correspondiente a la torre "origen" (líneas 430-500) dejándola en el estado que debe tener. El proceso que sigue es:

Imprime, SOLAMENTE aquellos rectángulos cuyas correspondientes variables torre valga 1. En caso de encontrar alguna variable que valga 0 tienen el mismo sentido que las anteriormente citadas 270-320. En la línea 550 ponemos la variable arriba a 0 indicando que, en principio, no hay ningún rectángulo en la posición más alta de la torre. El proceso que viene a continuación (líneas 560-580) es el de selección de la nueva variable arriba y el método para encontrarlo es recorrer la variable torre hasta encontrar el rectángulo más grande. Una vez encontrado el mayor rectángulo saltamos a dibujar la torre "destino".

El proceso es idéntico a los anteriores, imprimiendo únicamente aquellos rectángulos cuya variable torre este a 1. En caso contrario deja la línea en blanco.

Las líneas 710-800 sirven para controlar cuando se produce el final del juego este vendrá dado cuando todas las variables torre (3,X) valgan 1, que es lo que se comprueba en estas líneas. Por esta parte del programa, se pasa cada vez que hacemos un movimiento; por ello se introduce la línea 760 (GOTO 330) para que en el caso de que no sea el final (cuenta < >4) vuelva a pedir otros movimientos y continuar el proceso.

La última parte del programa (líneas 805 al final) posee las calificaciones obtenidas con arreglo al número de movimientos.

2. FUNCIONAMIENTO

El comienzo del juego viene mostrado por la aparición en pantalla de una torre de rectángulos de diferente tamaño cada uno.

El sentido del mismo es llevarse la torre (rectángulo a rectángulo) número 1 a la torre número 3 en el mínimo número de movimientos posibles teniendo en cuenta la siguiente restricción:

* Los rectángulos SIEMPRE han de guardar la misma forma: de manera a que nunca podremos colocar un rectángulo de la base (10 caracteres) sobre el del punto más alto (4 caracteres). Esta regla se generaliza con todas las posibles combinaciones que se puedan hacer con los diferentes rectángulos.

Relación de variables usadas en el programa LAS TORRES:

— SUBINDICADAS:

* Torre (3,10): Almacena 1 cuando una torre posee un determinado rectángulo.
* arriba (3): Almacena el valor del rectángulo que hay en la parte superior de cada torre.
* cuad$ (4): Posee la forma de los 4 rectángulos.

— REGULARES:

* i, j, v: Variables usadas en bucles como índices.
* long: Longitud de cada rectángulo.
* blanco: Número de blancos a imprimir para centrar los rectángulos.
* o$, d$: Variables que almacenan la torre origen y destino, respectivamente.
* o, d: Idem, que o$ y d$; pero en forma numérica.
* Veces: Número de movimientos que se hacen.

PROGRAMA comentado en

BASIC

* Ficha: Rectángulo al que se hace referencia en cada momento.
* cuenta: Usada para comprobar el final del juego. Contiene el número de rectángulos que hay en la torre 3.
* res$: Comentario final.
* y$: Variable que indica si se vuelve a jugar.
* contet$: Idem. que sirve para comprobar si se juega o no.

```
10 REM ************************************************************
20 REM ********** LAS TORRES:ANGEL LARUMBE ************************
30 REM ************************************************************
40 MODE 2:forma$=CHR$(143):DIM torre(3,10)
50 DIM arriba(3)
60 ON BREAK STOP
65 REM -------------- CREACION DE LOS BLOQUES USADOS --------------
70 FOR i=1 TO 10
80 cuad$(1)=cuad$(1)+forma$:torre(1,10)=1
90 NEXT
100 FOR i=1 TO 8
110 cuad$(2)=cuad$(2)+forma$:torre(1,8)=1
120 NEXT
130 FOR v=1 TO 6
140 cuad$(3)=cuad$(3)+forma$:torre(1,6)=1
150 NEXT v
160 cuad$(4)=forma$+forma$+forma$+forma$:torre(1,4)=1
170 arriba(1)=4
175 REM --------------- DIBUJO DE LA CABECERA --------------------
180 WINDOW£1,1,80,1,3
190 WINDOW£2,1,80,20,22
200 WINDOW£3,5,20,5,10
210 WINDOW£4,21,36,5,10
220 WINDOW£5,37,52,5,10
230 CLS£1:LOCATE£1,10,1:PRINT£1,"TORRE 1"
240 LOCATE£1,25,1:PRINT£1,"TORRE 2"
250 LOCATE£1,40,1:PRINT£1,"TORRE 3"
255 REM ---------------- DIBUJO DE LA TORRE INICIAL ----------------
260 FOR i=4 TO 1 STEP -1
270 long=LEN(cuad$(i)):blanco=INT(8-long/2)
280 FOR j=1 TO blanco
290 PRINT£3," ";
300 NEXT
310 PRINT£3,cuad$(i)
320 NEXT
325 REM ----------------- INTRODUCCION DE MOVIMIENTOS ----------------
330 CLS£2:INPUT£2,"Origen",o$:o=VAL(o$)
340 IF o<1 OR o>3 THEN 330
350 CLS£2:INPUT£2,"Destino",d$:d=VAL(d$)
360 IF d<1 OR d>3 THEN 350
370 IF arriba(o)<>0 THEN 400
380 CLS£2:PRINT£2,"MOVIMIENTO ILEGAL";:INPUT£2," ",B$
390 GOTO 330
400 IF arriba(d)<arriba(o) AND arriba(d)<>0 THEN 380
410 VECES=VECES+1
420 torre(o,arriba(o))=0:arriba(d)=arriba(o):torre(d,arriba(o))=1
425 REM -------------- DIBUJO DE LA TORRE ORIGEN ------------------
430 CLS£o+2
440 FOR i=arriba(o) TO 10 STEP 2
```

```
450 ficha=4:IF i=6 THEN ficha=3
460 IF i=8 THEN ficha=2
470 IF i=10 THEN ficha=1
480 IF torre(o,i)=0 THEN PRINT£o+2," ":GOTO 540
490 long=LEN(cuad$(ficha)):blanco=INT(8-long/2)
500 FOR j=1 TO blanco
510 PRINT£o+2," ";
520 NEXT j
530 PRINT£o+2,cuad$(ficha)
540 NEXT
550 arriba(o)=0
560 FOR i=arriba(o) TO 10 STEP 2
570 IF torre(o,i)<>0 AND NOT w THEN arriba(o)=i:GOTO 590
580 NEXT
585 REM ------------ DIBUJO DE LA TORRE DESTINO ----------------
590 CLS£d+2
600 FOR i=4 TO 10 STEP 2
610 IF torre(d,i)=0 THEN PRINT£d+2," ":GOTO 700
620 ficha=4:IF i=6 THEN ficha=3
630 IF i=8 THEN ficha=2
640 IF i=10 THEN ficha=1
650 long=LEN(cuad$(ficha)):blanco=INT(8-long/2)
660 FOR j=1 TO blanco
670 PRINT£d+2," ";
680 NEXT
690 PRINT£d+2,cuad$(ficha)
700 NEXT
705 REM ---------------- COMPROBACION DE FINAL DE JUEGO ----------
710 FOR i=4 TO 10
720 IF torre(3,i)=1 THEN cuenta=cuenta+1
730 NEXT
740 IF cuenta=4 THEN 770
750 cuenta=0
760 GOTO 330
770 FOR i=1 TO 150
780 SOUND 1,i,2
790 NEXT
800 CLS
805 REM ---------- CALIFICACIONES ----------------------------
810 res$="HAY QUE ENTRENARSE BASTANTE MAS"
820 IF veces>=18 AND veces<=20 THEN res$="HAY QUE PRACTICAR UN POCO"
830 IF veces=17 THEN res$="ESTA BIEN"
840 IF veces=16 THEN res$="MUY BIEN JUGADO"
850 IF veces=15 THEN res$="SE NOTA QUE TE ENTRENAS"
860 IF veces<15 THEN res$="HA SIDO INCREIBLE"
870 PRINT "LO HAS LOGRADO EN ";VECES;" INTENTOS.";RES$
880 INPUT "Quieres jugar otra vez ";y$
890 contest$=LEFT$(y$,1)
900 IF contest$="S" OR contest$="s" THEN RUN
910 FOR i=150 TO 1 STEP -1
920 SOUND 1,i,2
930 NEXT:PRINT "TU TE LO PIERDES":END
```

