AÑO II N.º 35

MICROHOBBY

AMSTRAD Semanal

REVISTA INDEPENDIENTE PARA USUARIOS DE ORDENADORES AMSTRAD

AÑO II N.º 35

160 Ptas.
Canarias **165** pts.

VENCE A NUESTRO DOMINO Y A SUS 9 NIVELES DE JUEGO

HAZ COPIA DE SEGURIDAD DE TUS PROGRAMAS FAVORITOS

COMPRESS BAS: Menos memoria y más rapidez para tus programas

FORMATOS DE IMPRESION AL ALCANCE DE TODOS: PRINT USING

HOBBY PRESS

Ofites Informática

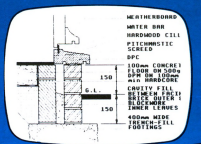
Presenta: la tableta gráfica

GRAFPAD II-

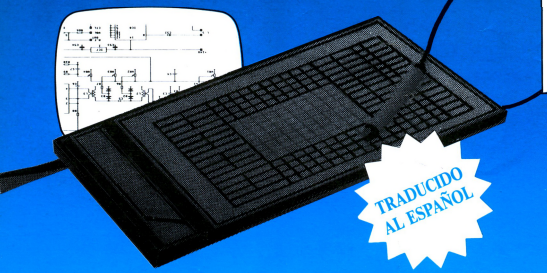
LO ULTIMO EN DISPOSITIVOS DE ENTRADA DE GRAFICOS PARA AMSTRAD, COMMODORE Y BBC

La primera tableta gráfica, de bajo costo, en ofrecer la duración y prestaciones requeridas por las aplicaciones de negocios, industria, hogar y educación. Es pequeña, exacta y segura. No necesita ajustes ni mantenimiento preventivo. GRAFPAD II es un producto único que pone la potencia de la tecnología moderna bajo el control del usuario.

DIBUJO A MANO ALZADA
SOFTWARE DE ICONOS



DISEÑO DE ARQUITECTURA
CON SOFTWARE DDX



TRADUCIDO AL ESPAÑOL

ESPECIFICACIONES

RESOLUCION:
1.280 x 1.024 pixels.
PRECISION:
1 pixel.
TASA DE SALIDA:
2.000 pares de coordenadas por segundo.
INTERFACE:
paralelo.
ORIGEN:
borde superior izquierdo o seleccionable.
DIMENSIONES:
350 x 260 x 12 mm.

DISPONIBLE AMSTRAD:
CASSETTE .... 23.900 ptas.
DISCO ..... 25.900 ptas.
(IVA NO INCLUIDO)

- FACIL DE USAR.
- TRAZADO PCB.
- C.A.D.
- AREA DE DISEÑO DIN A4.
- COLOR EN ALTA RESOLUCION.
- USO EN HOGAR Y NEGOCIOS.
- VARIEDAD DE PROGRAMAS DISPONIBLES.
- DIBUJO A MANO ALZADA.
- DIAGRAMAS DE CIRCUITOS.

COMBINA EN UN UNICO DISPOSITIVO TODAS LAS PRESTACIONES DE LOS INTENTOS PREVIOS DE MECANISMOS DE ENTRADA DE GRAFICOS. LAS APLICACIONES SON MAS NUMEROSAS QUE EN LOS DEMAS DISPOSITIVOS COMUNES E INCLUYEN:

• selección de opciones • entrada de modelos • recogida de datos • diseño lógico • diseño de circuitos • creación de imágenes • almacenamiento de imágenes • recuperación de imágenes • diseño para construcción • C.A.D. (diseño asistido por ordenador) • ilustración de textos • juegos • diseño de muestras • educación • diseño PCB.

DE VENTA EN LOS MEJORES COMERCIOS DE INFORMATICA

Si Vd. tiene alguna dificultad para obtener la tableta gráfica, puede dirigirse a:

Ofites Informática

Avda. Isabel II, 16 -8º
Tels. 455544 - 455533
Télex 36698
20011 SAN SEBASTIAN

CONDICIONES ESPECIALES PARA DISTRIBUIDORES

Director Editorial
José I. Gómez-Centurión
Director Ejecutivo
José Mª Díaz
Redactor Jefe
Juan José Martínez
Diseño gráfico
José Flores
Colaboradores
Javier Barceló
David Sopuerta
Robert Chatwin
Eduardo Ruiz
Francisco Portalo
Pedro Sudón
Miguel Sepúlveda
Francisco Martín
Jesús Alonso
Pedro S. Pérez
Amalio Gómez
Secretaria Redacción
Carmen Santamaría
Fotografía
Carlos Candel
Portada
M. Barco
Ilustradores
J. Igual, J. Pons, F. L. Frontán, J. Septien, Pejo, J. J. Mora

Edita
HOBBY PRESS, S.A.

Presidente
María Andrino
Consejero Delegado
José I. Gómez-Centurión

Jefe de Producción
Carlos Peropadre

Marketing
Marta García
Jefe de Publicidad
Concha Gutiérrez
Publicidad Barcelona
José Galán Cortés
Tel: (93) 303 10 22/313 71 62

Secretaria de Dirección
Marisa Cogorro

Suscripciones
M.ª Rosa González
M.ª del Mar Calzada

Redacción, Administración y Publicidad
La Granja, 39
Polígono Industrial de Alcobendas
Tel.: 654 32 11
Telex: 49 480 HOPR

Dto. Circulación
Paulino Blanco

Distribución
Coedis, S. A. Valencia, 245
Barcelona

Imprime
ROTEDIC, S. A. Crta. de Irún. Km. 12,450 (MADRID)
Fotocomposición
Novacomp, S.A.
Nicolás Morales, 38-40
Fotomecánica
GKOF
Ezequiel Solana, 16
Depósito Legal:
M-28468-1985

MICROHOBBY

AMSTRAD

sumario

Año II • Número 35 • 29 de Abril al 5 de Mayo de 1986
160 ptas. (incluido I.V.A.)
Canarias, 155 ptas. + 10 ptas. sobretasa aérea
Ceuta y Melilla, 155 ptas.

5 Primera plana

El nuevo especial de micromanía, a punto.

22 Serie Oro

Dominó es un programa que no necesita ni de presentación ni de comentario. Baste con decir que el ordenador juega muy bien y que los gráficos del programa son excelentes.

6 Primeros pasos

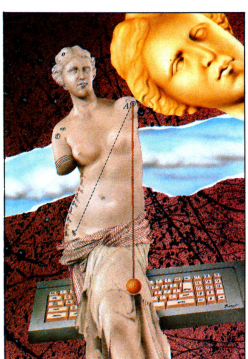
*Segunda parte del exhaustivo estudio de la sentencia **PRINT USING**. Recordemos que esta orden es absolutamente esencial para dar un formato serio y profesional a los textos que aparecen en pantalla.*

28 Utiles del programador

*Abrimos una nueva sección dedicada a los programadores, con idea de suministrarles programas o técnicas de utilidad decisivas a la hora de desarrollar software. Comenzamos con un plato fuerte. **COMPRESS.BAS** comprime un programa Basic, eliminando todo lo superfluo y haciéndole correr más deprisa en menos memoria.*

Código Máquina 18

Presentamos un programa que permite hacer copias de seguridad de cinta a cinta.



Mr. Joystick 12

*Analizamos esta semana un juego que podríamos llamar de acción, por la centelleante velocidad en la que se desarrolla: el **«Ping Pong»**.*

ProgramAcción 14

*Revisamos las últimas técnicas, o casi, que nos quedan por estudiar para así sacar el máximo partido de las facultades sonoras del **Amstrad**.*

¡ No estamos para juegos !

LO NUESTRO ES HACER BUENAS GESTIONES

Para AMSTRAD 8256 y 6128

CONTROL DE ALMACEN + I.V.A.

Sepa lo que tiene, su costo, proveedor, totales por artículos y general. Pida cuantos resultados quiera de su explotación, por producto, proveedor, etc., etc. (**15.300 ptas.** incluido I.V.A.)

CUENTAS PROVEEDORES-BANCOS-CLIENTES

Sencillo control de cuentas separadas (cuantas quiera), con lo que eliminará el problema diario de la pequeña empresa con buen movimiento (**8.600** ptas. incluido I.V.A.)

RECIBOS

Programa que resuelve el mecanografiado interminable. Asociaciones, comunidades, colegios, clubes, podrán hacer los recibos normalizados con domiciliación bancaria y posibilidad de correcciones. (**18.300** ptas. incluido I.V.A.)

CLIENTES

Datos actualizados, etiquetas correspondencia. Petición por 5 campos los resúmenes (**8.600** ptas. incluido I.V.A.)

FACTURACION

Numeración correlativa automática, fecha automática, resúmenes clientes y totalizaciones (**15.300** ptas. incluido I.V.A.)

PRESUPUESTOS

Presupuesto sencillamente cuanto quiera y transfórmelo en factura de forma sencilla. Posibilidad transformaciones y reformas. (**18.300** ptas. incluido I.V.A.)

PARA EL COMPRADOR-RETENER
Este diskette programa, viene presentado en plástico precintado, etiqueta azul, garantía y n.º de serie. Si le ofrecen otra cosa, formule reclamación.

1 AÑO DE GARANTIA

HACEMOS PROGRAMAS A MEDIDA

Encargos llamar o contactar con Juan Luis Ruiz. Tno.: 474 55 32

3 IMPACTOS 3 NOVEDADES

OFERTA
8256 6128
143.360 106.000
(Incluido I.V.A.)
Regalo de 15.300
(Programa Almacén + I.V.A.)

AMPLIACIONES MEMORIA
Por Interface 664
a 128K—15.200 464
a 320K—27.800
(Incluido I.V.A.)

PAPEL DE IMPRESORAS
- Continuo
- Impreso s/original
- Recibos
- Albaranes
- Facturas
- Cartas
- Etiquetas

AUTOCOPIANTE
A MEDIDA

PEDIDOS, TELEFONO, CARTA O TELEX
REEMBOLSO SIN GASTOS.

ESPECIAL A COLABORADORES
RESTO DE ESPAÑA

informática GROTUR, S.A.
C/ JAIME EL CONQUISTADOR, 27
28045 MADRID. Tno. 474 55 00
474 55 32
Télex: IGSA 48452

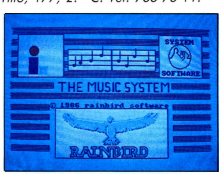
COMPONIENDO EN UN CPC

MUSIC FILES, editado por la compañía inglesa Rainbird, constituye uno de los programas más completos de música, que hoy día existen para **Amstrad.** Distribuido por **SERMA** en España, resulta agradable de usar por su cuidada estética, y presentación. Todo el programa se desarrolla a base de menús seleccionados desde el teclado de función, que se distribuyen en pantalla por medio de ventanas superpuestas.

Las posibilidades son muy variadas y van desde interpretar música ya grabada, hasta la composición más profesional, con todos los garabatos musicales que esto implica. Una vez compuesta la música, es posible modificar la velocidad de ejecución, el tono y acompañar la melodía con el acorde que deseamos.

Un buen programa que hará las delicias de los melómanos, y sobre todo de aquéllos, que como yo, no tienen ni idea de qué es esto de componer música y quisieran hacerlo.

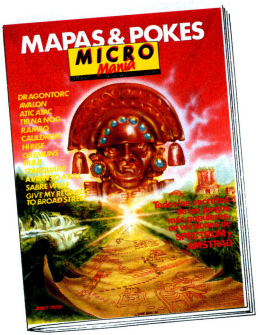
Serma se encuentra en: *Bravo Murillo, 177, 2.º C. Tel. 733 73 11.*



LA PIRAMIDE DEL FARAON

Editado por **QLS** y distribuido por esta misma compañía junto con ABC, **Faraón** puede encuadrarse dentro de los juegos de la más pura línea conversacional. La misión consiste en construir, por encargo del sumo faraón, una pirámide que albergue sus huesos durante el eterno descanso. Para esta magna empresa dispondremos de casi 5.000 esclavos y una cantidad de dinero, que tendremos que administrar sabiamente si no queremos que nuestros sirvientes mueran de hambre. A medida que transcurre la construcción, habremos de enfrentarnos con múltiples problemas, guerras, rebeliones y fiestas que harán menguar nuestro capital y aumentarán o disminuirán la moral de nuestros hombres.

Primera PLANA



En el mes de mayo

MICROMANIA
LANZA UN NUMERO
«MUY ESPECIAL»

MICROMANIA, nuestra hermana mayor que como sabéis se dedica mes a mes a analizar los mejores juegos para todos los ordenadores, va a lanzar el próximo mes de mayo un especial, al precio de 400 ptas., dedicado por entero a una de las secciones que más le caracterizan: **Pistas Arriba.**

Pokes y **mapas** a todo color de los juegos más populares, **Rambo, Starquake, Cauldron, Atic Atac, Tir Na Nog...** y así hasta un total de 13, se desarrollan a lo largo de 84 páginas con un gran despliegue de fotos e instrucciones para resolverlos más fácilmente.

PRIMERA FERIA AMSTRAD

Con enorme satisfacción hemos recibido la noticia de que Indescomp va a organizar en el transcurso de la segunda quincena del mes de mayo, la primera feria **Amstrad** en España, para poder ofrecer al público la gama de productos de esta marca.

La pretensión de Indescomp es agrupar bajo un mismo techo, aún sin especificar, a todas las compañías, nacionales y extranjeras, que de alguna manera tienen relación con nuestro ordenador. Software, Hardware y papel por y para **Amstrad** en una feria que promete ser no ya sólo la primera, sino el comienzo de una sana costumbre, que si bien arraigada en el extrajero, aún no ha hecho mella en nuestro país.

Bien por Indescomp y nuestro ánimo a todos aquéllos que trabajan con el propósito de aumentar las posibilidades de **Amstrad**. Allí nos vemos...!

UNA PLANTILLA MUY NUMERICA

Ya conocemos la forma de «fabricar» una plantilla adecuada para poder escribir una variable literal o cadena de texto con un formato determinado. Además, nuestro ordenador posee la habilidad de poder hacer esto mismo con números.

En esta ocasión vamos a intentar explicar cómo es posible definir la plantilla del formato que en cada momento necesitamos. No olvide que PRINT es la instrucción que usa el lenguaje Basic del **Amstrad** para hablarnos y que, por tanto, es necesario llegar a dominarla para **«cuidar»** al máximo nuestra comunicación.

Vamos al grano. Para escribir —*o PRINT*— un texto en la pantalla de manera que aparezca componiendo líneas de elementos que posean unas características que nosotros previamente hemos fijado, utilizamos la instrucción:

PRINT USING

Sólo nos falta añadirle, o mejor definir, la **«plantilla»** a la que ha de amoldarse la escritura. Por ejemplo ya no debe representarle ninguna dificultad hacer que el ordenador haga aparecer en la pantalla los 3 primeros caracteres de cualquier palabra.

Se puede imaginar que para ello vamos a tener que hacer uso del comando PRINT USING, pero en este caso, **¿qué plantilla tendremos que utilizar?**

Teclee:

```
PRINT USING " \  \"; «casa»
PRINT USING " \  \"; «chiripitiflàutico»
```

y si no nos hemos equivocado el ordenador escribirá las tres primeras letras —o caracteres— de cada una de estas palabras.

Pero puede preguntarse: **¿Cuál es la plantilla empleada?** Observe atentamente la cadena de caracteres especiales que hemos colocado a continuación de PRINT USING:

" \ \"

Con este formato le estamos diciendo al **Amstrad** que lo que va a escribir lo ha de hacer solamente con el número de caracteres contenidos entre las comillas: así lo determina el carácter especial " \ ". O sea que, en este caso dos barras inclinadas hacia la izquierda más un espacio en blanco = tres caracteres en total.

Y por eso es por lo que las dos instrucciones que ha escrito anteriormente, hacen que se impriman los tres primeros caracteres de cada palabra.

Y, **¿existe la posibilidad de emplear esto con los números?** Si queremos reservar cuatro espacios, por ejemplo, para escribir allí cualquier número, ¿es válida esta plantilla?

Teclee:

```
PRINT USING " \  \"; 12
```

y vea lo que ocurre une vez haya pulsado RETURN.

Le suena de algo el mensaje:

Type mismatch

Claro, es justamente el que acaba de aparecer en la pantalla. Nos está diciendo que hay un error en algún sitio.

Este mensaje de error significa que la plantilla que hemos definido, no está de acuerdo con el tipo de dato que va a formatear: espera una cadena de caracteres y nosotros le estamos dando un número.

Bueno, pues no es válida. ¡Qué desilusión!

¡Más plantillas!

Pero no se preocupe, hay muchas plantillas más. Nuestro **Amstrad** está provisto de casi todo. ¿Sigue queriendo reservar cuatro espacios para escribir un número? Escriba:

```
PRINT USING "####"; 12
```

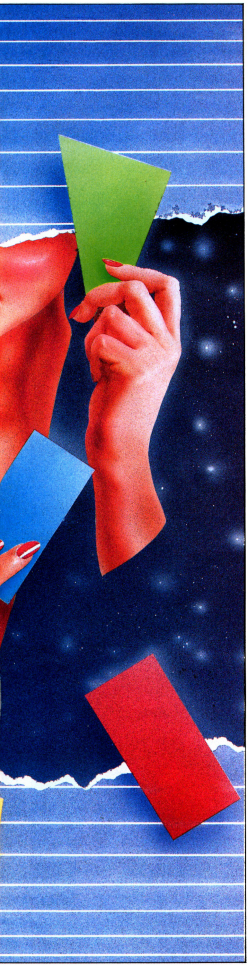
y verá cómo lo consigue.

En esta ocasión el número no va a estar «pegado» al borde de la pantalla, bueno, o casi pegado (recuerde el espacio que el ordenador guarda para el signo). Aparecerá un poco más dentro.

Si cuenta el número de espacios que hay antes del número 12, comprobará que no le hemos engañado, y que en efecto hemos destinado un número determinado de «celdas» de carácter, o huecos destinados a ser ocupados por una letra o un número, para que contengan el 12: hay dos espacios en blanco y dos cifras = cuatro caracteres como queríamos.

Luego ya hemos encontrado la plantilla que necesitábamos.

```
PRINT USING ####; 12
```



le está indicando al ordenador que escriba el número 12 en la pantalla y que además lo haga reservando para ello tantas posiciones como caracteres especiales «#» encontremos entre las comillas —cuatro en este caso.

Haga distintas pruebas cambiando la cantidad de «#». Si teclea:

PRINT USING ######; 12.

el ordenador guardará 6 espacios para escribir allí nuestro ya viejo conocido número 12. Haga todas las variaciones que se le puedan ocurrir.

Pero esta plantilla no sólo nos sirve para imprimir número formateados sino que también podremos utilizarla con variables numéricas.

Introduzca en la memoria del ordenador el Programa I y lo comprobará.

Input inteligente

Es muy sencillito pero va a servirnos perfectamente para afianzar los conceptos que tenemos sobre este formato numérico.

Simplemente está formado por una instrucción INPUT, mediante la que el micro nos va a pedir un dato numérico y lo va a guardar en la variable **«número»** —línea 30.

Fíjese en el comando PRINT USING de la 50. La plantilla usada en esta ocasión es «#####»; o sea, que hemos especificado seis posiciones paa contener el número que vayamos a escribir.

Pero ahora no vamos a imprimir un número fijo o constante, sino el valor que esté almacenado en la zona de memoria, que el ordenador ha reservado para la variable «número». Es otra de las posibilidades de PRINT USING.

La línea 60 hace que el micro dé un salto atrás a la línea 30 y nos pida un nuevo valor.

Ejecute el Programa y dele a la variable «número» todas las cantidades enteras que se le ocurran. Haga sus propias pruebas y anote las conclusiones sacadas de cada una de ellas. Es una buena forma de aprender.

Obsérvese que siempre que introduzcamos datos correctos, el **Amstrad** nos va a reservar las seis posiciones y allí va a escribir el número. Y siempre nos guarda los mismos espacios.

Intentemos otra prueba. Teclee ahora, bueno, cuando al ejecutar el Programa I se lo pida el ordenador, un número de más de seis cifras a ver qué pasa. **¿Nos dará error?**

Es evidente que si el **Amstrad** nos reserva 6 posiciones para escribir un número y nosotros, con mucha mala idea, le vamos a dar uno de 7, 8 o más cifras, no va a poder imprimirlo por la sencilla razón de que allí no le cabe. **¿Qué hará entonces?**

Eche un vistazo a la pantalla. Allí aparece el número tal como nosotros lo habíamos escrito —con el mismo número de cifras— pero ahora precedido del símbolo «%». Analicemos qué puede significar este carácter especial.

