AÑO I N.º 8

MICROHOBBY AMSTRAD semanal

REVISTA INDEPENDIENTE PARA USUARIOS DE ORDENADORES AMSTRAD

AÑO I N.º 8

150 Ptas.

Canarias 160 ptas.

GRAFICOS VIVOS A TODO COLOR

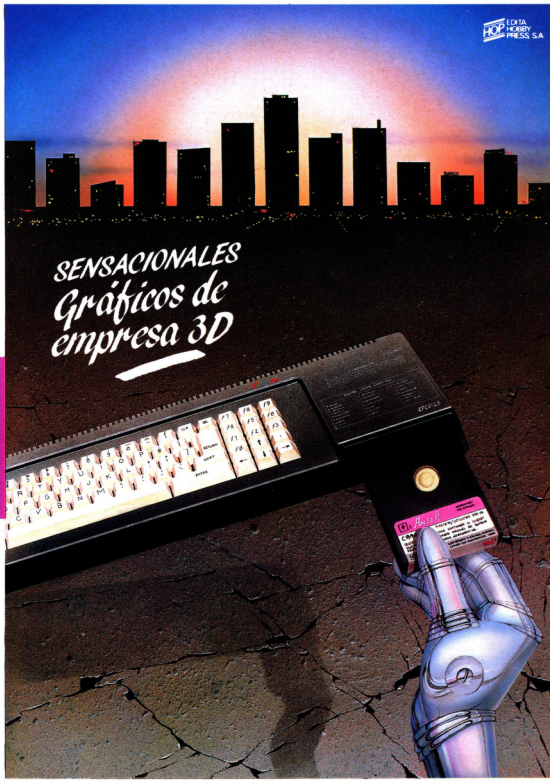
INFUNDE INTELIGENCIA A TUS PROGRAMAS MEDIANTE LA TOMA DE DECISIONES

AMSAMBLADOR: UN COMPLETISIMO ENSAMBLADOR PARA LOS ENTUSIASTAS DEL CODIGO MAQUINA

GENERADOR DE CLAVES SECRETAS

SOFTWARE

007 ATACA DE NUEVO: A VIEW TO A KILL



MICROHOBBY
AMSTRAD
Semanal
REVISTA INDEPENDIENTE PARA USUARIOS DE ORDENADORES AMSTRAD

LE OFRECE AHORA SUS PROGRAMAS YA GRABADOS, PARA QUE VD. NO TENGA QUE TECLEARLOS

Todos los programadores y aficionados a la microinformática sabemos lo tedioso y propenso a errores que resulta el teclear un listado de un programa. Para facilitar tu labor al máximo y que no tengas que estar horas sobre el teclado de tu ordenador tratando de descifrar incomprensibles mensajes de error, ***AMSTRAD SEMANAL*** te ofrece cada mes los programas publicados de los cuatro números correspondientes en una cinta de cassette, sólo por ***675 ptas.*** (*sin más gastos por envío*).

Programas incluidos en la cinta número 1	
Título	**Revista número**
EASYDRAW	1
EGGBLITZ	2
CODIGO SECRETO	2
VENTANAS	2
BIORRITMOS	3
MAD ADDER	3
HEXER	3
CHARGEN	4
PROGRAMACION	4

Programas incluidos en la cinta número 2	
Título	**Revista número**
GRAFICOS	8
MUSICA	6
TRON	6
ENSAMBLADOR	8
HEXERL	8
TOOLKIT	8
PRIMEROS PASOS	7
INCOGNITON	7
MONITOR	5
ANALISIS	5-8
CEDRIC	8
ANIMACION1	7
ANIMACION2	8
SMALEY	7

Todos los programas de nuestras cintas se encuentran desprotegidos, con el objeto de facilitar su copia en disco y la revisión de los listados.

Envíanos con la menor demora posible, el cupón correspondiente.

Director Editorial
José I. Gómez-Centurión
Director Ejecutivo
Víctor Prieto
Subdirector
José María Díaz
Redactora Jefe
Marta García
Diseño
José Flores
Colaboradores
Francisco Portalo
Pedro Sudón
Miguel Sepúlveda
Francisco Martín
Jesús Alonso
Pedro S. Pérez
Amalio Gómez
Juan J. Martínez
Secretaría Redacción
Carmen Santamaría
Fotografía
Carlos Candel
Javier Martínez
Portada
Manuel Barco
Ilustradores
J. Igual, J. Pons, F. L. Frontán, J. Septien, Pejo, J. J. Mora, Luigi Pérez

Edita
HOBBY PRESS S.A.