PROGRAMA comentado en
BASIC

DESCRIPCION Y FUNCIONAMIENTO DEL PROGRAMA

TRAGAPERRAS

1. DESCRIPCION

El programa simula el funcionamiento de una máquina tragaperras con la ventaja (o inconveniente) de que el dinero utilizado es ficticio y por lo tanto no es posible perder dinero (tampoco ganar).

Lo primero que se realiza internamente al ejecutar dicho programa es que se leen de una sentencia DATA todas aquellas combinaciones que puedan resultar ganadoras y los correspondientes premios que por ellos se dan.

Posteriormente, se declaran un número de ventanas (6 en concreto) las cuales pasan a describirse a continuación:

* La ventana número 1 (WINDOW£1, 20, 60, 8, 18) es la encargada de dibujar la máquina (Colorearla de oscuro).
* La segunda ventana (WINDOW£2, 24, 32, 10, 14) tiene la función de dibujar un cuadrado donde se presentará la primera combinación resultante.
* La siguiente (WINDOW£3, 36, 44, 10, 14) tiene la misma finalidad anterior sólo que presenta la segunda combinación obtenida.
* La cuarta ventana (WINDOW£4, 48, 56, 10, 14) muestra la última de las combinaciones u órdenes que se den a lo largo del programa.
* La última ventana que se presenta (WINDOW£6, 21, 59, 16, 18) es utilizada para presen-

tar el conjunto de posibles combinaciones ganadoras con sus correspondientes premios.

La nota común de todas estas ventanas de texto es el formato que poseen: El primer número corresponde al número de ventana. Los dos siguientes, marcan el rango expresado en columnas (inicial y final respectivamente). Por último, los otros dos números delimitan la longitud de la ventana en forma vertical (líneas).

La asignación de colores a cada ventana se realiza con motivo de hacer de estas algo más vistoso. Así, la primera ventana fondo de color MAGENTA; la tinta con que se escribirá será blanca (PEN£1, 13). Los efectos de estos cambios se hacen patentes al ejecutarse la sentencia de borrado de esa ventana (CLS£1). Las ventanas 2 a 4 tendrán fondo blanco (PAPER£1, 13) y tinta negra.

Las líneas 140-160 nos muestran (en la ventana 6) las combinaciones que pueden resultar ganadoras y el premio obtenido en cada caso.

La línea 170 limpia la ventana número 5 (la de los comentarios), para solicitar una cantidad y jugarla. La línea siguiente comprueba que la cantidad introducida con anterioridad, sea divisible por 25 (precio de cada partida). Caso de que no fuere divisible, se repetiría el proceso, solicitando una nueva cantidad para continuar el juego.

En 150 se muestra el dinero a nuestro favor que hay para jugar, a la vez que genera a un pequeño pitido indicando que hemos pulsado la tecla [ENTER] y comienza el juego.

La línea 212 es una variable que va a generar un número aleatorio comprendido entre 1 y 4. Este será el número de vueltas que dé el bucle de la línea 215. Esto es, el número de veces que se van a calcular combinaciones.

El bucle que comienza en la línea 220 es el que marca el número de combinaciones que han de aparecer. Estas se almacenan en la variable subindicada result (i) y será un número al azar comprendido entre 1 y 4.

El sentido de la variable ci es el de tener en cada momento preciso, la columna donde han de imprimirse las combinaciones que se van generando con la fórmula result(i) = int (rnd*4) + 1 (Esto es, toma la parte entera de un número que oscila entre 0 y 3, sumándole una unidad, con lo que obtenemos el

número entre 1 y 4). Este valor que ha sido calculado se muestra, a través de la ventana correspondiente (LOCATE£I + 1, CI, 1), produciendo un pitido indicativo de que se ha calculado y presentado una combinación.