«%» es la forma que tiene el **Amstrad,** de

Primeros PASOS

decirnos que algo no funciona, que la plantilla de formato que hemos elegido no es del todo válida: ¡falla!

Y claro que falla. Hemos especificado 6 posiciones para contener un número de 7 o más cifras, luego el número es demasiado grande como para entrar el formato definido.

Pero sin embargo, el ordenador lo ha escrito con todas sus cifras, no lo ha cortado ignorando las que siguen a la sexta. Informáticamente hablando se dice que no lo ha «truncado», sino que lo escribe entero.

Ahora bien, es eficiente y nos avisa del fallo por medio del signo «%». Por todo esto decimos que la plantilla no es del todo válida: el número aparece en la pantalla tal como nosotros lo tecleamos, pero el ordenador nos avisa de que no lo hace como hubiéramos deseado. **¡A corregir toca!**

Intentemos hacer otra prueba. Ejecute el Programa I otra vez y meta como dato un número fraccionario. La variable «número» va a contener en esta ocasión un número que no es entero.

¿Qué plantilla hemos diseñado? Pues una que nos va a reservar 6 espacios en la pantalla para allí imprimir un número entero de seis cifras como máximo.

Observe ahora el resultado de esta ejecución. Aparece un número entero y las cifras decimales han sido despreciadas.

Si hemos querido escribir solamente la parte entera nos puede servir esta plantilla, pero si queremos que nos aparezcan los decimales ha llegado el momento de poner en funcionamiento nuestra mente e intentar definir otro formato que nos trate también la parte decimal del número. Y siempre llegamos a la misma pregunta: **¿Cómo podemos hacerlo?** Veamos. **¿Cuántas cifras enteras queremos que nos aparezcan?** Por ejemplo seis, como antes. Y, **¿cuántos decimales?** Creemos que con 3 nos puede valer.

Escriba ahora en un papel un número compuesto por seis cifras enteras y 3 decimales. Sería algo parecido a: 613258.346

Sustituya ahora cada cifra, tanto entera como decimal, por el símbolo que especificaba una posición para un dígito: «#». Convertiremos el número en:

#####.###

es decir, espacio para 6 cifras enteras, el punto decimal seguido de tres posiciones para otras tantas cifras decimales. Y si todo esto lo mete entre comillas:

''######.###''

obtendrá la plantilla del formato que estamos buscando.

Por tanto, el punto «.» que hemos colocado en un especificador de formato que le indica al ordenador la posición del punto decimal.

Sustituya la línea 50 por:

50 PRINT USING "######.###"; número

en el Programa I y ya nos admitirá números decimales, o no enteros y nos lo imprimirá reservando para cada número 6 posiciones para las cifras enteras y 3 para las decimales separadas por un punto.

Entreténgase ahora en teclear el Programa II y analícelo. No le resultará muy difícil.

Cifras encolumnadas

Su misión va a consistir en mostrarnos cómo quedan alineados todos los puntos decimales.

Nos va a pedir cinco números reales —que pueden ser enteros o con decimales— repitiendo la instrucción INPUT tantas veces como se ejecute el cuerpo del bucle.

Después el programa nos va a sacar en pantalla cada uno de los números con el formato que hemos definido —línea 110.

La plantilla es la misma que empleamos en el ejemplo anterior. O sea, seis enteros y el punto seguido de tres cifras decimales.

Ejecútelo y compruebe por sus propios medios que no le hemos dicho ninguna mentira cuando le indicamos que los puntos decimales salen alineados.

Pasemos a ver otro formato. En alguna ocasión puede que necesitemos separar las cifras enteras con comas en grupos de 3. Habrá visto infinidad de veces un número escrito de la forma siguiente: 341,115.123

Este método se emplea generalmente para facilitarnos la lectura y comprensión de un número sobre todo cuando es muy grande y por lo tanto, está compuesto por muchos dígitos. *(Si le extraña la puntuación, recuerde que los ordenadores son creaciones inglesas y que en este idioma la coma indica millar y el punto decimal).*

Como nuestro **Amstrad** conoce esta cuestión, nos va a proporcionar la herramienta para poder definir una plantilla que nos permita imprimir cualquier número con sus cifras agrupadas.

Para ello, empleamos el especificador de formato «,» colocado justamente delante del punto decimal. Es el único lugar donde lo podemos situar: antes del punto. Con él, como dijimos antes, especificamos al ordenador que agrupe de tres en tres los dígitos situados a la izquierda del punto decimal y los separe por comas.

Sustituya la línea 110 del Programa II por:

110 PRINT USING "######,.###"; número (contador)

y vea que cuando se ejecute el programa producirá los resultados que hemos buscado. Pruébelo.

Hay que tener en cuenta una cosa. La coma, además de decirle al ordenador que separe las cifras enteras en grupos de 3, también especifica una posición.

''####.##''

es una plantilla, que permitirá reservar 4 espacios para dígitos enteros —parte izquierda del punto— agrupados de tres en tres y separados por una coma y guardará dos huecos para otros tantos decimales. Recuerde, la «,» se considera también como una posición.

Y no son éstos todos los formatos. Puede que se nos ocurra rellenar con asteriscos los espacios libres que nos quedan a la izquierda del número. También tenemos una plantilla para conseguirlo.

Basta con añadir dos asteriscos al comienzo de la cadena que da forma a la plantilla de formato y punto.

Le vamos a hacer trabajar un poquito más. De nuevo cambie la línea 110 del primitivo Programa II por:

110 PRINT USING ''**####.##''; número *(contador)*

y ejecútelo **¿Hemos conseguido lo que queríamos?** ¡Sí, cómo no! Todos los espacios libres están ahora ocupados por asteriscos.

Como ocurre con el especificador «,», en este caso también «**» nos está definiendo la posición —o mejor las dos posiciones— reservadas a otros tantos caracteres. No eche en «saco roto» esta advertencia ya que si no lo considera escribirá sus números favoritos con un formato erróneo.

Podemos hablar ahora de dinero. La mayoría de los ordenadores están todavía pensados para la parte del mundo de habla inglesa y por tanto de momento sólo disponemos de plantillas que nos asocien cantidades a las monedas inglesas o de EE. UU. —libras o dólares.

Vamos a intentar que antes de cada número aparezca el signo correspondiente a una de estas dos monedas. Volvamos de nuevo a nuestro superutilizado Programa II.

Por «enésima» vez sustituya la línea 110 por:

110 PRINT USING ''######.###''; número *(contador)*

o por:

110 PRINT USING ''$$####.##''; número *(contador)*

y ejecute el Programa. **¿Qué ocurre?**

Según la sustitución que hagamos, aparecerán en la pantalla cada uno de los números precedidos por el signo de la moneda que hayamos elegido.

NOTA: Tanto $$ como ## también definen dos posiciones reservadas para ser ocupadas por algún carácter. No lo olvide.

¿Por qué no volver a rellenar los espacios vacíos a la izquierda con asteriscos? Vamos a ello. Los formatos que emplearemos ahora serán:

''**####''

y

''**$###''

que actuaron como los especificadores «**» y «##» en un caso o bien como «**» y «$$» en otro pero esta vez combinados, es decir, con asteriscos a la izquierda del número y además con los signos # o $. Es lo que queríamos, ¿no?

Un último cambio

Si le insinuamos que cambie una línea por..., en el Programa II, inmediatamente habrá pensado en la 110, !otra vez!

Teclee la siguiente variación:

110 PRINT USING ''**$###''; número *(contador)*

o

110 PRINT USING ''**####''; número *(contador)*

Haga correr de nuevo el programa y le mostrará en la pantalla los efectos de este nuevo formato definido. ¡Algún día podremos hacer lo mismo con nuestra moneda!

Y otra vez creemos necesario decir que «**$» y «**#» también nos van a especificar tres huecos donde tenemos la posibilidad de escribir un carácter.

¿Nos metemos ahora con los signos? Sobre todo en algún problema matemático o de contabilidad podemos necesitar que nos aparezcan a la izquierda o a la derecha de una cantidad. Pues bien, incluyá*moslos en el formato.

''+######.###''

es una plantilla que nos va a guardar espacio para poder escribir 6 dígitos en la parte entera y 3 en la parte decimal, pero en esta ocasión va a ir precedido de su correspondiente signo ya sea el + o el —.

Si introducimos un signo + al final de la plantilla:

''######.###+''

al emplear un PRINT USING con ella ahora el signo estará colocado al final del número (independientemente que sea + o —).

¿Qué ocurre si ahora en vez de + incluimos en la plantilla el —? Confesaremos una cosa: el — sólo podremos colocarlo al final.

''######.###—''

sería el formato ahora distinto. Le dice al ordenador dos cosas:

— Si el número es positivo, que deje un espacio vacío a su derecha.

— Si es negativo, el **Amstrad** escribirá el número con su formato correspondiente y le añadirá el signo menos a la derecha.

Y, ¡cómo no! Nos faltaba proponerles que hicieran un ligero cambio en la línea 110 del Programa II. Pues, ¡ánimo, hágalo!, pero ahora incluya estas dos últimas plantillas usted mismo. No le vamos a poner la «muestra».

Es todo por el momento. Emplearemos un **«formato»** tradicional para despedirnos. **Sin otro particular le saluda atentamente.**

¡Hasta pronto!

PROGRAMAS

```
10 REM program I
20 CLS
30 INPUT "escribe un numero:  ",num
ero
40 PRINT "123456 espacios"
50 PRINT USING "######";numero
60 GOTO 30
```

```
10 REM programa II
20 CLS
30 DIM numero (5)
40 contador=0
50 WHILE contador<5
60 INPUT "teclea un numero: ";numer
o(contador)
70 contador=contador+1
80 WEND
90 contador=0
100 WHILE contador<5
110 PRINT USING "######.###";numero
(contador)
120 contador=contador+1
130 WEND
```

BUCLES ANIMADOS

Primeros repasos

La semana pasada vimos que un bucle repite una serie de instrucciones por un determinado número de veces, las que nosotros decidimos. Las líneas comprendidas entre FOR y NEXT son lo que llamamos cuerpo del bucle o instrucciones que se repiten.

Vamos a resumir lo que ya sabemos sobre bucles para continuar profundizando en su utilización.

El número de veces que el bucle gira está determinado por la **«variable de control»,** que comienza teniendo un contenido igual al valor inicial y se incrementa —*o decrementa cuando el número que siga a STEP sea negativo*— cada vez que el programa llega a NEXT.

El bucle termina cuando el valor de la variable de control se salga del rango establecido por los valores inicial y final.

Las instrucciones contenidas entre FOR y NEXT pueden ser cualquiera, así que vamos a ampliar las posibilidades del bucle, haciendo que el cuerpo del mismo sea otro FOR... NEXT. Es lo que llamamos **«bucles anidados».**

Su funcionamiento es exactamente igual a lo que ya sabemos. Cada vez que se repite el bucle exterior, el interior se ejecuta completamente hasta totalizar su número de repeticiones. Pero vamos a verlo gráficamente con el programa 1. Observe que cada uno de los bucles lleva su propia variable de control.

No se deje impresionar por los bucles anidados, parecen mucho más complicados de lo que en realidad son. La clave de todo está en asumir, que por cada ciclo sencillo del exterior, el interior completa todos los suyos.

Para mayor claridad añada la siguiente línea al programa:

55 PRINT «*EXTERIOR*=»exterior, «*INTERIOR*=»interior

Mientras estuvimos con el programa 1 debimos señalar que no se necesita poner la variable de control detrás de NEXT. El **Amstrad** está capacitado para mantenerse al tanto de las cosas, pero ¿y nosotros?

Le recomendamos que las ponga siempre, porque añaden una mayor claridad a la hora de seguir el programa. Aunque funcione correctamente sin ellas, nosotros podemos terminar hechos un lío.

Más allá de dos bucles anidados

Hasta aquí sólo hemos anidado un bucle FOR... NEXT dentro de otro. Es perfectamente válido tener 3 o más bucles anidados, consiguiendo que nuestros programas sean mucho más flexibles y poderosos. Veamos un ejemplo en el que hay más de dos bucles anidados. Añada al programa:

45 FOR intermedio=1 TO 3
47 PRINT TAB (5) «Este es el bucle intermedio»
intermedio
75 NEXT intermedio

y estas tres únicas líneas, le mostrarán la enorme potencia de esta técnica, al ver el incremento de los resultados producidos.

Una cosa, debemos ser muy cuidadosos cuando manejemos bucles anidados, sobre todo en dos puntos:

— Los NEXT deben ir colocados en su orden correcto ya que si no nos exponemos a confundir al **Amstrad** que nos dedicará uno de sus **«mejores»** mensajes de error.

— Vigilar los límites de la variable de control para que no provoquen un número mayor o menor de ciclos que los deseados. ¡Ojo con ellas!

Hablando de variables, podemos hacer que el número de veces que se repite cada vez el bucle interior dependa directamente de la variable de control del exterior.

Basta, por ejemplo, con poner como límite superior de la que controla el bucle interior el contenido que tiene en cada vuelta la del exterior.

De este modo, además de estar anidado, el bucle exterior controla al interior. El programa 2 es bastante revelador en este sentido. Estúdielo detenidamente.

Pero además estos bucles pueden hacerse con valores decrecientes. Véalo cambiando la línea 20 del programa 2 por:

20 FOR longitud=10 TO 1 STEP —1

para realizar el mismo triángulo pero al revés. Y también hacer con ellos una imagen «reflejada en un espejo», como lo demuestra el programa 3.

O volverlo al revés cambiando las líneas:

30 FOR línea=10 TO 1 STEP —1
50 LOCATE 11-longitud, 11-línea

Los bucles anidados no son el uso más importante que podemos darle a nuestro **Amstrad,** pero le serán muy útiles si los aplica en sus propios programas.

```
10 REM PROGRAMA I
20 FOR exterior=1 TO 3
30 PRINT"Este es el bucle exterior"
exterior
40 FOR interior=1 TO 3
50 PRINT TAB(10)"Este es el bucle i
nterior"interior
60 NEXT interior
70 NEXT exterior


10 REM PROGRAMA II
20 FOR longitud=1 TO 10
30 FOR linea=1 TO longitud
40 PRINT"*";
50 NEXT linea
60 PRINT CHR$(13)
70 NEXT longitud


10 REM PROGRAMA III
20 CLS
30 FOR linea=1 TO 10
40 FOR longitud=1 TO linea
50 LOCATE 11-longitud,linea
60 PRINT"*"
70 NEXT longitud
80 NEXT linea
```

MICRO-1

el IVA lo paga MICRO-1

C/ Duque de Sesto, 50. 28009 Madrid
Tel.: (91) 275 96 16/274 53 80
(Metro O'Donell o Goya)
Aparcamiento gratuito en Felipe II

SOFTWARE: ¡¡2 PROGRAMAS POR EL PRECIO DE 1!!
Y además, completamente gratis, un magnífico reloj de cuarzo. Increíble ¿verdad?

	Ptas.
PING PONG	2.295
SABOTEUR	2.295
RAMBO	2.295
YIEAR KUNG FU	2.295
WORLD SERIES BASEBALL	2.095
MAPGAME	2.750
RAID	2.295
HYPERSPORTS	2.295
HIGHWAY ENCOUNTER	1.750
HIGHWAY ENCOUNTER DISCO	3.300
ALIEN B	1.750

	Ptas.
DYNAMITE DAN	2.100
SABRE WULF	1.650
THEY SOLD A MILLION	2.500
FIGHTER PILOT	1.975
MASTER OF T. LAMP	1.950
NIGHIT SHADE	1.950
HACKER	1.950
SUPER TEST	2.300
TORNADO LOW LEVEL DISCO	3.300
TORNADO LOW LEVEL	1.750
KNIGHT LORE	1.750

SOFTWARE DE REGALO: ¡¡OFERTA 2 x 1!!

Beach Head Decathlon Dummy Run Beach Head Southern Belle

Fabulosos precios para tu Amstrad CPC-464 CPC6128 PCW-8256 y PCW-512

SOFTWARE DE GESTION PROFESIONAL

DBA II	17.800	DR. GRAPH	15.100
CBASIC	15.100	CONTABILIDAD	
DR DRAW	15.100	Y VTOS.	16.600

IMPRESORAS ¡¡20% DTO. SOBRE P.V.P.!!

COMPATIBLE IBM PC-XT 256 K Y DOS DISKETTES DE 360 K **229.900 PTAS.**

UNIDAD DE DISCO 5¼'' PARA AMSTRAD **34.900 PTAS.**

LAPIZ OPTICO + INTERFACE **3.495 PTAS.**

CINTA VIRGEN ESPECIAL ORDENADORES **69 PTAS.**

SINTETIZADOR DE VOZ EN CASTELLANO 15% DTO.
CASSETTE ESPECIAL ORDENADOR **5.295 PTAS.**

JOSTICK QUICK SHOT II **1.995 PTAS.**
JOYSTICK QUICK SHOT V **2.295 PTAS.**
con la compra de un joystick ¡¡GRATIS 1 RELOJ DE CUARZO!!

DISKETTE 5¼'' **295 PTAS.**

DISKETTE 3'' **990 PTAS.**

Pedidos contra reembolso sin ningún gasto de envío. Tels. (91) 275 96 16/274 53 80, o escribiendo a Micro-1. C/ Duque de Sesto, 50. 28009 Madrid

PING PONG

Competir contra el ordenador en cualquier prueba deportiva, siempre es un reto a nuestra habilidad. En esta ocasión Konami nos invita a disputar una emocionante partida de ping pong, contra un adversario que lo devuelve todo.

El deporte constituye un importante filón para realizar juegos de ordenador; recordemos los programas de fútbol, béisbol, competiciones olímpicas, decathlon, etc.

Una vez agotados los deportes más populares y conocidos, las casas de software dedicadas a la producción de programas, tienen que recurrir a deportes más minoritarios pero que aporten un cierto atractivo, y en los que la competición y el afán de superar al contrario,, sean pieza clave creando una adicción difícil de contener.

Konami, ha escogido adaptar el ping pong al ordenador, como hermano menor del tenis, al jugarse en una superfice mucho más pequeña, con una pelota ínfima y demás elementos, la adaptación presentaba serios problemas, los cuales se han resuelto con gran astucia por parte de los diseñadores del juego. En vez de reproducir el partido tal y como se juega de verdad, se han introducio ciertos trucos que nos permiten disfrutar al máximo del programa.

En primer lugar hemos de señalar que se ha prescindido de la figura humana de los jugadores, medida muy oportuna pues al tener un tamaño casi tan grande como la mesa de juego no nos dejarían apreciar el movimiento de la pelota.

En su lugar lo único que aparecen son las dos raquetas con la mano que las empuña, permitiendo de esta manera que tengamos una visión total de la mesa de juego.

Nuestro jugador se encuentra en la parte más próxima a la pantalla y el adversario se encuentra en el fondo de la pista, de este modo podemos apreciar perfectamente el recorrido de la bola cuando viene hacia nosotros, podemos ejecutar con precisión los golpes que imprimimos a la pelota y la dirección en que la movemos.

En este punto es donde el juego adquiere su máxima brillantez, pues se han contemplado todos los golpes que se utilizan en el ping pong.

Con el uso del joystick y un poco de práctica, apreciaremos la amplia gama de ellos, que podemos utilizar.

Tenemos dos golpes de drive; uno listado y otro cortado.

El saque se ejecuta lanzando la pelota al aire y golpeando con la raqueta con el golpe que queramos, bien de derecha o de revés. De revés también se ejecutan los golpes con listado y cortado. Cuando la pelota es devuelta por el otro jugador a una altura considerable, inmediatamente entra en acción el mate de la bola, el cual lo podemos ejecutar de drive o revés según nos convenga.

También cuando nos lanzan un mate, podemos amortiguar la bola con un golpe cortado, el cual nos permitirá recuperar el control del punto.

Como se ve, la gama de golpes es completa y permite acercarse a lo que es en realidad el juego del ping pong.

Mister JOYSTICK

Pero todo esto no sería de nada, si no fuésemos capaces de cóntrolar la dirección de la bola, cosa imprescindible para mover al contrario y obligarle a fallar. Esta es perfectamente controlable, por la anticipación o retardo conque golpeemos la pelota; si dejamos que ésta venga hacia la raqueta, nuestro golpe irá dirigido hacia la derecha, por el contrario si nos anticipamos a la bola ésta se dirigirá hacia la izquierda.