Las líneas 260-295 convierten la combinación obtenida (que se encuentra en la variable result(i) a su correspondiente numérica. Para ello, se multiplica el número más a la izda. por 100., El siguiente por 10, y el último por 1. Sumando esas cantidades (compara = compara + result (i) * prod) obtendremos la combinación en forma numérica y, de esta manera, la podremos comparar con las combinaciones (numéricas) que hay en la variable comb(i) (combinaciones) ganadoras.

De 300 a 330 se compara la combinación (en forma numérica) obtenida con todas y cada una de las posibles ganadoras: mostrándolo en su caso a través de la ventana 5 y generando un sonido de felicitación.

Pulsando la tecla ESC el juego se interrumpe esté en la fase que esté, inicializando los atributos de pantalla (tinta, fondo, etc.) al estado original (líneas 1010, 1020).

2. FUNCIONAMIENTO

Tras dibujar el esquema lo que va a ser nuestra máquina, el ordenador solicita por el teclado, la introducción de una cantidad (divisible por 25: 25, 50, 100, 150, etc.) que va a ser , precisamente, la que se va a jugar. En este momento se presentará en pantalla el mensaje PULSE [ENTER] PARA JUGAR. Respondiendo a la misma comenzará a salir por cada ventana de la máquina las combinaciones que resulten en cada partida.

Caso de obtener un premio, la máquina lo hará saber mostrando la cantidad ganada así como haciendo sonar una música.

Independientemente de obtener, o no, un premio, se solicitará de nuevo la pulsación de la tecla ENTER o la introducción de una nueva cantidad de dinero (caso de que se nos haya terminado).

El programa puede interrumpirse en cualquiera de sus fases pulsando la tecla ESC deteniéndose la ejecución del mismo y restableciendo atributos de pantalla.

VARIABLES USADAS EN EL PROGRAMA "TRAGAPERRAS"

— SUBINDICADAS:

* Gana (15): Cantidades a ganar por cada combinación.
* Comb (15): Combinaciones ganadoras.
* Result (3): Combinación que resulta en cada jugada.

— RESULTARES:

* i, j, k: Indices para recorrer las matrices.
* Cant: Cantidad disponible para jugar.
* veces: Número veces a buscar una combinación.
* ci: Columna de cada ventana para colocar la comb.
* prod: Var. auxiliar para calcular la combinación en número.
* Compara: Combinación resultante, expresada numéricamente.
* pasa: Indica si ha habido combinación ganadora.

```
1 REM TRAGAPERRAS
2 REM ********************************** Angel Larrubia
5 DIM gana(15),comb(15)
10 ON BREAK GOSUB 1010
15 FOR i=1 TO 15:READ gana(i):NEXT
17 FOR i=1 TO 15:READ comb(i):NEXT
20 MODE 2
```

PROGRAMA comentado en

BASIC

```
25 REM ---------- DECLARACION DE LAS VENTANAS ---------
30 WINDOW£1,20,60,8,18
40 WINDOW£2,24,32,10,14
50 WINDOW£3,36,44,10,14
60 WINDOW£4,48,56,10,14
70 WINDOW£5,1,80,22,24
80 WINDOW£6,21,59,16,18
85 REM ---------- ASIGNACION DE COLORES ----------
90 BORDER 3:PAPER£0,3:PEN£0,0:CLS£0
100 PAPER£1,8:PEN£1,13:CLS£1
110 FOR i=2 TO 4
120 PAPER£i,13:PEN£i,0:CLS£i
130 NEXT
135 REM ---------- COMBINACIONES GANADORAS ----------
140 PRINT£6,"   444=40,111=36,222=30,333=26,443=20"
150 PRINT£6,"   211=18,422=16,133=14,134=12,221=10"
160 PRINT£6,"   121=08,124=06,432=04,242=02,132=01"
165 REM ---------- COMIENZO DEL JUEGO ----------
170 CLS£5:INPUT£5,"Introduzca cantidad para jugar (25 Pts. por partida) ",cant
180 IF cant MOD 25<>0 THEN 170
190 CLS£5:PRINT£5,"PULSE [ENTER] PARA JUGAR";
200 IF cant>0 THEN INPUT£5," ",b$:CLS£5:cant=cant-25:GOTO 210
205 GOTO 170
210 LOCATE 1,1:PRINT "CANTIDAD RESTANTE: ";cant:SOUND 6,2,5
211 REM ---------- CALCULO DE LAS COMBINACIONES ----------
212 veces=INT(RND*4)+1
215 FOR j=1 TO veces
220 FOR i=1 TO 3
222 IF i=1 THEN ci=28
225 IF i=2 THEN ci=40 ELSE ci=52
230 result(i)=INT(RND*4)+1:REM Calcula un valor entre 1 y 4
240 LOCATE£i+1,ci,1:PRINT£i+1,result(i):SOUND 1,12,2
245 FOR k=1 TO 500:NEXT:REM Bucle de retardo para ver comb.
250 NEXT i,j
255 REM ---------- CONVIERTE COMBINACION A UN NUMERO ----------
260 FOR i=1 TO 3
270 IF i=1 THEN prod=100
280 IF i=2 THEN prod=10
285 IF i=3 THEN prod=1
290 compara=compara+(result(i)*prod)
295 NEXT i
297 REM ---------- COMPARA COMBINACION CON GANADORAS ----------
300 FOR k=1 TO 15
310 IF compara=comb(k) THEN cant=cant+25*gana(k):pasa=1
320 IF pasa=1 THEN CLS£5:PRINT£5,"GANA ";25*gana(k);" Pts."
325 IF pasa=1 THEN pasa=0:FOR z=1 TO 500:SOUND 1,z,1,12:NEXT
330 NEXT:compara=0:GOTO 190
1010 PEN 13:PAPER 0
1020 CLS
1030 END
2000 DATA 40,36,30,26,20,18,16,14,12,10,8,6,4,2,1
2010 DATA 444,111,222,333,443,211,422,133,134,221,121,124,432,242,132
```

PROGRAMA comentado en BASIC

BINGO

1. DESCRIPCION:

En la línea 50 se dimensionan 4 matrices, cada una de las cuales va a tener el siguiente cometido:

* núm (4, 3, 5): Contiene los números de cada cartón (El primer dígito corresponde al número del jugador: el segundo al número de fila dentro de cada cartón; y el tercero, al número de columna de cada cartón). Hay que resaltar el hecho de que cada cartón esta formado por 15 números distribuidos en 3 filas de 5 columnas cada una.

* fila (4,3): Posee la cantidad de número que hay sin tapar en cada fila de cada jugador. Como cada una posee 5 números, inicialmente se da el valor de 5.A. medida que van surgiendo números, se van restando unidades a esta matriz; de manera que cuando valga cero, se ha completado una fila (cantándolo).

* tot (4): Juega el mismo papel que la anterior, pero en vez de controlar la fila, lo que hace es informar sobre el posible bingo. Por ello se inicializa a 15 (línea 140).

* número (100): Posee un 1 si el número resultante ha sido extraído. Almacenará un cero cuando dicho número, aún, no se haya cantado.

* cartón (4,100): Declarado en la línea 60, tiene por misión impedir que un número se repita en el mismo cartón. si contiene un 1, es que ya ha salido. En caso contrario, poseerá un cero.

Las líneas 70-110 especifican las ventanas que van a ser usadas: Número 1 para el primer jugador; la 2 para el jugador 2 y así sucesivamente. La número 5 es utilizada para presentar el número que sale u otros mensajes.

De la línea 120 a 170 se inicializan las tablas que van a controlar el bingo y la línea (La primera a 15 pues son esos los números que hay por carton y la segunda a 5 que coincide con los números existentes en cada línea.

Entre la 200 y la 260 se lleva a cabo la creación de los cartones: Se calcula un número aleatorio entre 1 y 99 imprimiéndose en el cartón correspondiente que indica la variable ''i''.

El número que se va cantando a lo largo del juego está en la línea 280 (variable ''res''). Si este número ya ha salido, implica que número (res) = 1 por lo que se repetiría el proceso para calcular otro. Si no hubiera salido, se almacena 1 en la posición correspondiente para que no se repita de nuevo.

Las líneas 340-440 tienen la siguiente misión:

Si cada número de cada cartón que juega el ordenador (num(i, j, k) es distinto del que se ha obtenido (res) y es distinto de cero (implica que ese número aún no se ha tapado) se imprime tal como está (línea 430).

Si coincide el número del cartón con el obtenido se resta al contador de números por cartón y por línea una unidad.

En este último caso, se iguala num (i, j, k) a cero para indicar que ese número ya ha sido tapado y se imprime, en su lugar, un cuadrado oscuro (v$. Correspondiente al valor 207 del código ASCII).

Una vez actualizadas las tablas de bingo y línea pasa a comprobarse si son cero para cantar lo que corresponda (líneas 400 y 410).

A partir de la línea 460 y hasta la 570 viene reflejado el turno del jugador número 4 (que somos nosotros y corresponde al cartón de más abajo).

PROGRAMA comentado en BASIC

Las posibilidades que este tiene son 3:

* Pulsar ''s'': El ordenador buscará en nuestro cartón, el número que ha salido. Si lo encuentra lo tapa (línea 520).

* Pulsar ''l'': En el momento de ir a cantar línea (línea 540).

* Pulsar '' b'': Cuando vayamos a cantar bingo (línea 530).

La 550 imprime el cuadrado oscuro al encontrar el número.

La 560 imprime el número que no corresponde con el obtenido, tal y como está.

Las líneas 600-620 son la subrutina de ''línea''. Su objetivo el de informarnos a través de la ventana 1 quien ha sido el afortunado que ha cantado línea e impedir que otro jugador vuelva a cantarla (línea = 1).

De 640-68* es la subrutina de Bingo y lo que se consigue en las líneas 810-final es hacer sonar una música de felicitación.

2. FUNCIONAMIENTO

El juego consiste en simular el juego del bingo tratando de cantar línea y bingo.

El ordenador controla los tres primeros cartones mientras que nosotros hemos de rellenar el cuarto (situado en la parte más baja de la pantalla). Para ello disponemos de tres controles:

* ''s'' para tapar el número de nuestro cartón y que coincide con el presentado en la esquina superior izquierda.

* ''l'' se ha de pulsar cuando aparece (en la esquina superior izquierda) el número que nos queda para rellenar una fila siempre y cuando NADIE haya cantado ''línea'' hasta entonces. No hace falta pulsar, también, la ''s''.

* ''b'' se usa de forma parecida a ''l'' sólo que para el bingo. Tampoco hace falta pulsar la ''s''.