Para colmo el efecto de aproximación y alejamiento de la bola, se aprecia perfectamente por el efecto tridimensional, que hace que ésta aumente de tamaño cuando se acerca, disminuyendo al alejarse, cuando la bola se eleva su tamaño aumenta aún más y en cualquier momento sabemos a qué altura y distancia se encuentra de nosotros.

El juego está dotado de cinco niveles distintos, en los cuales la velocidad del juego y la pericia del jugador que controla el ordenador, aumentan según el nivel sea más alto, haciendo que cuando dominemos el nivel uno, pasemos al dos en el que nuestro adversario y la velocidad de la bola nos lo ponen más difícil, aumentanto la emoción él ping pong.

Un juego con peloteos y emoción incesante, que obliga a mantener una férrea concentración y nos tiene en constante tensión en la disputa de cada punto.

UN BIT SIGNIFICATIVO

Hemos hecho un largo recorrido en nuestra exploración sobre el parámetro de canal del comando SOUND. Vimos cómo los valores 1, 2 y 4 pueden utilizarse para reproducir notas por los canales A, B y C respectivamente.

También aprendimos cómo podíamos combinarlos de forma que:

SOUND 7,100,100,7

produzca la misma nota en los tres canales.

Nos encontramos con que añadiendo 8, 16 ó 32 provocamos que los sonidos sincronicen sus notas en los canales A, B y C. Esto significa que cuando intentemos producir una nota en el canal A con:

SOUND 33,100,200,6

no podremos escuchar nada hasta que le demos una «cita» con una nota del canal C usando:

SOUND 12,200,200,5

Recuerde que ambas notas deben señalarse con el parámetro de sincronismo. Usando uno tal como:

SOUND 4,200,200,5

no liberaríamos la nota anterior de la cola de espera del canal A. Pruébelo y lo verá.

Hablando de colas de espera, cada una de ellas puede contener cuatro notas además de la que está sonando en el momento, y esto nos conduce al parámetro retención.

Sumándole 64 al parámetro de canal de un comando SOUND seremos capaces de asegurar que cuando la nota alcance la cabecera de la cola espere allí hasta que la liberemos.

Número	Bit activado	Resultado
1	0	Usa canal A
2	1	Usa canal B
4	2	Usa canal C
8	3	Citada con A
16	4	Citada con B
32	5	Citada con C
64	6	Esperando a RELEASE
128	7	Limpia el canal

Tabla I: Parámetro de canal, valores y acciones.

En la práctica significa que cuando introducimos en el ordenador algo semejante a:

SOUND 1,300,100,7:
SOUND 65,400,100,7

sólo oiremos la primera nota. La segunda permanece en la cola de espera del canal A hasta que la hagamos aparecer con:

RELEASE 1

Finalmente vimos cómo podemos unir todos los valores combinándolos en un parámetro como:

SOUND 81,200,100,7

para producir una nota por canal A que no sólo está retenida sino que espera sincronizarse con otra del B. Para conseguir que este «tímido» sonido salga de su escondrijo tendremos que liberarlo con un:

RELEASE 1

y entonces sonará cuando aparezca una nota por el canal B tal como:

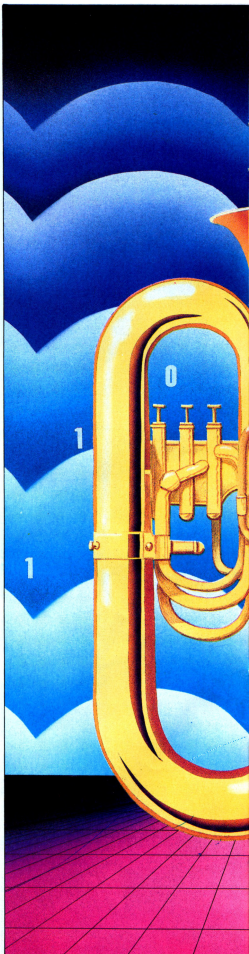
SOUND 10,100,100,3

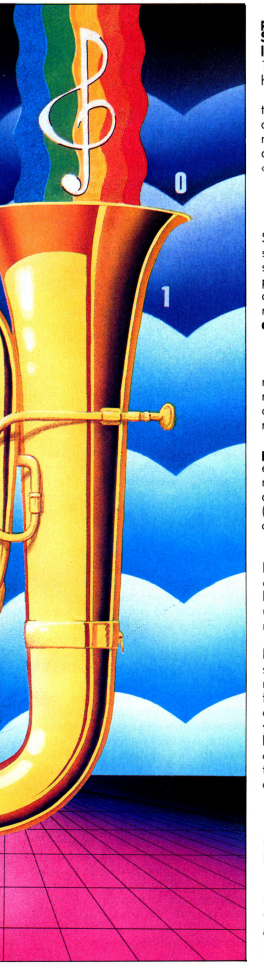
Teclas sonoras

Esperamos que se haya divertido con todo lo relacionado con los parámetros del comando SOUND. No es la mejor forma de aprender ya que, con el clip de sonido que tiene nuestro **Amstrad**, es bastante más útil practicar que quedarnos simplemente en la pura teoría. No hay duda de que cuando estemos manejándolo podemos encontrarnos metidos dentro de un «pozo» sin salida del que solamente nos sonará la tecla ENTER a la que previamente hemos asignado:

KEY 139,«SOUND 135,0,0,0»+CHR$(13)

Sin esta ayuda nos habrá dado frecuentemente dolor de cabeza intentar escapar de una cacofonía o un infranqueable muro de silencio.





SONIDO

¿Ha observado algo extraño en el parámetro de canal del comando SOUND mientras estaba escribiendo la anterior instrucción? Su valor es 135, bastante más alto que los que hemos utilizado hasta ahora.

Como habrá podido suponer, se trata de otro nuevo número que podemos añadirle al parámetro de canal. En este caso es el que origina que el canal o canales oportunos se «limpien» o liberen de notas.

Pruebe un sonido semejante a:

```
SOUND 1,200,100,7
```

Es un poco monótono, **¿verdad?** Suponga por un momento que necesitamos conseguir una nota que esté sonando en el canal A. Mientras el programa esté ejecutándose no podremos pulsar la tecla ENTER, definida previamente, así que **¿cómo podríamos hacerlo?**

Utilizando:

```
SOUND 1,200,100,7
```

no conseguiremos ningún efecto, tenemos atascada la cola de espera del canal A y la primera nota continúa hasta terminar.

¿Por qué no usar el parámetro limpiador 128? Lo que queremos hacer es despejar el canal A, que tiene como parámetro el 1, así que el valor que estamos necesitando es 129 (1+128). Con esto le estamos indicando que:

```
SOUND 129,200,100,7
```

hará el trabajo por nosotros, cortando la primera nota y eliminando sus huellas. Ahora comenzaremos a oír una de tono 200, volumen 7 que durará un segundo.

En otras palabras, el parámetro limpiador da prioridad a una nota sobre otra que está sonando o esperando en la cola. Un valor 129 echa fuera todas las notas que haya en el canal A mientras que 130 (2+128) y 132 (4+128) hacen lo propio en los canales B y C. Ocurre así incluso cuando las notas de la cola están retenidas o citadas para sincronizarse con las de otro canal.

Introduciendo:

```
SOUND 66,400,200,7
```

retenemos una nota en la cola de espera del canal B. Sin embargo:

```
SOUND 130,400,200,7
```

se deshace de la primera nota —*y de cualquier otra que pudiera estar esperando en dicho canal*— y hace sonar otra que tiene un tono 400, un volumen 7 durante 2 segundos.

De la misma forma:

```
SOUND 12,100,100,7
```

está esperando para sincronizarse con una del canal A. Cualquier otra nota del canal C tendrá que colocarse en un lugar más atrasado dentro de la cola de sonido y esperar turno hasta que se produzca la reunión. Y sucederá así, por supuesto, a menos que se cuele por medio:

```
SOUND 132,1000,300,6
```

que se libra de todas las demás notas y suena durante 3 segundos.

Limpiar el sonido

También podemos mezclarlo o emparejarlo con otros valores de este mismo parámetro. Si nos encontramos con 131 —1+2+128— cumple la misma función pero esta vez con A y C. Como podemos adivinar:

```
SOUND 135,0,0,0
```

barre las notas que haya en todos los canales. Aunque estén sonando, retenidas, aguardando para sincronizarse o simplemente esperando, actúa sobre todas. Pero esta vez la última nota tiene un tono, volumen y duración igual a cero, así que no suena ninguna nota más. En efecto:

```
SOUND 135,0,0,0
```

se deshace de todos los comandos SOUND precedentes. Como vimos cuando asignamos una función a la tecla ENTER, esta instrucción puede prestarnos una ayuda muy beneficiosa.

Y el parámetro **«limpiador»**, es el último que nos vamos a encontrar, le alegrará saberlo, en nuestro tratamiento sobre el parámetro de canal.

La Tabla 1 nos lo resume:

TABLA 1

bit number	7	6	5	4	3	2	1	0
binary byte	0	0	0	0	0	0	0	1

Figura I: Posiciones del bit en el binario 1.

Observe que hay algunos valores en ella a los que hemos denominado «bit activo», de un rango comprendido entre 0 y 7. Están indicándonos lo que en el parámetro del comando SOUND se conoce como «bit significativo».

Esto quiere decir que cuando pasamos a binario el número de canal, que generalmente está en decimal, los ceros y unos se usan como flags para activar o desactivar los diferentes aspectos de los parámetros de canal.

Con un byte basta

Para ilustrarlo veremos un sencillo ejemplo donde el parámetro de canal de un comando SOUND es 1. El número binario equivalente a 1 es, ¡sorpréndase!, 1, o mejor dicho 00000001 cuando queremos ponerlo con ocho cifras. Si no nos comprende del todo, introduzca en su micro:

```
PRINT BIN$(1,8)
```

y nos saldrá:

```
00000001
```

Cada una de estas ocho cifras es lo que llamamos un «bit» que forma parte de un «byte», o representación de un decimal en ocho cifras binarias. El bit que está más a la derecha —en este caso es el 1— es conocido como bit 0, el que está una posición más a la izquierda —aquí un 0— como bit 1, el siguiente como bit 2 y así sucesivamente hasta el último, el bit 7. La figura 1 muestra los números de los bits, o posiciones, que hay dentro de un byte.

Vamos a intentar conseguir expresar el número 2 como el contenido de un byte expresado en binario. Para ello tecleamos:

```
PRINT BIN$(2,8)
```

obteniendo:

```
00000010
```

En este caso el bit 0 es 0 mientras que el bit 1 es un 1. Los restantes (2 a 7) son todos 0. La figura 2 nos enseña cómo quedaría.

bit number	7	6	5	4	3	2	1	0
binary byte	0	0	0	0	0	0	1	0

Figura II: Posiciones de bit en el binario 2.

No sabemos usted, pero nosotros encontramos que hablar de que el bit 0 vale 0 y el bit 1 vale 1, es un poco confuso. Preferimos decir que un bit «se activa» si es igual a 1 o se desactiva si es igual a 0. De esta forma, en el último ejemplo sólo hay un bit **«activado»** —1— mientras que todos los demás están desactivados —0.

Veamos qué pasa con nuestro último parámetro de canal, 4, con un rápido:

```
PRINT BIN$(4,8)
```

y nos dará:

```
00000100
```

El bit 2 está activado mientras que todos los demás están desactivados.

Ahora compárelos con la columna «bit activo» de la Tabla 1. Con un parámetro de canal igual a 1 vimos que el bit 0 estaba activado. Por la Tabla sacamos que ha de ser el canal A el que usamos para producir la nota. Así que si el bit 0 está activo usamos el canal A.

De un modo semejante, si está activado el bit 1, usamos el canal B y si es el bit 2, seleccionamos el C. Podemos observar que si activamos un bit se produce el resultado indicado por la siguiente columna de la Tabla. Esto es lo que queríamos contarles cuando indicábamos que el parámetro de canal estaba compuesto por bits significativos. Obtenemos una acción diferente según qué bit de los que forman el byte esté activado.

Suponga ahora que tenemos una nota sonando por los 3 canales con un parámetro que vale 7. Su representación en binario es 00000111. Compruébelo con:

```
PRINT BIN$(7,8)
```

si duda de nuestras palabras.

Como habrá podido observar, los bits 0, 1 y 2 están activados. Mirando la Tabla vemos que esto significa que las notas van a salir por los canales A, B y C.

Y podemos generalizarlo para todos los demás valores que demos al parámetro de canal y que puede llegar hasta 255. Si los pasamos a un número binario de 8 bits deduciremos qué funciones se activan o desactivan para cada bit en particular.

Veamos un último ejemplo. Un parámetro de canal igual a 135 limpia de notas todos los canales. Si hacemos:

```
PRINT BIN$(135,8)
```

nos dará el siguiente byte:

```
10000111
```

Estudiándolo bit a bit es fácil asegurar que la acción limpiadora que se pone en marcha cuando el bit 7 está activo afectará a los 3 canales A, B y C, ya que los bits 0 a 2 están también activos.

Pruebe todo lo anterior dando cualquier valor al parámetro del canal y examine su representación binaria equivalente para ver qué ocurre. Es maravilloso poder conocer el efecto producido por determinado valor dado a dicho parámetro, por ejemplo 97, partiendo del análisis del número binario que le corresponde, 01100001. En este caso los bits 6, 5 y 1 están activados, así que es evidente, mirando la Tabla 1, que obtendremos una nota por el canal A esperando una «cita» o sincronismo con otra del canal C. Hasta que nos familiaricemos con ello, parece un poco extraño que la versión binaria de un número decimal detalle tan fielmente los resultados que va a producir el parámetro de canal.

Y esto es todo por esta semana. Para la próxima esperamos que haya comprendido perfectamente todo lo relacionado con los «bits significativos» y abandonaremos el parámetro de canal para irnos directamente al comando SQ. Sus conocimientos sobre la instrucción SOUND se verán incrementados un significativo «bit».

Con estos tres programas LO VERA TODO MAS CLARO

Contabilidad
P.V.P. 19.900

Facturación
P.V.P. 15.500

Control de stock
P.V.P. 14.900

PCW 8256

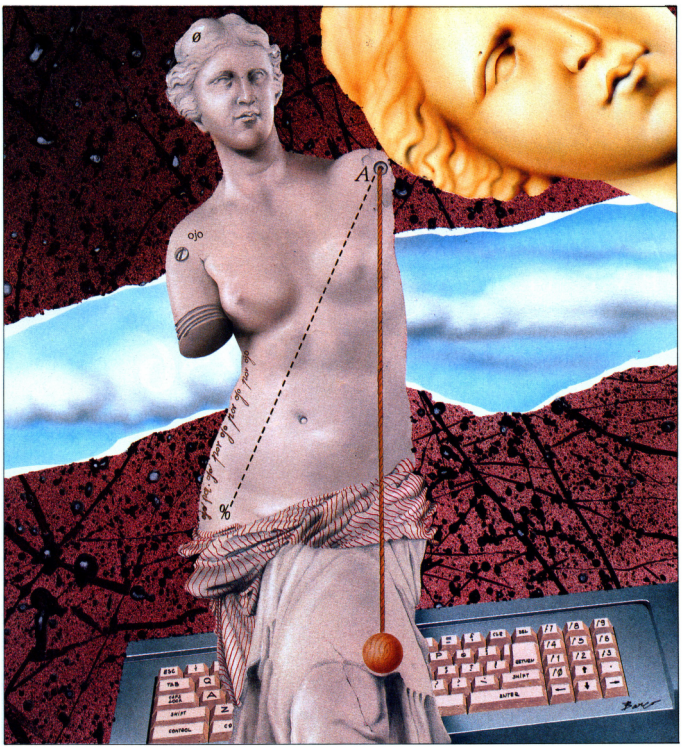
ORDEMANIA SOFT

te sorprenderá

Torres Quevedo, 34
Tel. (967) 22 79 44
Código Postal 02003
Albacete

MICROSAVE

Esta semana el programa que nos ocupa está escrito en una mezcla de Basic y código máquina; a priori tal vez parezca un tanto inusual, pero la razón de incluirlo en esta sección es muy clara: la importancia del tema que este programa aborda.
Se trata de una aplicación que pasa programas de cinta a cinta, y que, entre otras cosas, permite sacar copias de seguridad cómodamente de nuestros programas favoritos.



Se trata de un programa que copia, desprotege, acelera y da todos los datos de la cabecera de un programa.

Este programa dispone de nueve opciones cuyo uso expongo a continuación:

Load (L): Carga programas en la memoria del ordenador y da todos los datos (nombre, tipo, longitud, dirección, etc.) de cada bloque. Asimismo, da un código a cada bloque (A, B, C..., N) para poder identificar luego los bloques al grabarlos.

Save (S): Graba programas en la cinta. Su sintaxis es como la del LIST del Basic; se escribe: S-Bloque inicial-Bloque final. Por ejemplo: si quisiéramos grabar los bloques del C al F pondríamos: S C-F (es importante el espacio entre la S y la C). Si quisiéramos grabar desde el bloque D en adelante: S D-. Si quisiéramos grabar todos los bloques pondremos sencillamente S y, si sólo queremos grabar el E pondremos S E.

Cat (C): Realiza un catálogo de la cinta dando todos los datos. Para volver a Menú pulsar ESC y luego mantener el TAB hasta que vuelva al Menú.

Velocidad (V): Cambia la velocidad de grabación. Dispone de cuatro velocidades: Lenta (1.000 bandios), Normal (2.000), Rápida (2.500) y Turbo (4.000), con lo que conseguimos que cada bloque tarde 16, 8, 6 ó 4 segundos en grabarse. Pulsando V se cambia la velocidad.

Protección (P): Protege o desprotege los programas: tecleando P-ON, protege todos los bloques que se hallen en memoria y tecleando P-OFF o simplemente P los desprotege.

Name (N): Cambia el nombre de los programas. Su sintaxis es la siguiente: n-Nombre. Después, el ordenador coloca una flecha delante del primer bloque. Si se desea cambiar el nombre de ese bloque, se pulsa ESPACIO, si no ENTER, y lo mismo hacemos con el resto de los bloques.

Delete (D): Borra todos los bloques y deja espacio para cargar más programas.

Help (H): Hace un listado de los comandos que posee este programa.

End (E): Finaliza el programa y le deja en condiciones de ser grabado.

Este programa sólo sirve para copiar programas de cinta a cinta, no puede utilizar el disco, no obstante, es *compatible* con todos los modelos **Amstrad**, pero los que tengan la suerte de poseer disco, deberán teclear un paso más: 15 ITAPE

No es bueno añadir más pasos al

ESTRUCTURA DEL PROGRAMA

10-40	Inicialización de variables para código máquina.
50	Carga de código máquina.
60-80	Definición de la pantalla.
90-120	Definición de variables y de funciones para el programa.
130	Presentación de comandos.
140-210	Entrada de opciones y llamada de subrutinas.
220-320	**Subrutina LOAD**
230	Activa motor del cassette.
260	Carga la cabecera.
270	Imprime datos.
280	Carga el programa.
290	Continúa con otro bloque o para el motor y vuelve.
330-520	**Subrutina SAVE**
340	En caso de no haber programas vuelve a menú.
350-400	Halla qué bloques debe cargar.
410	Activa el motor.
420	Bucle para grabar los bloques.
430	Bucle de espera.
460	Graba la cabecera.
470	Graba el programa.
490	Cierra el bucle.
510	Para el motor y vuelve a menú.
530-630	**Subrutina CAT**
540	Presentación.
560	Activa el motor.
580	Carga la cabecera.
590	Imprime datos.
600-620	Continúa o para el motor e imprime los programas que había en la memoria.
630	Vuelve a menú.
640-690	**Subrutina SPEED WRITE**
650	Cambia la velocidad.
670	Activa y escribe la velocidad Lenta o Normal.
680	Activa y escribe la velocidad Rápida o Turbo.
690	Vuelve a menú.
700-770	**Subrutina PROTECCION**
710	Halla si se ha de proteger o de desproteger.
720	Bucle que recorre todos los programas.
740	Protege o desprotege.
750	Imprime nuevo tipo de programa.
760	Cierra el bucle.
770	Vuelve a menú.
780-910	**Subrutina NAME**
800	Carga en n$ el nombre de los bloques.
810	Bucle que recorre todos los programas.
820-850	Señala cada programa y averigua si hay que cambiarle de nombre.
870	Cambia el nombre.
900	Cierra el bucle.
910	Vuelve a menú.
920-940	**Subrutina DELETE**
930	Limpia la pantalla y pone el n.º de programas a cero.
940	Vuelve a menú.
950-1000	**Subrutina HELP**
960-990	Imprime listado de comandos.
1000	Vuelve a menú.
1010-1050	**Subrutina END**
1020	Restablece el modo 2.
1030	Despedida.
1040	Rehabilita el ordenador para poder ser grabado.
1050	Finaliza el programa.
1060-1230	**Subrutina Datos:** Imprime los datos de la cabecera en pantalla.
1070	Imprime código (A, B, C,..., N).
1080-1110	Imprime Nombre.
1130-1180	Imprime Tipo: t=0 Basic t=1 Basic Protegido t=2 Binario t=3 Binario Protegido t=4 Pantalla t=22 ASCII
1200	Calcula la longitud, dirección inicial y dirección de autoejecución.
1210	Imprime el n.º de bloque y el n.º de bloques totales.
1220	Imprime longitud de bloque, dirección de carga, longitud total del programa y dirección de autoejecución.
1230	Vuelve.
1240-1290	**Subrutina Código Máquina**
1250	Establece el tope de memoria.
1260	Bucle para cargar el código máquina.
1270	Vuelve a programa principal.
1280	Datos de velocidad de grabación.
1290	Datas de LOAD cabecera. LOAD programa. SAVE cabecera. SAVE programa.

programa, pues está reducido al máximo pensando en los usuarios de cassette al grabar el programa en sólo dos bloques (4K).