Nota: Modificando la línea 320 podemos adaptar la velocidad con que salen los números a nuestro gusto. Si se quiere que vaya más rápido habrá que suprimirla, mientras que si lo que pretendemos es que sea más lento, habrá que poner 200, 300... lo que se quiera en vez de 50 (Bucle ''FOR'').

```
10 REM ************************************************************
20 REM ******** PROGRAMA DE 'BINGO': ANGEL   LARUMBE  **********
30 REM ************************************************************
40 GOSUB 780
50 MODE 2:DIM num(4,3,5),fila(4,3),tot(4),numero(100):v$=CHR$(207)
60 DIM carton(4,100)
70 WINDOW#1,49,70,10,17
80 WINDOW#2,29,50,2,9
90 WINDOW#3,9,30,10,17
100 WINDOW#4,29,50,18,24
110 WINDOW#5,1,80,1,1
120 REM ------ INCIALIZACION DE TABLAS PARA CONTROL DE LINEA Y BINGO ------
130 FOR i=1 TO 4
140 tot(i)=15
150 FOR j=1 TO 3
160 fila(i,j)=5
170 NEXT j,i
180 FOR i=1 TO 4:PAPER#i,3:PEN#i,0:CLS#i:NEXT i
190 REM ------------ REPARTO DE CARTONES -------------------------
200 FOR i=1 TO 4
210 PRINT#i," "
220 FOR j=1 TO 3
230 FOR k=1 TO 5
240 num(i,j,k)=INT(RND*99)+1:IF carton(i,num(i,j,k))=1 THEN 240
250 PRINT#i," ";:PRINT#i,USING "###";num(i,j,k);:carton(i,num(i,j,k))=1
260 NEXT k:PRINT#i," ":PRINT#i," ":NEXT j,i
270 REM ---------------- ELECCION DE UN NUMERO ----------------------
280 res=INT(RND*99)+1:CLS#5
290 IF numero(res)<>1 THEN PRINT#5,"Numero: ";res:SOUND 1,10,15
300 IF numero(res)=1 THEN 280
310 numero(res)=1
320 FOR z=1 TO 50:NEXT
330 REM --------- CONTROL DE LOS CARTONES QUE JUEGA EL ORDENADOR ----------
```

```
340 FOR i=1 TO 3
350 CLS#i:PRINT#i," "
360 FOR j=1 TO 3
370 FOR k=1 TO 5
380 IF num(i,j,k)<>res AND num(i,j,k)<>0 THEN 430
390 IF num(i,j,k)=res THEN fila(i,j)=fila(i,j)-1:tot(i)=tot(i)-1
400 IF tot(i)=0 AND num(i,j,k)=res THEN 640
410 IF fila(i,j)=0 AND num(i,j,k)=res AND linea=0 THEN GOSUB 600
420 PRINT#i," ";v$;v$;v$;:num(i,j,k)=0:GOTO 440
430 PRINT#i," ";:PRINT#i,USING "###";num(i,j,k);
440 NEXT k:PRINT#i," ":PRINT#i," ":NEXT j,i:b$=INKEY$:b$=UPPER$(b$)
450 REM -------------- TURNO DEL USUARIO (JUGADOR NUM. 4) ---------------
460 IF b$="L" OR b$="B" THEN 480
470 IF b$="" OR (b$<>"S") THEN 280
480 CLS#4:PRINT#4," "
490 FOR j=1 TO 3
500 FOR k=1 TO 5
510 IF num(4,j,k)<>res AND num(i,j,k)<>0 THEN 560
520 IF num(i,j,k)=res THEN fila(i,j)=fila(i,j)-1:tot(i)=tot(i)-1
530 IF tot(i)=0 AND num(i,j,k)=res AND b$="B" THEN 640
540 IF fila(i,j)=0 AND num(i,j,k)=res AND linea=0 AND b$="L" THEN GOSUB 600
550 PRINT#4," ";v$;v$;v$;:num(i,j,k)=0:GOTO 570
560 PRINT#4," ";:PRINT#i,USING "###";num(4,j,k);
570 NEXT k:PRINT#4," ":PRINT#4," ":NEXT j
580 GOTO 280
590 REM --------------- SUBRUTINA INDICATIVA DE 'LINEA' ---------------
600 CLS#5:PRINT#5,"JUGADOR NUMERO: ";i;" HA CANTADO LINEA":SOUND 5,15,15
610 FOR z=1 TO 1000:NEXT:linea=1:IF i=4 THEN cant=cant*2
620 RETURN
630 REM --------------- SUBRUTINA DE 'BINGO' --------------------------
640 CLS#5:PRINT#5,"EL JUGADOR NUMERO: ";i;"HA CANTADO  BINGO "
650 GOSUB 810
660 SOUND 10,10,10:FOR z=1 TO 2000:NEXT
670 IF i=4 THEN cant=cant*10
680 cant=cant-100:IF cant<=0 THEN RUN 40
690 REM -------------- PANTALLA FINAL ----------------------------
700 CLS:PRINT "AUN QUEDAN ";cant;" PESETAS"
710 INPUT "Quieres seguir jugando ";y$:CLS
720 IF y$<>"S" AND y$<>"s" THEN 760
730 FOR i=1 TO 100:numero(i)=0:NEXT
740 linea=0
750 GOTO 130
760 END
770 REM -------------- PANTALLA INICIAL ----------------------------
780 CLS:INPUT "Cantidad a jugar (100 Pts. carton)",c$
790 cant=VAL(c$):IF cant=0 OR cant MOD 100<>0 THEN 780
800 RETURN
810 READ nota,dur:IF nota=0 THEN RETURN
820 SOUND 1,nota/2,dur,7:SOUND 1,0,2
830 SOUND 4,nota*2,dur,6:SOUND 4,0,2
840 GOTO 810
850 DATA 322,30,287,30,256,30,242,60,322,90,242,30,256,30,242,30,215,60,287,90,
287,30,256,30,242,30,192,45,215,12,215,30,242,27,242,
30,256,30,287,30,256,30,242,120,0,0
```

PROGRAMA comentado en BASIC

SOLITARIO DE LOS ESPACIOS

1. DESCRIPCION

El ejecutar el programa, aparecen en pantalla todas las cartas de una baraja francesa excepto los ases. La disposición de las mismas es de 4 filas con 13 posiciones por cada fila. Cada carta está dibujada creando una ventana (número 2) que según la carta va ser de unas dimensiones u otras. Posteriormente se hará incapié en ello. Ahora, sólo queda decir (y es muy importante para entender lo siguiente) que cada carta esta formada por 5 caracteres en vertical por 4 en horizontal:

***** (Pretende ser una carta ex-
***** presada en sus correspon-
***** dientes longitudes).
*****

La ventana número 4 la usamos para poner en la primera fila el número de columna que ocupa cada carta.

La línea 40 es usada para representar los tréboles que no se están incluidos en el juego de caracteres ASCII. Consite en dibujar (lo mejor que se pueda) un trébol en un cuadrado de 8 × 8. Cada cuadrado relleno se sustiurá por dos elevado a n.º cuadrado-1. Se realiza esta operación en cada fila (sumándose cada potencia en caso de tener más de un cuadrado relleno por fila) y el resultado se pone en la intrucción 'SYMBOL'. El primer dígito va a ser el código en ASCII que va a representar. Nótese que tras este primer dígito, hay 8 más (correspondientes a cada fila del cuadrado).

En la línea 50 reservamos espacio para las siguientes tablas:

* posic$ (4,13): guarda el valor que hay en cada posición. El primero representa la fila; el segundo, la columna. Este array va a contener algo de la forma: ''4, 1'', ''13, 4'', ... Esto indica (en el primer caso) que la carta es un 4 y el palo es 1 (El 1 es carrós; el 2 picas, el 3 corazones y el 4 tréboles).