Consejos para su utilización: Grabar el programa antes de ejecutarlo.

No pulsar ESC a no ser en la opción de CAT y en este caso, pulsarlo una sola vez (*jnunca dos!*).

Hay algunos pasos de programa muy complicados para el programador de Basic: procure entenderlos, están basados en la programación código máquina.

Cuidado con los DATAs, cualquier error puede ser fatal.

VARIABLES

Direcciones de código máquina

vgr	Velocidad de GRabación.
lcb	Load CaBecera.
lpr	Load PRograma.
scb	Save CaBecera.
spr	Save PRograma.
moton	MOtor ON.
motoff	MOtor OFF.
n	Número de programas.
vel	Velocidad de grabación.
o$	Opción elegida.
n$	Catálogo de opciones.
o	Valor numérico de la opción elegida.
i, j	Bucles
w, v	Valores transitorios.
lc	Dirección de carga de cabeceras.
lp	Dirección de carga de programas.
vi, vf	Valor inicial y final de número de programa para SAVE.
nu	N.° de programa.
prt	Protección: 0=OFF/-=ON.
n$	Nombre de programa en NAME.
i$	Tecla pulsada.
t, tt	Tipo de programa.
w1, w2, w3, w4	Longitud del bloque, dirección de carga, longitud total, dirección de autoejecución.
a	Valor leído de read.
	Funciones definidas
c (w)	Dirección de carga de cabeceras.
p (w)	Dirección de carga de programas.
b (v) y a (w)	Conversores de decimal a hexadecimal.
d (w)	Conversar de hexadecimal a decimal.

```
10 REM *** COPION P&S 1986 ***
20 MODE 2
30 vgr=7931:lcb=7940:lpr=7952:scb=7
964:spr=7976
40 moton=&BCE:motoff=&BC71
50 GOSUB 1240
60 WINDOW #0,1,80,3,16:WINDOW #1,1,
80,1,2:WINDOW #2,2,79,18,23
70 PRINT #1,TAB(32);"*** COPION P&S
 ***"
80 PRINT #1,CHR$(24);" N Nombre
          Tipo                Bk  Nb
    Long.   Lugar   L.Tot    Ejec.";C
HR$(24);
90 m$="LSCVPNDHE"
100 DEF FN c(w)=7936+w*64:DEF FN p(
w)=&976+w*2048
110 DEF FN b(v)=INT(v/256):DEF FN a
(w)=w-FN b(w)*256:DEF FN d(w)=PEEK(
w)+PEEK(w+1)*256
120 n=0:v=1:SPEED WRITE 1
130 GOSUB 950:PRINT #2
140 WHILE INKEY$<>"":WEND
150 INPUT #2,"O";o$:o$=UPPER$(o$)
160 IF o$="" THEN 150
170 o=INSTR(1,m$,LEFT$(o$,1))
180 IF o=0 THEN PRINT #2,"Error":GO
TO 150
190 ON o GOSUB 220,330,530,640,700,
780,920,950,1010
200 PRINT #2
210 GOTO 140
220 REM #LOAD#
230 CALL moton
240 n=n+1:IF n=15 THEN n=14:PRINT #
2,"Memoria llena!":CALL motoff:RETU
RN
250 lc=FN c(n):lp=FN p(n)
260 POKE lcb+6,FN a(lc):POKE lcb+7,
FN b(lc):CALL lcb
270 LOCATE 1,n:nu=n:GOSUB 1060
280 POKE lpr+6,FN a(lp):POKE lpr+7,
FN b(lp):POKE lpr+3,PEEK(lc+19):POK
E lpr+4,PEEK(lc+20):CALL lpr
290 IF PEEK(lc+17)=0 THEN 240
300 FOR i=1 TO 300:NEXT i
310 CALL motoff
320 RETURN
330 REM #SAVE#
340 IF n=0 THEN PRINT #2,"Vacio":RE
TURN
350 o$=MID$(o$,3,3):IF o$="" THEN o
$="-"
360 w=ASC(LEFT$(o$,1))-64
370 IF w>=1 AND w<=n THEN vi=w ELSE
 vi=1
380 w=ASC(RIGHT$(o$,1))-64
390 IF w>=1 AND w<=n THEN vf=w ELSE
 vf=n
400 IF vi>vf THEN w=vi:vi=vf:vf=w
410 CALL moton
420 FOR i=vi TO vf
430 FOR j=1 TO 2000:NEXT j
440 LOCATE 1,i:PRINT "*";
450 lc=FN c(i):lp=FN p(i)
460 POKE scb+6,FN a(lc):POKE scb+7,
FN b(lc):CALL scb
470 POKE spr+6,FN a(lp):POKE spr+7,
FN b(lp):POKE spr+3,PEEK(lc+19):POK
E spr+4,PEEK(lc+20):CALL spr
480 PRINT CHR$(8);" "
490 NEXT i
500 FOR i=1 TO 2000:NEXT i
510 CALL motoff
520 RETURN
530 REM #CAT#
540 CLS
550 PRINT #2,"ESC y TAB para MENU"
560 CALL moton
570 lc=FN c(16)
580 POKE lcb+6,FN a(lc):POKE lcb+7,
FN b(lc):CALL lcb
590 PRINT " - ";:GOSUB 1080
600 IF INKEY(68)=-1 THEN 580
610 CALL motoff
620 CLS:FOR i=1 TO n:nu=i:lc=FN c(i
);:GOSUB 1060:NEXT i
630 RETURN
640 REM #SPEED WRITE#
650 vel=vel+1:IF vel=4 THEN vel=0
660 PRINT #2,"Velocidad: ";
670 IF vel<2 THEN SPEED WRITE vel:I
F vel=0 THEN PRINT #2,"Lenta" ELSE
PRINT #2,"Normal"
680 IF vel>1 THEN POKE vgr+1,130+(v
el=3)*35:POKE vgr+4,-(vel=3)*5:CALL
 vgr:IF vel=2 THEN PRINT #2,"Rapida
" ELSE PRINT #2,"Turbo"
690 RETURN
700 REM #PROTECCION ON/OFF#
710 prt=(MID$(o$,3,2)="ON")
720 FOR i=1 TO n
730 lc=FN c(i):lp=PEEK(lc+18)
740 IF prt=0 THEN POKE lc+18,p AND
254 ELSE IF p<4 THEN POKE lc+18,p O
R 1
750 nu=i:LOCATE 1,i:GOSUB 1060
760 NEXT i
770 RETURN
780 REM #NAME#
790 IF n=0 THEN PRINT #2,"Vacio":RE
TURN
800 n$=MID$(o$,3,16):n$=n$+STRING$(
16-LEN(n$),0)
810 FOR i=1 TO n
820 LOCATE 1,i:PRINT CHR$(243)
830 i$=INKEY$
840 IF i$=CHR$(13) THEN 890
850 IF i$<>" " THEN 830
```

SUBRUTINAS EN CODIGO MAQUINA

Velocidad de grabación: 7931

Ld HL, 130 d Velocidad.
Ld A, 0d. Precompensación.
CALL &BC68 Velocidad de cassette.
Ret

Load Cabecera: 7940

Ld DE, 64d. Tono de cabecera.
Ld A, 44d. Longitud de cabecera.
Ld HL, 0d. Dirección de carga (a determinar por programa).
CALL &BCA1 Load.
Ret

Load Programa: 7952

Ld A, 22d. Tono de programa.
Ld DE, 0d. Longitud de programa (a determinar).
Ld HL, 0d. Dirección de carga (a determinar).
CALL &BCA1 Load.
Ret

Save Cabecera: 7964

Como «Load Cabecera» pero llamando a:
&BC9E Save

Save Programa: 7978

Como «Load Programa» pero llamando a:
&BC9E Save.

DIVISION DE LA MEMORIA DEL ORDENADOR

A partir de:	
368	Programa en BASIC.
7931	Subrutinas en código máquina.
8000	Cabeceras de los programas de 64 bytes cada una.
9024	Programas grabados de longitud máxima 2048 bytes.
37696	Libre.

```
860 lc=FN c(i)
870 FOR j=0 TO 15:POKE lc+j,ASC(MID
$(n$,j+1,1)):NEXT j
880 LOCATE 4,i:PRINT n$;TAB(22)
890 LOCATE 1,i:PRINT " "
900 NEXT i
910 RETURN
920 REM #DELETE#
930 CLS:n=0
940 RETURN
950 REM #HELP#
960 PRINT #2,"Comandos:"
970 PRINT #2,"Load    Save     Cat"
980 PRINT #2,"Veloc   Protec   Name"
990 PRINT #2,"Delete  Help     End"
1000 RETURN
1010 REM #END#
1020 MODE 2
1030 PRINT "COPION P&S 1986":PRINT
```

Código MAQUINA

```
1040 MEMORY 20000
1050 END
1060 REM **Datos**
1070 PRINT " ";CHR$(64+nu);" ";
1080 IF PEEK(lc)=0 THEN PRINT "Sin
Nombre";:GOTO 1120
1090 FOR w=0 TO 15
1100 IF PEEK(lc+w)>31 THEN PRINT CH
R$(PEEK(lc+w)); ELSE PRINT" ";
1110 NEXT w
1120 PRINT TAB(23);
1130 t=PEEK(lc+18):tt=t AND 6
1140 IF tt=0 THEN PRINT "BASIC";
1150 IF tt=2 THEN PRINT "BINARIO";
1160 IF tt=4 THEN PRINT "PANTALLA";
1170 IF tt=6 THEN PRINT "ASCII";
1180 IF t AND 1 THEN PRINT " PROTEG
IDO";
1190 PRINT TAB(41);
1200 w1=FN d(lc+19):w2=FN d(lc+21):
w3=FN d(lc+24):w4=FN d(lc+26)
1210 PRINT USING"  ##";PEEK(lc+16);
INT((w3-1)/2048)+1;
1220 PRINT USING"     #####";w1;w2;w3
;w4;
1230 RETURN
1240 REM **Codigo Maquina**
1250 MEMORY vgr-1
1260 FOR i=vgr TO vgr+56:READ a:POK
E i,a:NEXT i
1270 RETURN
1280 DATA 33,130,0,62,0,205,104,188
,201
1290 DATA 62,44,17,64,0,33,0,0,205,
161,188,201,62,22,17,0,0,33,0,0,205
,161,188,201,62,44,17,64,0,33,0,0,2
05,158,188,201,62,22,17,0,0,33,0,0,
205,158,188,201
```

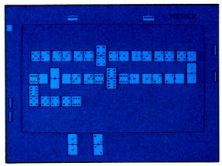
ESTRUCTURA GENERAL DEL PROGRAMA

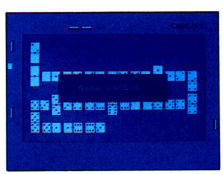
10-40	Inicialización de variables para código máquina.
50	Carga de código máquina.
60-80	Definición de la pantalla (ventanas y títulos).
90-120	Definición de variables y funciones para el programa.
130	Presentación de comandos.
140-210	Entrada de opciones y llamada de subrutinas.
220-320	Subrutina LOAD para cargar programas.
330-520	Subrutina SAVE para grabar programas.
530-630	Subrutina CAT para listar parcial o totalmente cintas.
640-690	Subrutina SPEED WRITE para variar la velocidad de 1.000 a 4.000 bandios.
700-770	Subrutina PROTECCION ON/OFF para proteger o desproteger programas.
780-910	Subrutina NAME para cambiar el nombre de los programas.
920-940	Subrutina DELETE para borrar programas y poder cargar más.
950-1000	Subrutina HELP para listar los comandos existentes.
1010-1050	Subrutina END para finalizar.
1060-1230	Subrutina Datos para listar los datos de las cabeceras en LOAD y CAT.
1240-1290	Subrutina Código Máquina para cargar el C/M que será utilizado por el programa en SAVE, LOAD y velocidad de grabación.

DOMINO

Eres un forastero que se une a una partida de dominó entre tres experimentados jugadores. Como tales su nivel de juego es bastante alto, por eso se te concede la oportunidad de elegir su nivel (0-9) y así evitar frustraciones.

Julio O. Marcos Matilla





Se reparten las fichas y... **¡Qué gane el mejor!**

Empieza, como siempre, aquél que tiene el seis doble y se van colocando fichas alternativamente hacia la derecha.

Al jugador que le corresponda tirar tiene su nombre escrito en la parte superior derecha de la pantalla y, para reconocerlo fácilmente, sus fichas en vídeo inverso.

Cuando sea su turno, pulsa cualquier tecla para avanzar en la elección de ficha y ***«ENTER»*** *para colocarla. Sólo avanzan hacia la derecha. Si no tienes ninguna ficha para tirar* ***«pasa»*** *simplemente pulsando la letra* ***«p»****.*

Se finalizará la partida, cuando algún jugador termine sus fichas o no se puedan colocar más. Se sumará a cada jugador los puntos de las fichas que le queden en su poder. Si alguno ha ganado, se le restan diez puntos al total que tenga en su haber.

Lógicamente irá ganando el que tenga menos puntuación. Al final aparecerá una tabla con los nombres de los jugadores, la puntuación de la última partida y los puntos totales. En otro color el que vaya en cabeza por puntos (el primero).

¡Ojo con las trampas! No se aceptan fichas inválidas, ni pasar cuando es posible colocar alguna ficha. Sólo conseguirás que renuncien tus compañeros de partida y que se te sumen 30 puntos al total.

Se te preguntará si quieres jugar otra partida, si dices sí, jugarás con el mismo nombre y continuará el juego; si no podrá jugar otra persona y comenzará la partida desde cero.

El juego

No resulta fácil describir un programa de 22 K en unas pocas hojas; sin embargo, podríamos dividirlo en tres partes: la pantalla inicial, el juego con la colocación de la ficha y el final.

En la pantalla inicial se incluye la rutina de caracteres y tintas, donde se crean los caracteres de las diferentes fichas en las diferentes posiciones y se crean las ventanas y tintas necesarias.

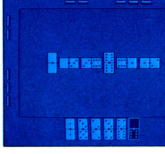
A continuación se crean las diferentes tablas y se dan valores a las variables. Las principales son:

p1$=número de ficha del jugador p1$ *(jugador, ficha)*

sen, col, n, x, y, doble son tablas utilizadas para la colocación de la ficha de la que ya hablaremos.

card$, valor, pasa y jugada son para la elección de la ficha durante el juego.

nam$ son los nombres de los jugadores.



Ahora asigna y reparte las fichas.

La ficha queda almacenada en forma de literal en card$ con ambos valores separados por una coma. Ejemplo: 6,5 es la ficha 6-5 ó 5-6.

Las fichas se reparten a cada jugador en la variable p1$ *(player, ficha)* y se vuelven a asignar de nuevo las fichas para que el ordenador sepa durante el juego las que van saliendo y las que quedan por salir.

Ahora ya podemos dibujar la pantalla inicial.

Se colorea el tablero *(c1s #1)* y se distribuyen las fichas. Las de los jugadores se colocan en vertical y las propias sobre el tablero con su valor.

Para dar el color del papel al gráfico se utiliza la rutina &BBE4. Según el número de parámetros, *(no el valor)*, que se manden a la rutina, el registro A se carga con el número total que se manden y la rutina se encarga de poner el papel de gráficos con el valor de la tinta que tenga A.

A continuación revisa las fichas de todos los jugadores para ver cuál tiene el seis doble; aquél que lo tenga, se le borra la ficha correspondiente y se coloca en el tablero para comenzar el juego.

NOTA: Cuando se escoge un nivel bajo, por ejemplo el 3, no significa que se vaya a coger la jugada con el nivel 3. Se calcula un número aleatorio entre 0 y 10-nivel elegido que sumado al nivel nos da una jugada entre 3 y 9; de esta forma se evita que la máquina haga muchas «burradas» en niveles inferiores. Lógicamente en el nivel 9 sólo saldrá la jugada ***«9»*** *pues se suma int (RND(10 — 9))=0*

```
10 REM
20 REM D O M I N O
40 REM
50 REM Por  Julio Marcos
60 REM
70 REM Para  MICROHOBBY AMSTRAD
80 REM
90 REM
100 MODE 1:DEFINT a-z
110 GOSUB 2930
120 GOSUB 410
130 GOSUB 170
140 GOSUB 3240
150 GOSUB 2460
160 GOTO 2210
170 REM --- Nombre y nivel ---
180 CLS
190 PEN 3:PAPER 0:CLS
200 GOSUB 3490
210 LOCATE 11,13:PRINT"ESCRIBE TU N
OMBRE"
220 nam$(4)=""
230 PEN 1
240 a$=INKEY$:IF a$="" GOTO 240
250 IF a$=CHR$(13) GOTO 330
260 IF a$<CHR$(127) THEN nam$(4)=LE
FT$(nam$(4),LEN(nam$(4))-1):GOTO 30
0
270 a$=UPPER$(a$)
280 IF a$<CHR$(32) OR a$>CHR$(128)
THEN PRINT#1,CHR$(7):GOTO 240
290 nam$(4)=nam$(4)+a$
300 LOCATE 16,20:PRINT nam$(4)+" ";
310 IF LEN(nam$(4))=9 GOTO 330
320 GOTO 240
330 REM --- Elige nivel ---
340 PAPER 0:PEN 3:INK 3,26:CLS
350 LOCATE 11,13:PRINT"ELIGE NIVEL
(0-9)";
360 a$=INKEY$:IF a$="" GOTO 360
370 IF a$<"0" OR a$>"9" GOTO 360
380 level=VAL(a$)
390 LOCATE 14,20:PEN 1:PRINT"Nivel
";level
400 RETURN
410 REM --- Asigna variables y tabl
as ----
420 ver=0:horr=1:si=1:no=0:casi=2
430 DIM pl$(4,7),sen(3),col(3),n(3)
,x(3),y(3),doble(3)
440 DIM card$(28),valor(4,7,3),pasa
(4,4),jugada(9,2),nam$(4)
450 nam$(1)="MARCOS":nam$(2)="MERCE
":nam$(3)="CARLOS"
460 RETURN
470 REM ******* ELIGE LA FICHA QUE V
A A TIRAR ********
480 REM
490 player=player MOD 4:dob=0
500 IF player>4 THEN player=1
510 CLS#2
520 PRINT#4,nam$(player)
530 GOSUB 3490
540 REM --- Comprueba que la ficha
se puede colocar
550 total=0
560 FOR ficha=1 TO 7
570 IF pl$(player,ficha)="" THEN ri
=10:n2=10:GOTO 600
580 ri=VAL(LEFT$(pl$(player,ficha),
1))
590 n2=VAL(RIGHT$(pl$(player,ficha)
,1))
600 FOR lado=1 TO 3 STEP 2
610 valor(player,ficha,lado)=0
620 IF ri=n(lado) OR n2=n(lado) THEN
 valor(player,ficha,lado)=1:total=1
630 NEXT lado
640 NEXT ficha
650 IF total=0 GOTO 1440
660 REM --- Asigna valores a las fi
chas ---
670 FOR ficha=1 TO 7
680 CLS#2
690 FOR lado=1 TO 3 STEP 2
700 IF valor(player,ficha,lado)=0 G
OTO 950
710 ri=VAL(LEFT$(pl$(player,ficha),
1))
720 n2=VAL(RIGHT$(pl$(player,ficha)
,1))
730 REM --- Comprueba si es ficha d
oble (100 p) ---
740 IF ri=n2 THEN valor(player,fich
a,lado)=100+ri+n2:dob=1:GOTO 950
750 IF n2=n(lado) THEN t=ri:ri=n2:n
2=t
760 REM --- Comprueba si la ficha c
ierra el juego ---
770 REM --- buscando mas fichas con
 n2 ---
780 total=0
790 FOR w=1 TO 28
800 IF INSTR(1,card$(w),RIGHT$(STR$
(n2),1))<>0 THEN total=total+1
810 NEXT w
820 IF total=1 GOTO 950
830 valor(player,ficha,lado)=valor(
player,ficha,lado)+10+n2+ri
840 REM --- Comprueba si algun play
er paso de n2 ---
850 FOR pl=1 TO 4
860 IF pl=player GOTO 980
870 IF pasa(pl,n2)=si THEN valor(pl
ayer,ficha,lado)=valor(player,ficha
,lado)+5+n2+ri
880 NEXT pl
890 REM --- Mira las fichas de play
er que contengan n2 ---
900 total=0
910 FOR fi=1 TO 7
920 IF INSTR(1,pl$(player,fi),RIGHT
$(STR$(n2),1))<>0 THEN total=total+
1+n2+ri
930 NEXT fi
940 valor(player,ficha,lado)=valor(
player,ficha,lado)+2*total
950 NEXT lado
960 NEXT ficha
970 REM --- Coloca y ordena los val
ores de las fichas ---
980 ERASE jugada:DIM jugada(9,2)
990 FOR ficha=1 TO 7
1000 FOR lado=1 TO 3 STEP 2
1010 IF valor(player,ficha,lado)<ju
gada(9,1) GOTO 1030
1020 jugada(9,1)=valor(player,ficha
,lado):jugada(9,2)=ficha:GOSUB 1310
1030 NEXT lado
1040 NEXT ficha
1050 REM --- Elige la ficha segun e
l nivel ---
1060 IF dob=1 THEN w=9:GOTO 1080
1070 w=level+INT((10-level)*RND)
1080 IF valor(player,jugada(w,1),ju
gada(w,2))=0 THEN w=w+1:GOTO 1080
1090 ficha=jugada(w,1)
1100 lado=jugada(w,2)
1110 ri=VAL(LEFT$(pl$(player,ficha)
,1))
1120 n2=VAL(RIGHT$(pl$(player,ficha
),1))
1130 v=1
1140 WHILE card$(v)<>pl$(player,fic
ha)
1150 v=v+1
1160 WEND
1170 card$(v)=""
1180 pl$(player,ficha)=""
1190 PLOT -5,-2,0
1200 GOSUB 1970
1210 GOSUB 3920
1220 GOSUB 2490
1230 REM --- Comprueba si player ha
 terminado ---
1240 total=0
1250 FOR fi=1 TO 7
1260 IF pl$(player,fi)<>"" THEN
total=1
1270 NEXT ficha
1280 IF total=0 GOTO 6550
1290 player=player+1
1300 IF player<>4 GOTO 470 ELSE 156
0
1310 REM --- Ordena los valores de
las fichas ---
1320 mcdi=fi+1
1330 WHILE mcdi<=si
1340 mcdi=mcdi+no
1350 FOR w=0 TO 8
1360 IF valor(player,jugada(w,1),ju
gada(w,2))<=valor(player,jugada(w+1
,1),jugada(w+1,2)) GOTO 1410
1370 FOR v=1 TO 2
1380 t=jugada(w,v):jugada(w,v)=juga
da(w+1,v):jugada(w+1,v)=t
1390 NEXT v
1400 mcdi=si
1410 NEXT w
1420 WEND
1430 RETURN
1440 REM --- Player pasa ---
1450 playpas=playpas+1:IF playpas=4
 GOTO 7540
1460 pasa(player,n(1))=si
1470 pasa(player,n(3))=si
1480 PRINT#2,"PASO";
1490 SOUND 1,700,10,7
1500 SOUND 1,700,30,7
1510 FOR q=1 TO 3000:NEXT
1520 CLS#2
1530 IF player=4 THEN GOSUB 3830
1540 GOSUB 3490
1550 GOTO 1290
1560 REM --- Turno del jugador nume
ro 4 ----
1570 CLS#2:player=4
1580 PRINT#4,nam$(player)
1590 PRINT CHR$(23)+CHR$(1)
1600 PLOT -2,-3,2
1610 ficha=1
1620 GOTO 1700
1630 WHILE INKEY$<>"":WEND
1640 a$=INKEY$:IF a$="" GOTO 1640
1650 IF a$="P" OR a$="p" GOTO 1440
1660 IF a$<CHR$(12) GOTO 1730
1670 GOSUB 3830
1680 ficha=ficha+1
1690 IF ficha>7 THEN ficha=1
1700 IF pl$(4,ficha)="" GOTO 1680
1710 GOSUB 3830
1720 GOTO 1630
1730 REM tira jugador 4
1740 REM comprueba la validez de la
 ficha
1750 ri=VAL(LEFT$(pl$(player,ficha)
,1))
1760 n2=VAL(RIGHT$(pl$(player,ficha
),1))
1770 IF ri=n(1) OR ri=n(3) OR n2=n(
1) OR n2=n(3) GOTO 1820
1780 PRINT#2,"INVALIDO";:SOUND 1,80
,60,7
1790 FOR w=1 TO 2000:NEXT
1800 CLS#2
1810 GOTO 1640
1820 REM ficha valida
1830 PRINT CHR$(23)+CHR$(0)
1840 PLOT -2,-2,0
1850 GOSUB 3920
1860 w=1
1870 WHILE pl$(4,ficha)<>card$(w)
1880 w=w+1
1890 WEND
1900 card$(w)=""
1910 pl$(4,ficha)=""
1920 IF pasa(4,ni)=si GOTO 7360
1930 t=ni:ni=n2:n2=t
1940 IF pasa(4,ni)=si GOTO 7360
1950 IF beginni=1 THEN RETURN
1960 GOTO 1230
1970 REM ------ Borra ficha de los j
ugadores 1-3 ---------
1980 REM
1990 CALL &BBE4
2000 TAG
2010 ON player GOSUB 2040,2100,2150
2020 TAGOFF
2030 RETURN
2040 REM -- Borra ficha de player 1
 ---
2050 xp=616
2060 yp=68+38*ficha
2070 MOVE xp,yp:PRINT CHR$(247);
2080 MOVE xp,yp+18:PRINT CHR$(246);
2090 RETURN
2100 REM --- Borra ficha de player
2 ----
```

```
2110 xp=150+38*ficha
2120 yp=392
2130 MOVE xp,yp:PRINT CHR$(24B);CHR
$(249);
2140 RETURN
2150 REM --- Borra ficha de player
3 ---
2160 xp=6
2170 yp=6B+38*ficha
2180 MOVE xp,yp:PRINT CHR$(251);
2190 MOVE xp,yp+16:PRINT CHR$(250);
2200 RETURN
2210 REM --- COMIENZO DE LA PARTIDA
 ---
2220 begin=si
2230 REM --- Busco el 6 doble ---
2240 player=1:ficha=1
2250 WHILE pl$(player,ficha)<>"6,6"
2260 ficha=ficha+1
2270 IF ficha=B THEN ficha=1:player
=player+1
2280 WEND
2290 LOCATE #1,10,3:PRINT#1,"COMIEN
ZA "jnam$(player)
2300 pl$(player,ficha)=""
2310 card$(2B)=""
2320 n(1)=3*n(1)+1:n(2)=3*n(2)+1:n(3
)=230
2330 topizd=62:topder=586
2340 sen(1)=-1:sen(3)=1
2350 col(1)=hor:col(3)=hor
2360 n(1)=6in(3)=6:doble(1)=si:doubl
e(3)=no
2370 lado=1
2380 ni=6:n2=6
2390 IF player=4 THEN GOSUB 1B20:GO
TO 2420
2400 GOSUB 1970
2410 GOSUB 3920
2420 LOCATE#1,10,3:PRINT#1,SPACE$(2
3);
2430 PAPER 2
2440 begin=no
2450 GOTO 1290
2460 REM --- Pantalla inicial ---
2470 REM
2480 PAPER 0:PEN 1:PAPER#1,2:PEN #1
,1
2490 INK 3,26:CLS:CLS#1
2500 MOVE 46,36B:DRAWR 546,0,1:DRAW
R 0,-306:DRAWR -546,0:DRAWR 0,306
2510 PRINT CHR$(23)+CHR$(0)
2520 TAG:PLOT -2,-2,1
2530 CALL &BBE4
2540 REM jugador 1
2550 x=616
2560 FOR y=106 TO 356 STEP 3B
2570 PLOT -2,-2,1
2580 MOVE x,y:PRINT CHR$(247);
2590 MOVE x,y+16:PRINT CHR$(246);
2600 MOVE x+B,y-12:DRAWR 0,26,3:MOV
E x+10,y-12:DRAWR 0,26,3
2610 NEXT y
2620 REM
2630 REM jugador 2
2640 y=392
2650 FOR x=1BB TO 452 STEP 3B
2660 PLOT -2,-2,1
2670 MOVE x,y:PRINT CHR$(24B)+CHR$(
249);
2680 MOVE x+2,y-4:DRAWR 26,0,3:MOVE
 x+2,y-6:DRAWR 26,0,3
2690 NEXT x
2700 REM
2710 REM jugador 3
2720 x=6
2730 FOR y=106 TO 356 STEP 3B
2740 PLOT -2,-2,1
2750 MOVE x,y:PRINT CHR$(251);
2760 MOVE x,y+16:PRINT CHR$(250);
2770 MOVE x+4,y-12:DRAWR 0,26,3:MOV
E x+6,y-12:DRAWR 0,26,3
2780 NEXT y
2790 TAGOFF
2800 REM
2810 REM jugador 4
2820 CALL &BBE4,0,0,0
2830 PLOT -2,-2,1
2840 ficha=1
2850 FOR x=1BB TO 452 STEP 3B
2860 y=60
2870 ni=VAL(LEFT$(pl$(4,ficha),1))
2880 n2=VAL(RIGHT$(pl$(4,ficha),1))
2890 GOSUB 5990
2900 ficha=ficha+1
2910 NEXT x
2920 RETURN
2930 REM ------------------ CARACTER
ES Y TINTAS ----------
2940 REM
2950 BORDER 11:INK 0,11:INK 1,0:INK
 2,13:INK 3,26
2960 PAPER 0:CLS
2970 WINDOW#1,4,37,3,21
2980 WINDOW#2,1,8,1,1
2990 WINDOW#3,7,34,7,17
3000 WINDOW#4,32,40,1,1
3010 WINDOW#5,11,30,10,14
3020 PAPER#1,2
3030 PAPER#2,0:PEN#2,1
3040 PAPER#3,1:PEN #3,3
3050 PAPER#4,0:PEN #4,1
3060 PAPER#5,1:PEN #5,0
3070 SYMBOL 240,126,126,126,126,126
,126,126,127
3080 SYMBOL 241,127,126,126,126,126
,126,126,126
3090 SYMBOL 242,1,255,255,255,255,2
55,255,0
3100 SYMBOL 243,128,255,255,255,255
,255,255,0
3110 SYMBOL 244,126,126,126,126,126
,126,126,254
3120 SYMBOL 245,254,126,126,126,126
,126,126,126
3130 SYMBOL 246,126,114,114,114,114
,114,114,115
3140 SYMBOL 247,115,114,114,114,114
,114,114,126
3150 SYMBOL 248,1,255,128,128,255,2
55,255,0
3160 SYMBOL 249,128,255,1,1,255,255
,255,0
3170 SYMBOL 250,126,78,78,78,78,78,
78,206
3180 SYMBOL 251,206,78,78,78,78,78,
78,126
3190 SYMBOL 252,255,128,128,128,128
,128,128,128
3200 SYMBOL 253,255,1,1,1,1,1,1,1
3210 SYMBOL 254,128,128,128,128,128
,128,128,255
3220 SYMBOL 255,1,1,1,1,1,1,1,255
3230 RETURN
3240 REM ------------ REPARTE FICHA
S ----------
3250 REM
3260 RANDOMIZE TIME
3270 GOSUB 3390
3280 REM Reparto Fichas
3290 FOR player=1 TO 4
3300 FOR ficha=1 TO 7
3310 x=INT(RND*2B)+1
3320 IF card$(w)="" GOTO 3310
3330 pl$(player,ficha)=card$(w)
3340 card$(w)=""
3350 NEXT ficha
3360 NEXT player
3370 GOSUB 3390
3380 RETURN
3390 REM ------------ ASIGNA FICHAS
 ----------
3400 REM
3410 w=1
3420 FOR a=0 TO 6
3430 FOR b=a TO 6
3440 card$(w)=RIGHT$(STR$(a),1)+","
+RIGHT$(STR$(b),1)
3450 w=w+1
3460 NEXT b
3470 NEXT a
3480 RETURN
3490 REM --- Colorea las fichas de
los jugadores al tirar ---
3500 REM
3510 PRINT CHR$(23)+CHR$(1)
3520 PLOT -2,-2,2:CALL &BBE4
3530 TAG
3540 ON player GOSUB 3570,3660,3740
3550 TAGOFF
3560 RETURN
3570 REM jugador 1
3580 xp=616
3590 FOR ficha=1 TO 7
3600 IF pl$(player,ficha)="" GOTO 3
640
3610 yp=6B+38*ficha
3620 MOVE xp,yp:PRINT CHR$(241);
3630 MOVE xp,yp+16:PRINT CHR$(240);
3640 NEXT ficha
3650 RETURN
3660 REM jugador 2
3670 yp=392
3680 FOR ficha=1 TO 7
3690 IF pl$(player,ficha)="" GOTO 3
720
3700 xp=150+38*ficha
3710 MOVE xp,yp:PRINT CHR$(242);CHR
$(243);
3720 NEXT ficha
3730 RETURN
3740 REM jugador 3
3750 xp=6
3760 FOR ficha=1 TO 7
3770 IF pl$(player,ficha)="" GOTO 3
810
3780 yp=6B+38*ficha
3790 MOVE xp,yp:PRINT CHR$(245);
3800 MOVE xp,yp+16:PRINT CHR$(244);
3810 NEXT ficha
3820 RETURN
3830 REM Colorea la ficha de player4
3840 TAG
3850 xp=1B8+(ficha-1)*38
3860 FOR yp=60 TO 0 STEP -15
3870 MOVE xp,yp
3880 PRINT CHR$(143);CHR$(143);
3890 NEXT yp
3900 TAGOFF
3910 RETURN
3920 REM ------------ CALCULA COO
RDENADAS DE LA FICHA ----------
3930 REM
3940 playpas=0
3950 IF player<>4 THEN a=lado:GOTO
 4040
3960 IF ni<>n(3) AND ni<>n(1) THEN
 tn=1:ni=n2:n2=t
3970 IF n2<>n(3) AND n2<>n(1) THEN
RETURN:REM Ficha no valida
3980 IF ni=n2 GOTO 5080
3990 IF ni(>n(3) GOTO 4020
4000 a=RND*2+1
4010 IF a>4 THEN a=3 ELSE a=1
4020 IF (ni=n(3) AND n2=n(1)) OR (n
2=n(1) AND n2=n(3)) THEN GOSUB 5540
:GOTO 4040
4030 IF ni=n(1) THEN a=1 ELSE a=3
4040 IF ni=n2 GOTO 5080
4050 IF col(a)=ver GOTO 4550:REM co
loca despues de una vertical
4060 IF doble(a)=1 GOTO 4330:REM co
loca despues de una doble horizonta
l
4070 REM coloca despues de una hori
zontal normal
4080 x=x(a)+34*sen(a):y=y(a)
4090 IF x>topizd AND x<topder GOTO
4270:REM caso 1
4100 x=x(a)+32*sen(a)
4110 IF x>topizd AND x<topder GOTO
4200:REM caso 2
4120 REM caso 3,top=si
4130 x=x-2
4140 IF sen(a)=1 THEN x=x-2
4150 y=y(a)+34*(a-2):IF a=3 THEN y=
y-2
4160 top(a)=si
4170 col(a)=ver
4180 doble(a)=no
4190 GOTO 4650
4200 REM caso 2,top=casi
4210 x=x(a)+34*sen(a):IF sen(a)=-1
THEN x=x-2
4220 y=y(a)
4230 IF a=3 THEN y=y-2
4240 top(a)=casi
4250 col(a)=ver
4260 GOTO 5650
4270 REM caso 1,top=no
4280 x=x(a)+6*sen(a)
4290 y=y(a)
4300 top(a)=no
4310 col(a)=hor
4320 GOTO 5650
4330 REM despues de ficha doble
4340 IF top(a)=si GOTO 4480
4350 IF top(a)=casi GOTO 4420
4360 REM top=no;caso 1
4370 x=x(a)+6*sen(a)
4380 y=y(a)+16*(a-2)
4390 col(a)=hor
```

```
4400 GOTO 5650
4410 doble(a)=no
4420 REM top=casi;caso 2
4430 x=v(a)
4440 y=y(a)+66*(a-2)
4450 col(a)=ver
4460 GOTO 5650
4470 doble(a)=no
4480 REM top=si;caso 3
4490 sen(a)=-sen(a)
4500 x=x(a)-2*sen(a)
4510 y=y(a)
4520 col(a)=hor
4530 doble(a)=no
4540 GOTO 5650
4550 REM despues de ficha vertical
4560 IF doble(a)=si GOTO 4730:REM c
oloca despues de una doble
4570 IF top(a)=casi GOTO 4660
4580 REM top=si;caso 5
4590 sen(a)=-sen(a)
4600 x=x(a)+98*sen(a):IF sen(a)=1 T
HEN x=x+2
4610 y=y(a)+30*(a-2):IF a=3 THEN y=
y+2
4620 top(a)=no
4630 doble(a)=no
4640 col(a)=hor
4650 GOTO 5650
4660 REM top=casi;caso 4
4670 sen(a)=-sen(a)
4680 x=x(a)+64*sen(a):IF sen(a)=-1
THEN x=x+2
4690 y=y(a)+64*(a-2):IF a=3 THEN y=
y+2
4700 col(a)=hor
4710 top(a)=no
4720 GOTO 5650
4730 REM ficha despues de doble ver
tical
4740 IF top(a)=si GOTO 5000
4750 x=x(a)+64*sen(a)
4760 IF x>topiz AND x<topder GOTO
4930:REM caso 1
4770 x=x(a)+32*sen(a)
4780 IF x>topiz AND x<topder GOTO
4860:REM caso 2
4790 REM caso 3;top=si despues de d
oble vertical
4800 x=x(a)
4810 y=y(a)+64*(a-2):IF a=3 THEN y=
y+2
4820 top(a)=si
4830 doble(a)=no
4840 col(a)=ver
4850 GOTO 5650
4860 REM caso 2;top=casi despues de
 doble vertical
4870 x=x(a)
4880 y=y(a)+64*(a-2)
4890 top(a)=si
4900 doble(a)=no
4910 col(a)=ver
4920 GOTO 5650
4930 REM caso 1;top=no despues de d
oble vertical
4940 x=x(a)+66*sen(a):IF sen(a)=1 T
HEN x=x+2
4950 y=y(a)+16*(a-2)
4960 top(a)=no
4970 doble(a)=no
4980 col(a)=hor
4990 GOTO 5650
5000 REM coloca ficha despues de do
ble con top=si
5010 sen(a)=-sen(a)
5020 x=x(a)+98*sen(a):IF sen(a)=si
THEN x=x+2
5030 y=y(a)+30*(a-2):IF a=3 THEN y=
y+2
5040 top(a)=no
5050 col(a)=hor
5060 doble(a)=no
5070 GOTO 5650
5080 REM coloca ficha doble
5090 IF n1=n(1) THEN a=1 ELSE a=3
5100 IF col(a)=ver GOTO 5360
5110 x=x(a)+64*sen(a)
5120 IF x>topizd AND x<topder GOTO
5290
5130 x=x(a)+32*sen(a)
5140 IF x>topizd AND x<topder GOTO
5220
5150 REM top=si;caso 3
5160 x=x(a):IF sen(a)=1 THEN x=x-2
5170 y=y(a)+34*(a-2)
5180 top(a)=si
5190 col(a)=ver
5200 doble(a)=si
5210 GOTO 5650
5220 REM top=casi;caso 2
5230 x=x(a)+34*sen(a):IF sen(a)=-1
THEN x=x-2
5240 y=y(a)-16*(a-2)
5250 top(a)=casi
5260 col(a)=ver
5270 doble(a)=si
5280 GOTO 5650
5290 REM top=no;caso 1
5300 x=x(a)+34*sen(a):IF sen(a)=-1
THEN x=x-2
5310 y=y(a)-16*(a-2)
5320 top(a)=no
5330 col(a)=ver
5340 doble(a)=si
5350 GOTO 5650
5360 REM coloca ficha doble en hori
zontal
5370 IF top(a)=si GOTO 5460:REM cas
o 1
5380 REM top=casi
5390 sen(a)=-sen(a)
5400 x=x(a)+48*sen(a):IF sen(a)=1 T
HEN x=x+2
5410 y=y(a)+66*(a-2)
5420 col(a)=hor
5430 doble(a)=no
5440 top(a)=no
5450 GOTO 5650
5460 REM top=si
5470 sen(a)=-sen(a)
5480 x=x(a)+98*sen(a):IF sen(a)=1 T
HEN x=x+2
5490 y=y(a)+30*(a-2):IF a=3 THEN y=
y+2
5500 top(a)=no
5510 col(a)=hor
5520 doble(a)=no
5530 GOTO 5650
5540 REM ------- Decide lado a colo
car ----------
5550 IF y(1)<370-y(3) THEN po=20 EL
SE po=1
5560 LOCATE#1,9,po
5570 PRINT#1,CHR$(174)+"A que lado?
 (D/I)"
5580 a$=UPPER$(INKEY$):IF a$="" GOT
O 5580
5590 IF a$="D" THEN a=3:GOTO 5620
5600 IF a$="I" THEN a=1:GOTO 5620
5610 GOTO 5580
5620 LOCATE#1,9,po:PRINT#1,SPACE$(1
9);
5630 PAPER 0
5640 RETURN
5650 REM ---------- DIBUJA LA FIC
HA ----------
5660 REM --- Borra la ficha del jug
ador ---
```