* da (13,4): almacena 1 cuando la carta obtenida ya se ha dado. O en caso se dibujará.

La primera carta comienza en la línea 2 (filai = 2: o sea, fila inicial) para terminar 3 líneas más abajo (fila f = fila i + 3; esto es, fila final. Nótese que ocupa las líneas 2, 3, 4, 5 o sea 5 líneas en total). La impresión en ancho comienza en la

columna 2 (coli = 2, o sea columna inicial) y termina 4 más allá (col f = col i + 4: columna final. Nótese lo mismo que anteriormente se ha dicho). Con este resultado se invierten los colores de fondo y tinta (línea 140). Este método de impresión de cartas se va a utilizar a lo largo de todo este programa.

Como las reglas del juego exigen no colocar los ases, en la línea 190 obligamos un salto para dejar esa posición en vacío.

Lo que ocurre en las líneas 200-280 es lo siguiente:

— Si v$ (palo de la carta) es 1 se imprime el caracter 227 del código ASCII que es un rombo (carros).

— Si vale 2, se imprime el 299 (picas).

— Valiendo 3 se imprime un corazón (caracter 228).

— Cualquier otro valor que sea (4 evidentemente) para imprimir el carácter 250 que hemos creado (trébol).

Al finalizar el juego calcula las cartas colocadas correctamente sabiendo que la carta colocada en cada posición ha de ser una unidad mayor que su columna correspondiente (910) de manera que en la columna 1 h a de haber un 2 (una unidad más); de esta manera, en la columna 12 debe haber una carta con valor 13 (rey) y en la última columna (la 13) debe haber un blanco (el correspondiente a los ases suprimidos en el comienzo).

2. FUNCIONAMIENTO

El objeto del juego es conseguir 4 hileras de 12 cartas; del 2 al rey.

Se reparten todas las cartas y se retiran los ases. Los espacios dejados por los ases se rellenan con una carta más alta que la anterior teniendo muy presente que de haber más de un espacio junto, habrá de ponerse la carta elegida en el que este más a la izquierda. De manera que si un espacio sigue al 4 de picas sólo puede ocuparse con un 5. Un espacio a la derecha de un rey no puede ser ocupado y en la primera columna de la izquierda sólo puede colocarse un 2.

Las casrtas que se consideran mal colocadas son aquéllas que no están en sus columnas respectivas (el dos en la primera, 3 en la segunda...).

Los movimientos de cartas se realizan de acuerdo con las coordenadas de cada una (indicando fila y columna). La primera fila es la que está más alta en la pantalla; la segunda la siguiente más abajo; así hasta la cuarta fila que es la inferior. Las columnas son las marcadas en la primera línea de la pantalla.

La carta ''origen'' se refiere a aquélla que es seleccionada para trasladar a otra posición (este término es equivalente en ''fila'' y ''columna'').

La carta ''destino'', ''fila destino'' o ''columna destino'' son las coordenadas de la posición libre que se va a ocupar con la ''carta origen''.

— Si ''carta'' que contiene el valor de cada carta vale 11, se imprime una ''J''.

— En caso de valer 12 saldría una Q.

— En caso del 13, pondría una K. En cualquier otro caso (que la variable este comprendida entre 2 y 10) imprimirá su propio valor.

La línea 300 hace moverse la ventana a la siguiente carta. Es decir, actualiza los valores de columnas (inicial y final) y filas (inicial y final) para la siguiente carta. En la línea 300 mueve la columna inicial 6 posiciones a la derecha (contando un blanco entre carta y carta). Cuando se imprimen 13 columnas (salida del bucle en la línea 310) se inicializa las columnas iniciales y finales (coli y colf) puesto que se pasa a la línea siguiente y, sin embargo, se suma 5 unidades a la fila que teníamos antes (fila i = fila i + 5). Entre fila y fila, también existe un espacio en blanco.

En las líneas 340-370 se escoge una carta especificando sus coordenadas de fila y columna comprobándose que se trata de cartas existentes (la columna no excede de 13...)

Entre 380 y 440 se localizan (por el procedimiento anterior) las coordenadas de la carta que se quiere mover. Se invierten las tintas para marcar dicha carta y solicita (560-590) el lugar donde se va a colocar.

A continuación (600-670) se comprueba la validez del movimiento según las reglas establecida y que se darán más ampliamente en el punto 2:

* ''cartad'' contendrá el valor del destino. Este sólo puede ser un o (valor de los ases que quitamos al principio. En caso contrario indicaría que estamos tratando de colocar una carta sobre otra.

* En la línea 620 se comprueba que si la posición destino es la primera columna, la carta que se va a colocar en dicha posición sea un 2. Cualquier otra carta será rechazada.

* El objeto de la línea 630 es el de reconocer la carta anterior a la que ponemos. Esta carta ha de ser, obligatoriamente, inferior a la que ponemos. La variable ''cartaa'' es la que posee el valor de dicha carta. Si la diferencia entre la carta anterior y la que ponemos es distinto de 1, es rechazada (Tras un 8 sólo puede ir un 9...); esto corresponde a las líneas 650 y 655.

Si ''w'' vale 1, es indicativo de que la carta que hemos pretendido colocar es errónea por lo que volvemos a imprimirla en su posición original (línea 770).

Si ''w'' vale O implica la posibilidad de colocar esa carta en la posición escogida. Para ello calcula las coordenadas (680-740) es decir, se pone el valor de la posición inicial en la final y un 1 en la original (todo ello apoyado en una variable auxiliar).

El final de partida (805-850) viene dado cuando todos los espacios en blanco están tras los reyes (líneas 840 y 845). ''car'' contiene la carta anterior a la columna que apunta ''j'' y ''carp'' el valor de la carta de la columna.