DESCRIPCION DEL PROGRAMA

Se utilizan las siguientes variables para el juego:

a=1 para el lado izquierdo y a=3 para el derecho; de este forma:

n(a) es la ficha del extremo *(la que queda libre)* de cada lado.

x(a) e *y(a)* son las coordenadas gráficas de la última ficha colocada en el lado a.

col(a) es la colocación de la última ficha 1 si es horizontal y 0 si es vertical.

sen(a) es el sentido hacia donde se están colocando las fichas del lado a; su valor es —1 si es hacia la izquierda +1 si es hacia la derecha.

doble(a) indica si la última ficha colocada es doble. Si=1 no=0.

top(a) es una variable que hace referencia a los extremos derecho e izquierdo *(topder y topiz)*. Puede valer *1(si)* si no se pueden colocar más fichas hacia ese lado, ni verticales ni horizontales. Vale *0(no)* si es puede colocar una ficha horizontal y vale *2(casi)* si se puede colocar una vertical pero no una horizontal. Esta variable nos da una idea sobre cómo tienen que ir colocadas las fichas cuando llegan al extremo del tablero.

Antes de tirar, se comprueba que el jugador tiene alguna ficha para tirar, comparándolas con *n(1)* y *n(3)*. Si esto no ocurre se pasa directamente a la rutina **«paso»**.

A cada ficha se le da un valor atendiendo a los siguientes parámetros:

1.º Se coloca una ficha de doble *(100 puntos)*.

2.º Se comprueba que no cierra el juego, buscando más fichas con ese valor *(1 punto si no cierra el juego)*.

3.º Da 5 puntos más a cada ficha que haya pasado algún jugador. Las fichas de los jugadores que pasan se almacenan en la tabla pasa *(jugador, ficha)*.

4.º Suma 1 punto por cada ficha que tenga el jugador de ese valor.

5.º A cada valor de la ficha se le añade el de los puntos que le componen, de esta forma, se tirarán antes las fichas altas que las bajas para rebajar puntos.

Una vez efectuada la puntuación, se ordena la tabla de valoración, valor *(player, ficha, lado)*, según sus puntos y se pasan a la tabla de los jugadores *(jugada, nivel, parámetro)*.

Los parámetros son:

1 para la ficha.

2 para el lado a colocar 1=izquierdo y 3=derecho.

Ahora se elige la ficha según el nivel.

A nivel más alto se escoge la jugada (nivel, parámetro) más alta.

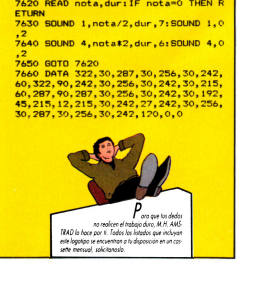
Después de cada tirada se comprueba si el jugador ha terminado y en el caso del jugador 4, si había pasado anteriormente de algún valor de esa ficha; si es así va a la rutina «renuncio» y se finaliza convenientemente *(30 puntos más)*.

Al jugador que acaben se le restan 10 puntos y a los demás se les suma el valor de las fichas. El programa está abundantemente ayudado por instrucciones **«rem»** que ayudan a una mejor comprensión de las rutinas.

La rutina gráfica forma por si sola una unidad independiente ya que es capaz de determinar si una ficha es o no válida.

Los parámetros de entrada son n1 y n2 para el valor de la ficha *(no importa el orden)*. Si el jugador es el 1, 2 o 3 se necesita la variable lado *(1ó 3)* para determinar el lugar a colocarlo; en caso de que el jugador sea el cuarto se le preguntara a qué lado quiere colocarlo, si la posición es ambigua. Nótese que el lado DERECHO SIEMPRE ES EL QUE VA HACIA ARRIBA Y EL IZQUIERDO HACIA ABAJO aunque las posiciones de las fichas en ese momento puedan parecer confusas.

```
5670 PRINT CHR$(23)+CHR$(0)
5680 PLOT -2,-2,0
5690 IF player=4 THEN GOSUB 3830:GO
TO 5720
5700 GOSUB 1970
5710 REM
5720 CALL &BBE4,0,0,0
5730 IF y>350+(col(a)-1)*16 THEN FO
R z=1 TO 3:LOCATE#1,1,1:PRINT#1,CHA
R$(8);y=y-16:y(1)=y:(1)-16:NEXT:GOTO
5730
5740 IF y<112+ABS((col(a)-1)*16) TH
EN FOR z=1 TO 2:LOCATE #1,1,25:PRIN
T#1:y=y+16:x(3)=x(3)+16:NEXT:GOTO 5
740
5750 x(a)=x
5760 y(a)=y
5770 IF n1<>n(a) THEN t=n1:n1=n2:n2
=t
5780 n(a)=n2
5790 IF sen(a)=-1 THEN t=n1:n1=n2:n
2=t
5800 IF col(a)=ver THEN 5940
5810 REM ------------ COLOCA FICHA
HORIZONTAL ------------
5820 REM
5830 LOCATE 1,1:PRINT CHR$(23)+CHR$
(0):
5840 PLOT -2,-2,1
5850 IF sen(a)=1 THEN x=x-64
5860 num=n1
5870 GOSUB 6100
5880 MOVE x+28,y-14:GOSUB 6470
5890 x=x+30:SOUND 2,600,5,7,0,0,2
5900 num=n2
5910 GOSUB 6100
5920 MOVE x,y-14:GOSUB 6470
5930 RETURN
5940 REM ------------ COLOCA FICHA
VERTICAL ----------------
5950 REM
5960 IF a=3 THEN y=y+32
5970 IF (a=1 AND sen(a)=-1) OR (a=3
 AND sen(a)=1) THEN t=n1:n1=n2:n2=t
5980 IF sen(a)=1 THEN x=x-32
5990 REM ---------- AQUI DIBUJA L
AS FICHAS DEL PRINCIPIO --------
6000 PRINT CHR$(23)+CHR$(0)
6010 PLOT -2,-2,1
6020 num=n1+7
6030 GOSUB 6100
6040 MOVE x+14,y-28:GOSUB 6470
6050 y=y-30:SOUND 2,600,5,7,0,0,2
6060 num=n2+7
6070 GOSUB 6100
6080 MOVE x+14,y:GOSUB 6470
6090 RETURN
6100 REM ------------DIBUJO DEL CUA
DRO----------
6110 REM
6120 TAG
6130 MOVE x,y:PRINT CHR$(252)+CHR$(
253);
6140 MOVE x,y-16:PRINT CHR$(254)+CH
R$(255);
6150 RESTORE
6160 READ w
6170 WHILE w<>num
6180 FOR z=1 TO 10
6190 READ w
6200 NEXT z
6210 WEND
6220 MOVE y+6,y-6
6230 FOR n=1 TO 3
6240 FOR c=1 TO 3
6250 READ t
6260 IF t=1 THEN GOSUB 6470
6270 MOVER 8,0
6280 NEXT c
6290 MOVER -24,-8
6300 NEXT b
6310 TAGOFF
6320 RETURN
6330 DATA 0,0,0,0,0,0,0,0,0,0
6340 DATA 1,0,0,0,0,1,0,0,0,0
6350 DATA 2,0,0,1,0,0,0,1,0,0
6360 DATA 3,0,0,1,0,1,0,1,0,0
6370 DATA 4,1,0,1,0,0,1,0,1
6380 DATA 5,1,0,1,0,1,0,1,0,1
6390 DATA 6,1,1,1,0,0,0,1,1,1
6400 DATA 7,0,0,0,0,0,0,0,0,0
6410 DATA 8,0,0,0,0,1,0,0,0,0
6420 DATA 9,1,0,0,0,0,0,0,0,1
6430 DATA 10,1,0,0,0,1,0,0,0,1
6440 DATA 11,1,0,1,0,0,0,1,0,1
6450 DATA 12,1,0,1,0,1,0,1,0,1
6460 DATA 13,1,0,1,1,0,1,1,0,1
6470 REM ------------- DIBUJO DEL
 PUNTO -------------
6480 REM
6490 PLOTR 0,0,1
6500 PLOTR 2,0
6510 PLOTR 0,-2
6520 PLOTR -2,0
6530 MOVER 0,2
6540 RETURN
6550 REM --- Final de la partida y
calcula puntos ---
6560 CLS#5
6570 PRINT#5:PRINT#5:PRINT#5,"    G
ana "jnam$(player)
6580 win=player
6590 GOSUB 7600:REM musica
6600 a=1:sen(a)=-1
6610 FOR player=1 TO 4:puntos(playe
r)=0:NEXT player
6620 CALL &BBE4,0,0,0:PLOT -2,-2,1
6630 sen(a)=-1
6640 REM player 1
6650 player=1:IF player=win THEN pu
ntos(player)=-10:GOTO 6750
6660 FOR ficha=1 TO 7
6670 IF pl$(player,ficha)="" GOTO 6
740
6680 v=574
6690 y=84+ficha*38
6700 n1=VAL(LEFT$(pl$(player,ficha)
,1))
6710 n2=VAL(RIGHT$(pl$(player,ficha
),1))
6720 puntos(player)=puntos(player)+
n1+n2
6730 GOSUB 5810
6740 NEXT ficha
6750 REM player 2
6760 player=2:IF player=win THEN pu
ntos(player)=-10:GOTO 6860
6770 FOR ficha=1 TO 7
6780 IF pl$(player,ficha)="" GOTO 6
850
6790 x=150+ficha*38
6800 y=398
6810 n1=VAL(LEFT$(pl$(player,ficha)
,1))
6820 n2=VAL(RIGHT$(pl$(player,ficha
),1))
6830 puntos(player)=puntos(player)+
n1+n2
6840 GOSUB 6000
6850 NEXT ficha
6860 REM player 3
6870 player=3:IF player=win THEN pu
ntos(player)=-10:GOTO 6970
6880 FOR ficha=1 TO 7
6890 IF pl$(player,ficha)="" GOTO 6
960
6900 x=0
6910 y=84+ficha*38
6920 n1=VAL(LEFT$(pl$(player,ficha)
,1))
6930 n2=VAL(RIGHT$(pl$(player,ficha
),1))
6940 puntos(player)=puntos(player)+
n1+n2
6950 GOSUB 5810
6960 NEXT ficha
6970 REM player 4
6980 player=4:IF player=win THEN pu
ntos(player)=-10:GOTO 7050
6990 FOR ficha=1 TO 7
7000 IF pl$(player,ficha)="" GOTO 7
040
7010 n1=VAL(LEFT$(pl$(player,ficha)
,1))
7020 n2=VAL(RIGHT$(pl$(player,ficha
),1))
7030 puntos(player)=puntos(player)+
n1+n2
7040 NEXT ficha
7050 REM --- Puntuacion ---
7060 CLS#3:PRINT#3
7070 PRINT#3,TAB(4);"NOMBRE" TAB(14
);"PUNTOS" TAB(23);"TOTAL"
7080 PRINT#3
7090 FOR player=1 TO 4
7100 total(player)=total(player)+pu
ntos(player)
7110 nam$=LEFT$(nam$(player)+STRIN
G$(11," "),11)
7120 sco$(player,1)=nam$(player):sc
o$(player,2)=STR$(puntos(player)):s
co$(player,3)=STR$(total(player))
7130 NEXT player
7140 REM ---- Orden la tabla ---
7150 modif=si
7160 WHILE modif=si
7170 modif=no
7180 FOR player=1 TO 3
7190 IF val(sco$(player,3))<=VAL(sc
o$(player+1,3)) GOTO 7240
7200 FOR w=1 TO 3
7210 t$=sco$(player,w):sco$(player,
w)=sco$(player+1,w):sco$(player+1,w
)=t$
7220 NEXT w
7230 modif=si
7240 NEXT player
7250 WEND
7260 FOR player=1 TO 4
7270 IF player=1 THEN PEN#3,0 ELSE
PEN#3,1
7280 PRINT#3,TAB(2);sco$(player,1)
TAB(15);sco$(player,2) TAB(24);sco$
(player,3)
7290 PRINT#3
7300 NEXT player
7310 LOCATE #3,9,19:PRINT#1,CHR$(17
4)+"Otra partida? (S/N)";
7320 a$=INKEY$:IF a$="" GOTO 7320
7330 IF a$="S" OR a$="s" THEN ERASE
 pasa:DIM pasa(4,6):GOSUB 330:win=0
:GOTO 140
7340 IF a$="N" OR a$="n" THEN RUN
7350 GOTO 7320
7360 REM ------ Renuncio ---
7370 CLS#3:LOCATE#3,8,4:PRINT#3,"R
E N U N C I O";
7380 PRINT#3:PRINT#3:PRINT#3
7390 INK 3,6,26
7400 SOUND 1,2000,500,7
7410 SOUND 2,200,500,7
7420 PRINT#3:PRINT#3," Antes pasas
te de ";
7430 IF n1=0 THEN PRINT#3,"BLANCAS"
;
7440 IF n1=1 THEN PRINT#3,"UNOS";
7450 IF n1=2 THEN PRINT#3,"DOSES";
7460 IF n1=3 THEN PRINT#3,"TRESES";
7470 IF n1=4 THEN PRINT#3,"CUATROS"
;
7480 IF n1=5 THEN PRINT#3,"CINCOS";
7490 IF n1=6 THEN PRINT#3,"SEISES";
7500 REM --- Penaliza ---
7510 puntos(4)=30
7520 FOR w=1 TO 5000:NEXT
7530 GOTO 7050
7540 REM ---- Se cierra el juego --
---
7550 REM musica
7560 REM musica
7570 PAPER#1,2:PEN#1,3
7580 LOCATE#1,10,20:PRINT#1,"Juego
 terminado!"
7590 GOTO 6610
7600 REM --- Felicidades ---
7610 RESTORE 7660
7620 READ nota,dur:IF nota=0 THEN R
ETURN
7630 SOUND 1,nota/2,dur,7:SOUND 1,0
,2
7640 SOUND 4,nota#2,dur,6:SOUND 4,0
,2
7650 GOTO 7620
7660 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
```



TU PROGRAMA DE RADIO claro!

Prográmatelo: Sábados tarde de 5 a 7 horas.
En directo y con tu participación.

- Entrevistas a fondo
- Exitos en Soft
- Noticias en Hard
- Concursos

LA COPE A TOPE.
—RADIO POPULAR 54 EMISORAS O.M.—
En Barcelona Radio Miramar

COPE

MAS VELOCIDAD Y MENOS MEMORIA

Programar ya no es lo que era, eso lo sabemos todos. En el principio fueron lenguajes de bajo nivel, después, como todo lo vivo, fueron evolucionando hasta alcanzar la verbosidad anglosajona del Basic, rey sin corona de la informática personal.

No obstante, no sólo de un lenguaje vive un programa, en la compleja labor de crear un programa, hay muchas partes que pueden ser resueltas por el ordenador sin apenas intervención humana, tareas repetitivas o carentes de creatividad que una máquina puede realizar perfectamente.

Entra aquí el concepto clave de **«Herramienta del programador»**, en el sentido de un programa que completa al lenguaje que estemos utilizando para dar luz un programa.

Algunos nombres famosos de herramientas: editores, optimizadores, generadores de programa, depuradores... Los hay por docenas, y todos, tarde o temprano, son necesarios. **AMSTRAD Semanal** es consciente de todo esto, y de qye el programador se encuentra bastante desasistido en este asunto; por ello, hemos decidido abrir una nueva sección que remedie el problema en la medida de lo posible.

Utilidades del programador se estrena con un gran programa: un compresor.

Esta utilidad está pensada para eliminar de cualquier programa todo aquello que al ordenador no le sirve para nada a la hora de ejecutar el programa, como por ejemplo espacios en blanco innecesarios y sentencias REM, La del compresor se traduce en que los programas pasados por su *filtro* ocupan bastante menos memoria y se ejecutan más rápidamente.

Esto permite tener dos versiones del programa: una de ellas, para nosotros, llena de todo lo que sea importante para hacerlo legible y fácil de modificar, y, la otra, especialmente dedicada al Amstrad, que **«pasa»** de tales banalidades.

Compresor es un programa que disminuye la ocupación de memoria de un programa BASIC, eliminando los caracteres que ocupan memoria y son imnecesarios para el correcto funcionamiento del programa. Su utilidad es:

— Ahorra espacio en disco y disminuye el tiempo de carga en cinta.

— Aumenta la velocidad de ejecución *(sólo un poco, no espere milagros)*.

— Permite mayor libertad de escritura. En efecto, cuando se necesita un programa muy largo que utilice tablas de dimensión elevada podemos obtener un MEMORY FULL si hemos llenado el programa de espacios y comentarios. Con compresor podrá escribir libremente y depurar con mayor facilidad, ya que en el momento en que se le agote la memoria bastará una pasada por él y tendrá la memeoria que necesita.

Puede pensarse que el suprimir líneas REM puede hacerse con facilidad a mano, pero un programa largo presenta mucho trabajo y además si existen «Gotos». «gosubs»..., dirigidos a las líneas REM *(lo cual es muy frecuente* deberá buscar esas líneas, editarlas...

Compresor realiza esta función automáticamente y además:

— Elimina todos los espacios no significativos de las líneas de programa.

— Suprime las posibles REM a media línea (también reconoce el apóstrofo).

— Quita las variables de control tras NEXT. Si la next es múltiple la sustituye por tantas NEXT, como sean necesarias.

Tal vez le parezca que NEXT i, j, k ocupa menos que NEXT:NEXT:NEXT; pero el modo que tiene BASIC de codificación interna hace que no sea así. Además el NEXT sin variable se ejecuta con más rapidez porque no necesita comprobar si el nombre coincide con el bucle actual.

Además de esto realiza una unión de líneas, es decir: cuando dos líneas seguidas pueden unirse sin modificar la estructura del programa *(la primera* no contiene un IF y la segunda no es objeto de un GOTO u otra referencia a su *número de línea)* suprime el número de línea y lo sustituye por *(:)*. Este proceso se reitera mientras la longitud de las dos líneas unidas no sea demasiado larga para que el BASIC la admita.

Casos no previstos:

— Si su programa utiliza una rutina de gestión de errores con referencia mediante ERL a números de línea determinados, puede ser que tras la compresión dicho número de línas no exista y la gestión de error no dé el resultado previsto.

— Si el programa utiliza CHAIN MERGE..., DELETE..., puede ocurrir que la versión comprimida le deje alguna línea de más o de menos.

— En fin, si su programa utiliza referencia a números de línea no prevista por el programa, los resultados son imprevisibles.

— Para solucionarlo se ha incluido una opción **«comprimir paso a paso»**, que tras comprimir una línea, antes de intentar unirla a la anterior espera la pulsación de una tecla. Si quiere impedir que el programa trate de llevar a cabo la unión «TAB», en caso contrario basta con pulsar cualquier otra tecla. Lo único que tiene que hacer es revisar el programa antes de comprimir y tomar nota de cuáles son las líneas que interesa dejar aisladas. Después lance el compresor, use esta opción y pulse teclas cada vez que un cursor parpadeante aparezca al final de la última línea del **«Programa comprimido»** en la ventana inferior. Pulse «tab» cada vez que lo necesite, y listo.

Modo de utilización

Cargue el programa compresor. Aparecerá en pantalla un recuadro con unas breves instrucciones de utilización. Pulse una tecla al terminar las instrucciones, aparecerá el mensaje «Espere un momento».

Tras unos segundos que emplea en inicializar variables, el programa pide el nombre del programa que se va a comprimir.

En este momento debe introducir en su **Amstrad** la cinta o disco que contiene el programa que desea comprimir **grabado en formato ASCII**.

La cuestión del formato requiere algún comentario: los manuales de usuario de los **Amstrad** indican simplemente que la orden save admite el parámetro (,A) después del nombre del programa para grabar el formato ASCII. Las diferencias del SAVE normal con éste son:

— Grabado en disco, no se incluye BAS como tipo del fichero.

— El fichero se graba sin cabecera: CP/M y la mayoría de los procesadores de textos podrán acceder a él *(EAMSWORD no; AMSWORD y TASWORD si)* por lo que podrá utilizar las facilidades de éstos para, por ejemplo, cambiar de nombre una variable todas las veces que aparezca.

— El programa se graba tal y como aparece listado, es decir, no se emplea la codificación interna de las palabras clave *(TOKENS)* sino que éstas son delanteras, siendo sencillo leer el programa mediante OPENINN Y LINE IMPUT 9, como hace este programa.

— Basic puede leer el prgrama en este formato, tanto «LOAD» como con «RUM». En realidad lo que hace al encontrar un fichero ASCII con estas órdenes lee líneas del fichero, las codifica y las introduce en memoria como si se hubiese tecleado directamente; excepto si no llevan un número al principio en cuyo caso aparecería el error DIRECT COMMAND FOUND. La salida del compresor está en formato ASCII, según lo que acabamos de explicar puede ser leída desde BASIC.

— No todo son ventajas; el defecto de grabar los programas como texto es que tanto la grabación como la lectura se hace algo más lentamente que en modo normal *(por algo es el modo normal)* y la ocupación de memoria en disco o cinta es un poco mayor.

La razón de que el compresor utilice este formato es la siguiente: el programa puede ser leído con las instrucciones normales de manejo de ficheros en disco o cinta, y la complejidad del programa es menor al no necesitar decodificar los TOKENS. Así el progrma compresor resulta legible, es fácil de depurar, y resulta fácil que cada usuario pueda modificarlo a su gusto si lo desea. Además, si tiene un AMSWORD II podrá usarlo para cambiar los nombres de las variables por otros más cortos o alguna manipulación similar, ya que la salida de COMPRESOR está enfocada hacia el uso de estas facilidades.

Algunas veces puede parecerle que el programa comprimido podría ser más pequeño. En realidad no es tan grande como la memoria que ocupa en cinta o disco. Lo que ocurre es que el número de línea se escribe con STR$, que incluye un espacio delante de él. No me he preocupado de suprimirlo ya que AMSWORD II trabaja mejor con texto de programa gracias a este espacio. Además no se pretende que la salida de Compresor sea la versión de trabajo del programa. Lo más adecuado es una vez obtenida la versión comprimida, cargarla desde BASIC con LOAD y grabarla con SAVE para tener la versión en formato Basic. Por supuesto que las modificaciones mediante procesador de textos deben hacerse antes de esto.

Comprimir un programa

Sigamos con el modo de usar el programa en sí. Una vez colocada la cinta o disco teclee el nombre del prgrama. En su monitor habrán aparecido tres recuadros, correspondientes a otras tantas ventanas. Por la segunda irá apareciendo el listado de su programa: así puede verificar qué Compresor está funcionando correctamente. En la primera, destinada a informes, aparece «Izleyendo línea...» y los mensajes de manejo del cassette.

Una vez terminado el programa, se le pedirá el nombre con el que quiere grabar la versión comprimi-

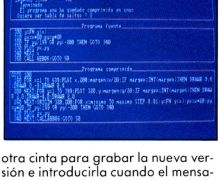
SUBRUTINAS PRINCIPALES

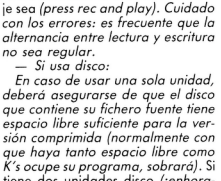
10- 20	Si estamos en un 464, cambiar a cinta.
20- 410	Presentación e instrucciones.
1000- 1220	Iniciación de tablas y variables. Definición de ventanas mediante llamadas a 9000 con los parámetros adecuados.
1230- 1270	Pregunta el nombre del programa, y rechaza la cadena vacía si se está operando con disco.
2000- 2190	Tronco principal del programa, que llama en cada vuelta a las rutinas que necesita.
2200- 2560	Programa terminado. Informes.
3000-33040	Lineas REM.
4000- 4140	Lineas no REM. En la primera vuelta la rutina termina en la línea 4030 gracias al IF vuelta=1 THEN RETURN.
5000- 5130	Inicialización para la segunda vuelta y pide el nombre del fichero de salida.
9000- 9100	Creación y recuadrado de ventana, según los parámetros i, vl, v2.
10000-10080	Incluir número en tabla de saltos.
11000-11220	Buscar las instrucciones con referencia a números de línea, leer éstos y mandarlos a 10000 en la primera vuelta y a 50000 en la segunda (ON vuelta GOSUB 10000, 50000).
12000-12040	Renovar ACTUAL si es necesario.
15000-15550	Buscar número en tabla, si no está el más alto de los menores que él y devolver su índice en anterior.
20000-20040	Leer el siguiente carácter de la línea en curso.
30000-30100	Leer siguiente línea del fichero de entrada.
31000-31050	Comprobar si la línea en curso es una REM.
40000-40110	Rutina de compresión de una línea. Subrutinas de ésta:
40120	Quitar REM y ' del final, si los hay.
40180	Entrecomillado.
40210	Carácter no alfanumérico.
40270	Carácter alfanumérico.
40310	Tratamiento de cada palabra encontrada.
41000	Quitar variables de control en los NEXT.
50000-50090	Cambiar una referencia a número de líneas por su ACTIVO, si está activado *(distinto de cero)*.
60000-60020	Aquí se lleva a efecto el cambio a cinta.
61000-61040	Tratamiento de errores en el nombre de los ficheros.

da. Una vez tecleado éste le preguntará «Comprimir paso a paso *(S/N)*. Pulse una de dichas teclas.

— Si usa cinta:

En este momento deberá rebobinar la cinta de la fuente e introducirla cada vez que el sistema le sugiere *(press play)*. También debe preparar





otra cinta para grabar la nueva versión e introducirla cuando el mensaje sea *(press rec and play). Cuidado con los errores: es frecuente que la alternancia entre lectura y escritura no sea regular.*

— Si usa disco:

En caso de usar una sola unidad, deberá asegurarse de que el disco que contiene su fichero fuente tiene espacio libre suficiente para la versión comprimida (normalmente con que haya tanto espacio libre como K's ocupe su programa, sobrará). Si tiene dos unidades disco *(¡enhorabuena!)* puede poner delante del nombre de la versión comprimida B: y se grabará en la unidad B,

Ahora se repetirá el proceso de la primera vuelta, con la diferencia de que ahora en la ventana inferior irá apareciendo la versión comprimida según se vaya creando. La opción «Paso a paso» está explicada anteriormente y también dentro del propio programa.

Una vez terminado el proceso se cierran los ficheros y el programa pregunta **«¿Quiere ver tabla de saltos?»**. Respondiendo **«s»** obtendrá una lista de las referencias a números de línea que hayan sido cambiadas y el número por el cual han sido cambiado éstos. Puede necesitarlos para alguna modificación posterior.

Con esto concluye el compresor su trabajo. La versión comprimida ya está en cinta o disco dispuesta para ser cargada. Probablemente haya ahorrado más del 10 por 100 de memoria respecto de la fuente. De todos modos le aconsejo que guarde una copia de la fuente: si quiere más adelante ampliar y/o mejorar el programa le será más sencillo hacerlo partiendo de esa versión. La versión comprimida pásela a formato Basic y tendrá la versión de trabajo ocupando el mínimo de memoria y con la máxima rapidez de carga.

Aspecto del análisis de este mismo número después de pasar por el programa compresor.

```
50 CLS:ORIGIN 0,0:WINDOW#1,1,40,25,
25:DEF FN y(x)=x^3/(x^2-4)
100 INPUT#1,"VALOR MINIMO DEL INTER
VALO ",minimo:CLS#1:INPUT#1,"VALOR
MAXIMO DEL INTERVALO ",maximo:CLS#1
:IF minimo<-16 OR maximo>16 OR mini
mo>maximo THEN GOTO 100
160 FOR x=1 TO 639:PLOT x,200:marge
n=x/20:IF margen=INT(margen)THEN DR
AWR 0,4:DRAWR 0,-8:DRAWR 0,4
200 NEXT:FOR y=1 TO 399:PLOT 320,y:
margen=y/20:IF margen=INT(margen)TH
EN DRAWR 2,0:DRAWR-4,0:DRAWR 2,0
260 NEXT:ORIGIN 320,200:FOR x=minim
o TO maximo STEP 0.01y=FN y(x):px=
x*20:py=y*20:IF py>199 OR py<-200 T
HEN GOTO 340
330 PLOT px,py
340 NEXT:CALL &BB06:GOTO 50
```

TABLA DE VARIABLES

Salto	Tabla que almacena las referencias a números de línea.
Punts %	Tabla de índice para ordenar saltos, contiene el subíndice del siguiente salto en orden creciente.
Activo	Almacena los modificaciones que deben realizarse. Su contenido indica: — Si es cero, que no debe efectuarse ningún cambio. — Si es negativo, que el número para cambio se encuentra en el índice opuesto (1). — Si es positivo, la referencia al número de línea debe ser cambiada por dicho número.
Numsal	Dimensión de las tablas anteriores. Se inicializa a 300, pero puede ser aumentado si el usuario lo necesita.
Numlínea	Número de la línea en curso (la que se está leyendo).
Actual	Indice del salto igual o inmediatamente anterior a numlínea en la tabla SALTOS.
Línea$	Contenido de la línea actual sin el número de línea ni los espacios iniciales.
Nuevalins$	Versión comprimida de línea$.
Nueva$	Línea que se está preparando para mandar al nuevo fichero.
Alfanum$	Contiene los números y todas las letras; se usa para comprobar rápidamente si un carácter es alfanumérico mediante INSTR.
Número	Número que se quiere buscar o incluir en a tabla saltos.
Remflag	Si vale uno, la línea en curso es una REM.
Unir	Si toma el valor cero, la línea en curso no se debe unir con la anterior.
Unirsig	Si toma el valor cero, la línea en curso no se debe unir con la siguiente.
J	Carácter que se está leyendo en la línea en curso (2).
Carac$	Contiene el carácter que se acaba de leer de la línea en curso o CRH$ (13) si se ha terminado la línea.
Carácter	Código ASCII de carac$, o CHR$ (13) si se ha encontrado el final de la línea o el comienzo de una zona de comentario *(REM o ')*.
FINTAB	Ultimo índice utilizado en tabla de saltos.
Transfer	Indice de salto dirigido a la última REM que está a la espera de ser cambiado, cero en caso contrario.
Comilla	Si es uno, la parte de línea que se está leyendo corresponde a un entrecomillado.
Palabra$	Nombre de variable, palabra clave del Basic o número que se acaba de leer en la línea en curso.
Go$	Instrucciones que admiten referencia a números de línea; el programa THEN; ELSE; GOTO; GOSUB; RESUME y RESTORE.
Pal	Dimensión de dicha tabla.

***(1)** Ejemplo: Si estamos tratando un GOTO 7000. Supongamos que lo encontramos en el índice 10 de esta forma. Quiere decir que no debe ser cambiado. Si lo encontramos es:*

Indice	*Salto*	*Activo*
10	*7000*	*0*
10	*7010*	*7020*

Quiere decir que debemos cambiar el GOTO 7000 por un GOTO 7020. Si lo que encontramos es:

10	*—7000*	*—14*
14	*7010*	*7020*

El menos significa que no se debe cambiar directamente por ese número, sino que debemos mirar en el subíndice 14 para encontrar el número por el que se debe cambiar. Esto ocurre cuando hay más de un salto que deban ser cambiados al mismo número; en el ejemplo, tanto las referencias al 7000 como al 7010 deben ser cambiadas al 7020. (Es raro que esto ocurra).