PROGRAMA comentado en

BASIC

```
1 REM ********************************************************************
2 REM ************** SOLITARIO  DE  LOS  ESPACIOS **********************
3 REM ********          ------------------------              ************
4 REM ********           ANGEL  LARUMBE  DORAL                ************
5 REM ********************************************************************
10 RANDOMIZE TIME
20 WINDOW#4,1,80,1,1
30 WINDOW#5,1,1,1,24
40 SYMBOL 250,126,60,153,255,255,219,153,24
50 MODE 2:DIM posic$(4,13),da(13,4):LOCATE 30,13:PRINT "COLOCANDO  CARTAS"
55 REM ------------- REPARTO DE LAS CARTAS EN SUS POSICIONES ---------------
60 FOR i=1 TO 13
70 FOR j=1 TO 4
80 c=INT(RND*13)+1:p=INT(RND*4)+1:IF da(c,p)=1 THEN 80
90 REM posic$ almacena las cartas colocadas en esas posiciones.
100 posic$(j,i)=STR$(c)+","+STR$(p):da(c,p)=1:NEXT j,i
110 coli=2:colf=coli+4:filai=2:filaf=filai+3:CLS
120 WINDOW#1,1,80,22,23
130 WINDOW#2,coli,colf,filai,filaf
140 PAPER#2,3:PEN#2,0
150 FOR i=1 TO 4
155 REM ------- IMPRESION POR PANTALLA DE LAS CARTAS RESULTANTES ------
160 FOR j=1 TO 13
170 LOCATE#4,coli+1,1:FRINT#4,USING "##";j
180 WINDOW#2,coli,colf,filai,filaf
190 IF VAL(LEFT$(posic$(i,j),3))=1 THEN 300
195 GOSUB 200:GOTO 290
197 REM -------- SUBRUTINA PARA IMPRIMIR 'J' POR 11,ETC... ----------
200 CLS#2:v$=RIGHT$(posic$(i,j),1)
210 IF VAL(v$)=1 THEN palo$=CHR$(227):GOTO 240
220 IF VAL(v$)=2 THEN palo$=CHR$(229):GOTO 240
230 IF VAL(v$)=3 THEN palo$=CHR$(228) ELSE palo$=CHR$(250)
240 carta=VAL(LEFT$(posic$(i,j),3))
250 IF carta=11 THEN pon$="J":GOTO 280
260 IF carta=12 THEN pon$="Q":GOTO 280
270 IF carta=13 THEN pon$="K" ELSE pon$=STR$(carta)
280 PRINT#2,pon$;" ";palo$:RETURN
290 IF j=13 THEN 310
300 coli=coli+6:colf=coli+4
310 NEXT j
320 coli=2:colf=coli+4:filai=filai+5:filaf=filai+3
330 NEXT i
335 REM -------- ELECCION DE LA CARTA A CAMBIAR DE SITIO ----------
340 CLS#1:INPUT#1,"Fila de origen ",f$:f=VAL(f$)
350 IF f<1 OR f>4 THEN 340
360 INPUT#1,"Columna de Origen ",c$:coli=2:colf=6:filai=2:filaf=5
370 c=VAL(c$):IF c<1 OR c>13 THEN 360
375 REM ------ BUSQUEDA DE LA POSICION DE LA CARTA ELEGIDA -------------
380 FOR i=1 TO f
390 FOR j=1 TO 13
400 IF i=f AND j=c THEN 450
410 coli=coli+6:colf=coli+4
420 NEXT j
430 coli=2:colf=coli+4:filai=filai+5:filaf=filai+3
440 NEXT i
450 WINDOW#2,coli,colf,filai,filaf
460 PAPER#2,2:PEN#2,11
470 carta=VAL(LEFT$(posic$(f,c),3)):IF carta=1 THEN PRINT CHR$(7);:GOTO 340
```

PROGRAMA comentado en

BASIC

```
472 REM ------ MARCA LA CARTA ELEGIDA CON FONDO OSCURO ---------
475 GOSUB 200
500 REM --------- SOLICITA EL DESTINO DE LA CARTA ELEGIDA ----------
560 CLS#1:INPUT#1,"Fila destino ",fd$:fd=VAL(fd$)
570 IF fd<1 OR fd>4 THEN 560
580 INPUT#1,"Columna destino ",cd$:cd=VAL(cd$)
590 IF cd<1 OR cd>13 THEN 580
595 REM -------- COMPRUEBA QUE SE PUEDA COLOCAR ---------------
600 cartad=VAL(LEFT$(posic$(fd,cd),3))
610 IF cartad<>1 THEN PRINT CHR$(7);:w=1:fd=f:cd=c:GOTO 660
620 IF cd=1 AND carta<>2 THEN w=1:PRINT CHR$(7);:fd=f:cd=c:GOTO 660
630 x=cd-1:IF x<1 THEN x=1
640 cartaa=VAL(LEFT$(posic$(fd,x),3))
650 IF (carta-cartaa<>1) THEN fd=f:cd=c:w=1:PRINT CHR$(7);
655 IF (carta-cartaa<>1) AND (carta<>2) THEN PRINT CHR$(7);:fd=f:cd=c:w=1
660 coli=2:colf=6:filai=2:filaf=5
670 IF cartaa=1 AND carta<>2 THEN w=1
675 REM ------- SI W=0 PUEDE COLOCARSE:PONE LA CARTA EN COORD. CALC. ------
680 CLS#2:FOR i=1 TO fd
690 FOR j=1 TO 13
700 IF i=fd AND j=cd THEN 750
710 coli=coli+6:colf=coli+4
720 NEXT j
730 coli=2:colf=coli+4:filai=filai+5:filaf=filai+3
740 NEXT i
750 WINDOW#2,coli,colf,filai,filaf
760 PAPER#2,3:PEN#2,0:CLS#2:PRINT#2,pon$;" ";palo$
765 REM ------- SI W=1 NO PUEDE PONERSE:VUELVE A DIBUJAR LA ELEGIDA -------
770 IF w=1 THEN w=0:GOTO 340
775 REM ---- PUEDE COLOCARSE:INTERCAMBIA VALORES DE LAS POS. -------
780 aux$=posic$(fd,cd)
790 posic$(fd,cd)=posic$(f,c)
800 posic$(f,c)=aux$
805 REM --------- COMPRUEBA SI ES FINAL DE PARTIDA --------------
810 FOR i=1 TO 4
820 FOR j=1 TO 13
830 car=VAL(LEFT$(posic$(i,j-1),3)):carp=VAL(LEFT$(posic$(i,j),3))
840 IF (car=13) AND (carp=1) THEN conta=conta+1
845 IF (car=1) AND (carp=1) THEN conta=conta+1
850 NEXT j,i:IF conta=4 THEN 870
860 conta=0:GOTO 340
865 REM ------------------- ES  EL  FINAL ----------------------
870 SOUND 1,10,15
875 CLS:PRINT "HA TERMINADO EL JUEGO.VAMOS A VER COMO ESTAN LAS CARTAS"
877 REM ------------ CALCULA CARTAS COLOCADAS CORRECTAMENTE ----------
880 FOR i=1 TO 4
890 FOR j=1 TO 13
900 carta=VAL(LEFT$(posic$(i,j),3))
910 IF carta=j+1 THEN contador=contador+1
920 NEXT j,i
930 FOR i=1 TO 10000:NEXT:CLS
940 LOCATE 20,12:PRINT "HA COLOCADO CORRECTAMENTE ";contador+4;" CARTAS"
950 FOR i=1 TO 15000:NEXT
955 REM ---------------- PARA VOLVER A JUGAR --------------------
960 INPUT "Quieres volver a jugar ";y$:y$=LOWER$(y$)
970 IF LEFT$(y$,1)="s" THEN RUN
980 END
```