(2) Es decir, el número de orden de dicho carácter dentro de LINEA$.

```
1 REM julian albo garcia c/gral fra
nco 55 , 3    monforte de lemos (lu
go)
10 ON ERROR GOTO 60000
20 !DISC
30 ON ERROR GOTO &1000
40 MODE 2
50 vi=2:v2=24
60 GOSUB 9000
100 PRINT "----------
----------COMPRESOR-
----------"
110 PRINT  TAB ( 36);"----------
120 PRINT  TAB ( 35);"VERSION 1.1
130 PRINT  TAB ( 35);"----------
140 PRINT  TAB ( 32);"Por Julian Al
bo.
150 PRINT " Instrucciones:
160 PRINT "     Este programa compr
ime programas  BASIC  LOCOMOTIVE
para  disminuir  su";
170 PRINT "ocupacion  de  memoria,
 eliminando  todos  los  comentario
s  y  los  espacios";
180 PRINT "innecesarios, y uniendo
las lineas sucesivas siempre  que
puede  hacerlo  sin";
190 PRINT "alterar  la estructura de
l programa. Este proceso se puede
 controlar  mediante";
200 PRINT "la opcion paso a paso.
210 PRINT "     La  cuenta debera es
tar grabada en formato ASCII; la ve
rsion  comprimida
220 PRINT "se grabara en el mismo f
ormato. Si la fuente no contiene er
rores,  la  version";
230 PRINT "comprimida funciona sin
problemas. Depure el programa antes
 de comprimirlo.
240 PRINT "     El programa funcion
a en dos vueltas:
250 PRINT "     La primera lee el p
rograma fuente buscando todos  los
 numeros  de  linea";
260 PRINT "destino de un salto, par
a poder mantener la estructura del
programa.
270 PRINT "     La segunda vuelve a
 leer el programa, y si va comprimie
ndo linea por linea";
280 PRINT "escribiendo el resultado
 en el fichero de salida.
290 PRINT "     Si el programa cont
iene DELETE debera asegurarse de qu
e la primera  linea";
300 PRINT "que vaya a ser borrada n
o se una a la anterior, y lo mismo
 la primera  que  no";
310 PRINT "deba serlo; para ello us
e la opcion paso a paso.
320 PRINT "     Pulse una tecla para c
ontinuar
330 WHILE INKEY$="":WEND:CLS
340 PRINT "     La opcion paso a pa
so, en la segunda vuelta, despues
 de  leer  la  linea";
350 PRINT "espera la pulsacion de u
na tecla. Si es la <TAB> impide que
 la linea se una  a";
360 PRINT "la anterior; si es cualq
uier otra, lo permite.
370 PRINT "     Si no la escoge, la
 compresion se llevara a cabo autom
aticamente.
380 PRINT " Pulse una tecla para c
ontinuar
390 WHILE INKEY$="":WEND:CLS
410 PRINT " Espera un momento."
1000 pal=5:unumsal=300
1010 DIM gos%(pal),salto(unumsal),pu
nt%  (numsal),activo(unumsal)
1020 DEF FN char$ (indice,lin$)=MID
$(lin$,indice,1)
1030 FOR i=0 TO pal:READ gos%(i):NE
XT
1040 DATA "THEN ","ELSE ","GOTO ","
GOSUB ","RESTORE ","RESUME "
1050 salto(0)=0:punt%(0)=numsal:fin
tab=1
1060 FOR i=1 TO numsal
1070 salto(i)=100000:punt%(i)=numsa
l
1080 NEXT i
1090 alfanum$="abcdefghijklmnopqrst
uvwxyz"
1100 alfanum$=alfanum$+UPPER$(alfan
um$)+"0123456789"
1110 cur$=CHR$(143)+CHR$(8)+" "+CHR
$(8)
1120 MODE 2
1130 FOR i= 0 TO 2
1140 READ v1,v2
1150 GOSUB 9000 '  Definir ventanas
1160 NEXT
1170 DATA 2,5
1180 DATA 3,13
1190 DATA 16,24
1200 LOCATE #3,33,1;PRINT #3, "Info
rmes"
1210 LOCATE #3,30,7;PRINT #3 ,"Prog
rama fuente"
1220 LOCATE #3,30,15;PRINT #3, "Pro
grama comprimido"
1230 PRINT "  Programa fuente ";
1240 INPUT prog$
1250 WHILE tape=0 AND prog$=""
1260 PRINT CHR$(7);" No es valido"
:INPUT "  Programa fuente ": prog$
1270 WEND
2000 REM ----------Bucle p
rincipal----------
2010 FOR vuelta=1 TO 2
2020 IF vuelta=2 THEN GOSUB 5000
' Inicializacion segunda vuelta
2030 CLS:PRINT :PRINT "  Vuelta "
vuelta
2040 OPENIN prog$
2050 LOCATE 1,3:PRINT " Leyendo lin
ea :"
2060 WHILE NOT EOF
2080 GOSUB 30000:PRINT #1,numlinea;
USING "&";linea$  '  Lee linea
2090 LOCATE 20,3:PRINT USING "#####
";numlinea
2100 GOSUB 31000  '  Comprobar si e
s rem
2110 GOSUB 12000  '  Actualizar
2120 goin=( salto(actual) = numline
a )
2130 IF pausa$="S" THEN a$="":WHILE
 a$="":a$=INKEY$:PRINT #2, cur$;:WE
ND:IF a$=CHR$(9) THEN unirsig=0
2140 ON remflag+1 GOSUB 4000 , 3000
2150 WEND
2160 CLOSEIN
2170 NEXT vuelta
2180 PRINT #9,nueva$
2190 CLOSEOUT
2200 PRINT CHR$(7);"  Terminado"
2210 PRINT "  El programa "+prog$+
" ha quedado comprimido en "+nuevpr
og$
2220 PRINT "  Quiere ver tabla de s
altos ? "i
2230 WHILE INSTR(" SN",UPPER$(a$))<
2:a$="":a$=INKEY$:PRINT cur$+a$;:WEND
2240 MODE 2:IF UPPER$(a$)="N" THEN
END
2250 i=0:v1=2:v2=24
2260 GOSUB 9000:CLS
2270 indice=0
2280 WHILE salto(indice)<100000
2290 val=activo(indice)
2300 IF salto<>0 THEN GOSUB 2500
2310 indice=punt%(indice)
2320 WEND
2330 PRINT " Pulse una tecla"
2340 WHILE INKEY$="":PRINT cur$;:WE
ND
2350 MODE 2:END
2500 PRINT salto(indice);" se cambi
a por ";
2510 ind=indice
2520 WHILE activo(ind)<0
2530 ind= -activo(ind)
2540 WEND
2550 PRINT activo(ind)
2560 RETURN
3000 REM ----------line
as rem----------
3010 IF goin=0 THEN RETURN
3020 IF transfer<>0 THEN activo(tra
nsfer)= -actual
3030 transfer=actual
3040 RETURN
4000 REM ----------lineas
 no rem----------
4010 IF transfer<>0 THEN activo(tra
nsfer)=numlinea:numero=numlinea:GOS
UB 10000:transfer=0
4020 unir=unirsig:unirsig=1:GOSUB 1
1000 '  Busca saltos y gestionalos
4030 IF vuelta=1 THEN RETURN
4040 GOSUB 40000  '  Comprimir la 1
linea
4050 IF goin<>0 THEN unir=0
4060 longitud= LEN (nueva$)+LEN (nu
eva$)
4070 IF longitud > 245 THEN unir=0
4080 IF unir=1 THEN nueva$=nueva$+"
:"+nuevalin$:PRINT #2,USING "&";":"
+nuevalin$:RETURN
4090 REM ----------linea al fiche
ro de salida----------
4100 PRINT #2
4110 PRINT #9,nueva$
4120 nueva$=STR$(numlinea)+" "+nuev
alin$:PRINT #2,USING "&";nueva$;
4130 unir=1
4140 RETURN
5000 REM----------Inicializ
acion segunda vuelta----------
5010 INPUT "  Nombre para el fiche
ro de salida ";nuevprog$
5020 WHILE nuevprog$="" AND tape=0
5030 PRINT CHR$(7);"  No es valido
 ":INPUT "  Nombre ";nuevprog$
5040 WEND
5050 PRINT "  Comprimir paso a paso
 ( s / n ) ?";
5060 WHILE INSTR(" SN",pausa$)<2:pa
usa$=INKEY$:WEND
5070 PRINT cur$+pausa$:pausa$=UPPE
R$(pausa$)
5080 WEND:PRINT" "
5090 unir=0:actual=0:unirsig=0
5100 IF tape=1 THEN PRINT "  Cuando
 aparezca PRESS PLAY ... introduzc
a la cinta fuente":PRINT :PRINT "
Cuando sea, PRESS REC AND PLAY ...
 la de destino."
5110 OPENOUT nuevprog$:PRINT " Pul
se una tecla para comenzar."
5120 WHILE INKEY$="":PRINT cur$;:WE
ND:FOR i=0 TO 2:CLS #i:NEXT
5130 RETURN
9000 '----------
Ventanas----------
---
9010 PEN #i,1:PAPER #i,0
9020 WINDOW #i,2,79,v1,v2
9030 e1=16*(26-v1)+3
9040 e2=16*(25-v2)-3
9050 MOVE 2,e1:DRAW 639,e2
9060 DRAW 639,e1:MOVE 638,e2
9070 DRAW 638,e1
9080 DRAW 3,e1:DRAW 3,e2
9090 MOVE 2,e2:DRAW 2,e1
9100 RETURN
10000 REM ----------poner sal
to en tabla----------
10010  ' El numero a incluir entra
 en numero
10020 GOSUB 15000
10030 IF salto ( siguiente )= numer
o THEN RETURN
10040 salto ( fintab )= numero
10050 punt% ( fintab )= siguiente
10060 punt% ( anterior )= fintab
10070 fintab = fintab + 1
10080 RETURN
11000 REM----------  Busca de sal
tos ----------
11010 FOR i = 0 TO pal
11020   a% = i
11030   i1 = INSTR ( a%,linea$,gos%
(i))
11040   WHILE i1<> 0
11050     GOSUB 11100
11060     a%= i1 + LEN ( gos%(i));i
1 = INSTR ( a%,linea$,gos%(i))
11070   WEND
11080 NEXT i
11090 RETURN
11100 REM Leer los numeros de linea
.
```

```
11110 j=i+ LEN (gos$(i))-2
11120 GOSUB 20000:WHILE caracter<>A
SC("+") AND caracter<>13
11130   WHILE caracter=32 OR caract
er=44
11140     GOSUB 20000
11150   WEND
11160 i=j-1:  num$=""
11170   WHILE INSTR ("0123456789",c
arac$) <> 0
11180     num$=num$+carac$:GOSUB 20
000
11190   WEND
11200   IF num$ <> "" THEN numero=
VAL(num$) :ON vuelta GOSUB 10000,50
000 ELSE RETURN
11210 WEND
11220 RETURN
12000 REM ---------------- Actualiza
r ----------------
12010 WHILE salto(punt%(actual)) <
> numlinea
12020 actual=punt%(actual)
12030 WEND
12040 RETURN
15000 REM --------------------busca
r numero en saltos----------------
---
15010 anterior = 0 : siguiente=punt
%(anterior)
15020 WHILE salto(siguiente) < nume
ro
15030 anterior=siguiente : siguient
e= punt% ( anterior )
15040 WEND
15050 RETURN
20000 REM---------------------- sigui
ente caracter----------------------
20010 j=j+1
20020 IF j>LEN (linea$) THEN carac$
=CHR$(13) ELSE carac$=FN char$ (j,l
inea$)
20030 caracter=ASC(carac$)
20040 RETURN
30000 REM---------------------------
Leer linea-------------------------
30010 LINE INPUT#9,linea$
30020 j=0:c$="0"
30030 WHILE INSTR(" 0123456789",c$)
<>0
30040 j=j+1:c$=MID$(linea$,j,1)
30050 WEND
30060 numlinea=VAL (LEFT$(linea$,j-
1))
30070 linea$=RIGHT$(linea$,LEN (lin
ea$)-j+1)
30080 IF numlinea=0 THEN ERROR 21
30090 IF numlinea>65535 THEN ERROR
6
30100 RETURN
31000 REM---------------------- Com
probar si es rem------------------
---
31010 IF LEN(linea$)<4 THEN linea$=
linea$+SPACE$(4)
31020 IF LEFT$(linea$,1)="'" THEN r
emflag=1:RETURN
31030 IF LEFT$(linea$,3)<>"REM" THE
N remflag=0:RETURN
31040 IF INSTR(alfanum$,MID$(linea$
,4,1))<>0 THEN remflag=0:ELSE remfl
ag=1
31050 RETURN
40000 REM---------------------- comp
rimir la linea---------------------
---
40010 comilla =0:alfanum=0: palabra
$="":finpalabra=0:unirsig=1
40020 nuevalin$="":j=0
40030 caracter=1
40040 WHILE caracter<>13 '  bucle p
rincipal
40050   GOSUB 20000 ' Siguiente ca
racter
40060   IF comilla=1 THEN GOSUB 40
180 ELSE IF INSTR(alfanum$,carac$)=
0 THEN GOSUB 40210 ELSE GOSUB 40270
40070   IF caracter=13 THEN carac$
=""
40080   nuevalin$=nuevalin$+carac$
40090 WEND ' Fin del bucle principa
l
40100 IF comilla=0 THEN GOSUB 40120
:ELSE nuevalin$=nuevalin$+CHR$(34)
40110 RETURN
40120 REM----------Quitar rem, '
y : del final--------------
40130 IF RIGHT$(nuevalin$,3)="REM"
THEN nuevalin$=LEFT$(nuevalin$,LEN
(nuevalin$)-3)
40140 WHILE INSTR(":",RIGHT$(nueva
lin$,1))<>0
40150   nuevalin$=LEFT$(nuevalin$,
LEN(nuevalin$)-1)
40160 WEND
40170 RETURN
40180 REM entrecomillado
40190 IF caracter=34 THEN comilla=0
40200 RETURN
40210 REM  Caracter no alfanumerico
40220 IF LEN(palabra$)>0 THEN GOSUB
 40310 ' Fin de palabra
40230 IF carac$="" THEN caracter=1
3
40240 IF caracter=32 THEN carac$=""
 ELSE finpalabra=0
40250 IF caracter=34 THEN comilla=1
40260 RETURN
40270 REM  Caracter alfanumerico
40280 IF finpalabra=1 THEN nuevalin
$=nuevalin$+SPACE$(1)
40290 palabra$=palabra$+carac$:finp
alabra=0
40300 RETURN
40310 ' Final de palabra
40320 IF palabra$="IF" THEN unirsig
=0
40330 IF palabra$="REM" THEN caract
er=13
40340 IF palabra$="NEXT" THEN GOSUB
 41000
40350 finpalabra=1:palabra$=""
40360 RETURN
41000 REM     Quitar variables en l
os next
41010 numvar=1
41020 WHILE INSTR(":"+CHR$(13),cara
c$) =0
41025 GOSUB 20000
41030 IF carac$="," THEN numvar=num
var+1
41040 WEND
41050 FOR n=2 TO nuevar
41060 nuevalin$=nuevalin$+":NEXT"
41070 NEXT n
41080 finpalabra=0
41090 RETURN
50000 REM ----------cambiar salto-
----------
50010 GOSUB 15000
50020 act=siguiente
50030 IF numero<>salto(act) THEN RE
TURN
50040 IF activo(act)=0 THEN RETURN
50050 WHILE activo(act) < 0
50060 act= -activo(act)
50070 WEND
50080 linea$=LEFT$(linea$,j1)+STR$(
activo(act))+RIGHT$(linea$,LEN(line
a$)-j+1)
50090 RETURN
60000 ' Cambiar a cinta
60010 tape=1
60020 RESUME NEXT
61000 '----------------------Errores
61010 IF ERR=32  THEN PRINT "  Nomb
re incorrecto o error de acceso a d
isco.":IF ERL=2040 THEN RESUME 1230
 ELSE IF ERL=5110  THEN RESUME 5010
61020 IF ERR=25 THEN PRINT CHR$(11)
;" Fichero no valido.":tl:CLOSEIN:RE
SUME 1230
61030 ON ERROR GOTO 0
61040 END
```

Para que tus dedos no realicen el trabajo duro, M.P. AMS TRAD lo hace por ti. Todos los listados que incluyan este logotipo se encuentran a tu disposición en un cassette mensual, solicítanoslo.

RÓTULOS GIRATORIOS

El programa permite enviar a través de la pantalla mensajes que van de izquierda a derecha, como si se tratase de un rótulo giratorio.

Puede usarse en títulos, o en medio de un programa para enviar un mensaje cualquiera.

Funcionamiento: es muy sencillo, consiste en crear una matriz, en la cual se van almacenando las partes del mensaje que va a aparecer en la pantalla, para luego imprimir sucesivamente los diferentes elementos de la matriz, creando ilusión de movimiento.

Variables:

a$= Mensaje que se quiere enviar.

b$= Matriz en la que se almacenan las distintas partes del mensaje.

Nota: el programa ha sido realizado en el modo 40 columnas *(modo 1)* pero es muy sencillo hacer que funcione en cualquiera de los otros dos modos, con tal de cambiar en las líneas 60,70,80,110 y 170 los «40» que aparecen por «80» *(modo 2)* o «20» *(modo 0)*.

Rafael Collantes

```
10 REM RAFAEL COLLANTES 1986
20 CLS:CLEAR
30 INK 0,13:INK 1,0:BORDER 13
40 a$="PULSA UNA TECLA PARA CONTINU
AR"
50 CLS:LOCATE 6,9:PRINT "TITULOS (R
afael Collantes 1986)"
60 DIM b$(LEN(a$)+40)
70 FOR i = 1 TO 40+LEN(a$)
80 FOR j = 1 TO 40-i
90 b$(i)=b$(i)+" "
100 NEXT
110 IF i<40 THEN k=0 ELSE k=i-40
120 b$(i)=b$(i)+MID$(a$,1+k,i)
130 FOR j= 1 TO 1-LEN(a$)
140 b$(i)=b$(i)+" "
150 NEXT
160 NEXT
170 FOR i = 1 TO 40 + LEN(a$)
180 LOCATE 1,9:PRINT b$(i)
190 IF INKEY$<>"" THEN 220
200 NEXT
210 GOTO 170
220 CLS:CLEAR
```

Mercado común

Con el objeto de fomentar las relaciones entre los usuarios de AMSTRAD, *MERCADO COMUN* te ofrece sus páginas para publicar los pequeños anuncios que relacionados con el ordenador y su mundo se ajusten al formato indicado a continuación.

En *MERCADO COMUN* tienen cabida, anuncios de ventas, compras, clubs de usuarios de AMSTRAD, programadores, y en general cualquier clase de anuncio que pueda servir de utilidad a nuestros lectores.

Envíanos tu anuncio mecanografiado a: **HOBBY PRESS, S.A.**
AMSTRAD SEMANAL.
Apartado de correos 54.062
28080 MADRID
¡ABSTENERSE PIRATAS!

Corta y pega este cupón en la casilla correspondiente de la página 16 del número 31 de **AMSTRAD Semanal, una vez completada la página**, envíanosla junto con tus datos. ¡SUERTE!

Urge vender ordenador Oric-1 (48 K) con garantía. Se encuentra en perfecto estado. Regalo manual de uso en inglés y en castellano, cables de conexión TV y cassette, así como alimentación y juegos de diversos tipos. Todo ello por tan sólo 20.000 ptas. Interesados llamar al (93) 239 15 22. Preguntar por Cristina a partir de las 3 de la tarde. Barcelona.

Vendo CPC 464 con monitor en color, por 70.000 ptas. o también lo cambiaría por CPC 6128 abonando 40.000 ptas. de diferencia, junto con el ordenador doy: garantía vigente, manuales del usuario, más los libros 40 juegos sensacionales, curso automático de Basic y 10 juegos, entre ellos: Exploiding Fist, Knight Lore y Everyone a Wally's. José. Valencia. Tel. (96) 363 60 86.

Vendo videojuego Atari CX-2600 AS, comprado hace 5 meses, con mando, 3 cartuchos, todo por 1.200 ptas. Tel. 475 22 92 de Madrid. Preguntar por José de 10 a 14 h.

Vendo Amstrad CPC 464, con monitor de color, con él incluido 8 cintas comerciales y 2 manuales en castellano, se halla en perfectas condiciones con 3 meses de uso; está en garantía y lo vendo por 75.000 ptas. Interesados llamar al tel. 693 05 70 de Madrid, preguntando por Miguel o Ana.

Vendo Amstrad 464 monitor en F. verde, con manuales en castellano, joystick y unos 25 programas (Matchday, Decathlon, Bruce Lee...) comprado en octubre de 1985 en el Corte Inglés. Todo por 60.000 ptas. Javier Muñoz Barrena. Avda. San Fco. Javier, s/n «Urbis» bloque 6, piso 3.°, puerta 2.ª. Tel. (954) 65 51 55. Sevilla 41005.

Amstrad CPC 464. Monitor color. Nuevo. Regalo 14 programas. Baseball, Raid, Exploding, Knight lore, Alien 8, Macht day, Procesador, etc. 75000. Luis. Comidas. 273 64 27.

Compraría Amstrad 464, o cambiaría amplificador de 100 wat. por cualquier ordenador. Interesados llamar al tel. 25 29 11. de Córdoba. Juan.

Cambiaría por Dun Darach o el One On One, cualquiera de estos míos: Hacker, Millonaria, Radid Over Moscow, Ghstbusters o Snooker. Escribir a Fernando Díez Flores. C/ Santa Ana, 38, 3.° izq. 24003 León.

Quisiera establecer contacto con usuarios de **Amstrad** en Jaén para listados, programas etc. Preguntar por Ismael de 9 de la tarde hasta las 10. Tel. 22 24 34 o bien escribiendo a: Avda. Madrid, 60 a, 5.° A. Jaén.

Ofites Informática

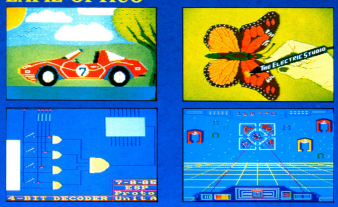
Presenta:
el lápiz al que gusta decir SI
mientras nuestros competidores dicen no
UNICO PARA AMSTRAD, CON PRECISION PIXEL

FUNCIONES	ESP	dk'tronics	OTROS
UNICO MENU DE PANTALLA	SI	NO	
ARRASTRE OBJETOS PANTALLA	SI	NO	
TRASLADO OBJETOS PANTALLA	SI	NO	
TRASLADO DE CURSOR	SI	NO	
CAJAS ELASTICAS	SI	SI	
LINEA ELASTICA	SI	SI	
TRIANGULO ELASTICO	SI	NO	
ELIPSE ELASTICO	SI	NO	
DIAMANTE ELASTICO	SI	NO	
POLIGONO ELASTICO	SI	NO	
HEXAGONO ELASTICO	SI	NO	
OCTOGONO ELASTICO	SI	NO	
CUBO ELASTICO	SI	NO	
PIRAMIDE ELASTICA	SI	NO	
CIRCUNFERENCIAS	SI	SI	
CIRCULOS RELLENOS	SI	NO	
CAJAS RELLENAS	SI	NO	
ELIPSES RELLENAS	SI	NO	
CUÑAS	SI	NO	
SIMULADOR DE CORTES	SI	NO	
DISEÑO DE ZOOM	SI	SI	
IMAGEN ESPEJO E INVERTIDA	SI	NO	
FONDO DE REFERENCIA	SI	NO	
REJILLA DE FONDO	SI	NO	
OPCION DISPLAY X, Y	SI	NO	
RELLENADO CON COLOR	SI	SI	
LAVADO DE COLOR	SI	NO	
VOLCADO PANTALLA RESIDENTE	SI	NO	
DIBUJO DE BORDES EN 3 D	SI	NO	
TEXTO	SI	SI	
9 TAMAÑOS DE BROCHA	SI	NO	
18 TOBERAS MOSTRADORAS	SI	NO	
4 MEZCLAS BASICAS	SI	NO	
VARIADOR DE MEZCLAS	SI	NO	
SOMBREADO DE MEZCLAS XOR	SI	NO	
FICHERO ICONOS RESIDENTES	SI	NO	
FICHERO RELLENOS RESIDENTES	SI	NO	
26 COLORES DE PAPEL	SI	NO	
PALETA DE 15 TONOS DE COLOR	SI	NO	
POSICIONAMIENTO DE PUNTO	SI	SI	
RAYOS DESDE UN PUNTO FIJO	SI	NO	
DIBUJO REFLEJADO (ESPEJO)	SI	NO	
FUNCION HOME	SI	NO	
CONTROL DESDE TECLADO	SI	SI	
CONTROL CON JOYSTICK	SI	NO	
DISPONIBLES MODOS 1 Y 2	SI	?	

Compare con otros lápices

DEBIDO A LA FALTA DE ESPACIO NO PODEMOS LISTAR LAS OTRAS 40 FUNCIONES MAS QUE NUESTRO LAPIZ ES CAPAZ DE HACER.

ESTOS SON ALGUNOS EJEMPLOS DE LOS GRAFICOS QUE VD. PODRA REALIZAR CON NUESTRO LAPIZ OPTICO



DISPONIBLE PARA:

CPC 464 CASSETTE	4.900 Ptas.
CPC 464-664 DISCO	6.900 Ptas.
CPC 6128 DISCO	6.900 Ptas.

(IVA no incluido)

CONDICIONES ESPECIALES PARA DISTRIBUIDORES

DE VENTA EN LOS MEJORES COMERCIOS DE INFORMATICA

Si Vd. tiene alguna dificultad para obtener el lápiz óptico, puede dirigirse a:

Ofites Informática
Avda. Isabel II, 16 -8º
Tels. 455544 - 455533
Télex 36698
20011 SAN SEBASTIAN

GP-50 •	La pequeña 40 cps. Papel normal con interface paralelo, serial y Spectrum	17.990 ptas.
GP-700 *	La de color 50 cps. 7 colores. 80 columnas. Tracción y fricción. Papel de 10 pulgadas	64.990 ptas.
SP-1.000 *	La programable 100 cps.24 cps en alta calidad 96 cart. programables en RAM. Introductor hoja a hoja.♦	64.990 ptas.
SP-1.000AS	La programable 100 cps.24 cps en alta calidad con interface RS-232. Introductor hoja a hoja.♦	59.900 ptas.
MP-1.300AI	La polivalente 300 cps, 60 cps en alta calidad, interface paralelo y RS-232. Introductor hoja a hoja.♦&	119.900 ptas.
BP-5.200 *	La de oficina 200 cps, 106 en alta calidad.Buffer 4K.Carro de 15".Tracción y fricción.♦	199.900 ptas.
BP-5.420 *	La más rápida 420 cps. 106 cps en alta calidad. Buffer de 18K. Paralelo y RS-232.♦	339.900 ptas.

Interfaces: Serie RS-232C, Spectrum, IBM, COMMODORE, MSX, QL, Apple Macintosh, HP-IB
♦ Introductor automático de documentos opcional.
& Kit de color opcional.
Nota: I.V.A. 12%, no incluido en los precios arriba indicados

* con interface paralelo
• con interface Spectrum

Avda. Blasco Ibáñez, 116
Tel. (96) 372.88.89
Telex 62220 - 46022 VALENCIA

Muntaner, 60-2.º-4.ª
Tel. (93) 323.32.19
08011 BARCELONA

Agustín de Foxá, 25-3.º-A
Tels. (91) 733.57.00-733.56.50
28036 MADRID

DIRAC