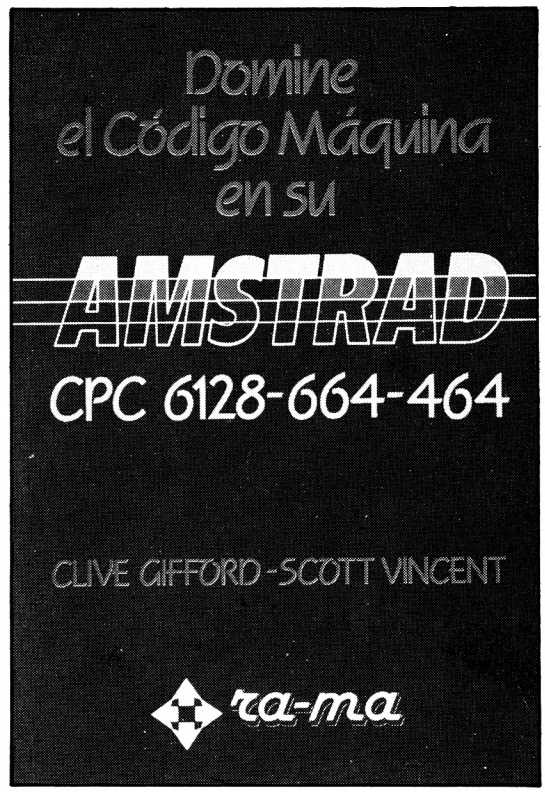
BIBLIOGRAFIA

EL DOMINIO DEL AMSTRAD PCW 8256/8512

"El dominio del Amstrad PCW 8256/8512" le sitúa en la vía rápida hacia el éxito en los negocios. Lo mismo si está Vd. empezando a utilizar uno de estos ordenadores, como si está considerando comprárselo, este libro le mostrará rápidamente cómo rentabilizar al máximo su inversión.

Una gran parte del libro trata del proceso de texto.

Esta parte del libro es un medio didáctico excelente, que le conducirá desde los usos más sencillos del proceso de textos, a los más avanzados. Una sección aparte cubre el uso de NewWord, la potente alternativa de NewStarSoftware.



DOMINE EL CODIGO MAQUINA EN SU AMSTRAD

Ahora tiene la oportunidad de aprender a programar en código máquina en su ordenador Amstrad. Clive y Scott —dos programadores muy competentes, con gran experiencia en libros y software a sus espaldas— son los guías ideales para ayudarle a comprender las interioridades de la programación en código máquina del Amstrad.

Debe ir trabajando en código máquina del Amstrad.

Debe ir trabajando a lo largo del libro, saltándose las secciones que le presentan una especial dificultad la primera vez que las lea. Cuando haya terminado su primera lectura, tendrá los suficientes conocimientos como para poder comprender aquellas secciones que dejó sin completar la primera vez que pasó por ellas.

AMSTRAD
CPC 464, 664 y 6128
PROGRAMACION ESTRUCTURADA

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

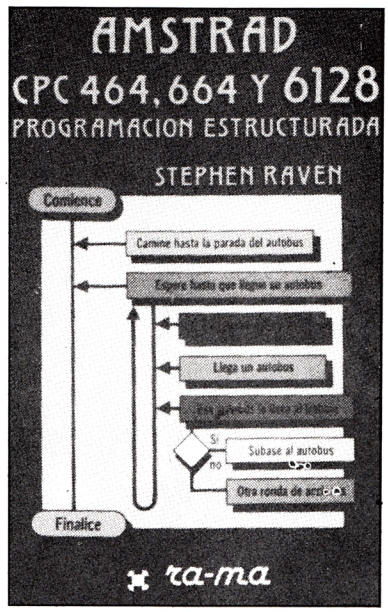
La buena nueva es que los programas escritos en BASIC para el 464 y el 664 funcionarán sin alteración en el 6128. Eso significa que debieran «currar» todos los programas basados en el disco 664 y el 99,5% de los programas en cinca 464. **Eso significa también que puedes aprovechar todas las pistas y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

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

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

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

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



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

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

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

P.V.P. 1.700 ptas.

Boletín de suscripción

A remitir a GTS. S.A. C/Bailén, 20. 1.º Izqda. 28005 Madrid.

Deseo suscribirme a los 11 números anuales de Amstrad Educativo por sólo 2.500 ptas. a partir del próximo número.

El importe lo haré efectivo:

- ☐ **Por giro postal n.º** .....
- ☐ **Por talón nominativo adjunto.**
- ☐ **Contra reembolso a la recepción del primer ejemplar, más gastos de envío.**

Nombre y apellidos: .....

Domicilio: .....

Ciudad: .....Teléfono .....

Fecha: .....Firma .....

¡Prográmate!

Envía hoy mismo tu boletín de

SUS CRIP CION

AMSTRAD EDUCATIVO

SELLO

Bailén, 20 - 1.º Izq.

28005 - MADRID

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

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 _____

_____ Nº _____ 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