Presidente
María Andrino
Consejero Delegado
José I. Gómez-Centurión

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

Secretaría de Dirección
Marisa Cogorro

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

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

Dto. Circulación
Carlos Peropadre

Distribución
Coedis, S. A. Valencia, 245
Barcelona

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

MICROHOBBY
AMSTRAD
sumario

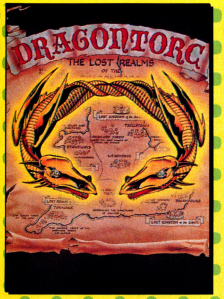
Año I • Número 8 • 22 al 28 de Octubre de 1985
150 ptas. (sobretasa Canarias, 10 ptas.)

¡NUEVO!

SIEMPRE LOS PRIMEROS EN TENER LO ULTIMO

circulo de soft
MICROAMIGO S.A.

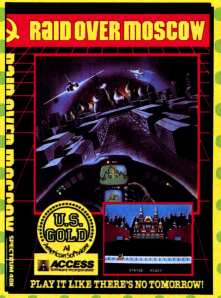
P.º de la Castellana, 268, 3.º C. 28046-MADRID.
Tel.: (91) 733 25 00



DRAGONTORC

Cerca de 200 pantallas con miles de objetos diferentes y más de cien personajes con animación en tres dimensiones, hacen que de este juego la revista inglesa Crash Micro haya llegado a decir «Dragontorc es lo mejor que hemos visto en juegos de acción y aventura».

P.V.P.: 2.300 ptas.
Precio Socios C. de Soft: 2.070 ptas.



RAID OVER MOSCOW

Defiende a USA y Canadá del ataque nuclear que ha lanzado Rusia contra ellos. Con tu escuadrilla habrás de hacer un viaje lleno de peligros hasta llegar al mismísimo Kremlin y destruir las bases de lanzamiento soviéticas. Gráficos y acción sensacionales.

P.V.P.: 2.300 ptas.
Precio Socios C. de Soft: 2.070 ptas.

SOUTHERN BELLE

Siente la emoción de conducir una locomotora de vapor, através de un maravilloso recorrido desde Londres a Brigthon, manejando la caldera, el silbato, atravesando túneles, etc. Estamos ante uno de los juegos más brillantes y originales aparecidos para ordenador.

P.V.P.: 2.300 ptas.
Precio Socios C. de Soft: 2.070 ptas.

¡¡¡...Y LOS TRES PROGRAMAS POR SOLO 5.500 PTAS.!!!

¡HAZTE HOY MISMO SOCIO DEL CIRCULO DE SOFT! Además de poder adquirir tus programas al mejor precio, recibirás información de forma periódica y gratuita, del mejor software que aparezca en el mercado.

¿QUE HAY QUE HACER PARA SER SOCIO DEL CIRCULO DE SOFT? Así de fácil: envíanos por correo tu nombre, dirección y modelo de ordenador, o bien, pide por teléfono o por correo tu primer programa. ¡Y entrarás a formar parte del CIRCULO DE SOFT de forma inmediata!

☐ SÍ, quiero ser SOCIO desde hoy mismo del CIRCULO DE SOFT y recibir periódicamente información de novedades de software, así como beneficiarme desde hoy mismo de los precios reducidos reservados a los SOCIOS y de sus Ofertas Especiales. El ser SOCIO no me obliga a compra alguna.

Si prefieres formalizar tu compra por teléfono puedes hacerlo llamando al (91) 733 25 00. ¡¡NO SE COBRAN LOS GASTOS DE ENVIO POR CORREO!!

TITULO	P.V.P.	ORDENADOR

☐ Contrarreembolso ☐ Giro Postal ☐ Talón adjunto a Microamigo, S.A. ☐ Tarjeta VISA n.º _____ Fecha caducidad _____

Nombre _____ Apellidos _____ Edad _____

Domicilio _____ Teléfono _____

Localidad _____ C.P. _____ Provincia _____

Primera plana

DECVILLE, CIUDAD SIN LEY

DEC *(Digital Equipment Corporation)* presentó en el sótano del palacio de festivales de Cannes una ciudad, llamada ingeniosamente Decville, la cual, aparte de estar en un sótano, presenta la característica de estar completamente informatizada, hasta los detalles más nimios.

Para los aficionados a la informática y a la comodidad, un increíble sueño hecho realidad, y una muestra de cómo pueden ser nuestras ciudades en un futuro, nos tememos que lejano, debido más que nada, al costo del evento: unos 20 millones de dólares en equipos informáticos. Pero es un primer paso, que muestra una sugestiva aplicación práctica de los ordenadores.

SI NO PUEDES VENCERLOS...

Bajo el paradójico lema **«socio de mis competidores»** , la firma sueca Ericsson, ferviente partidaria de la compatibilidad IBM, aumenta su escalada ofreciendo una gama de terminales compatibles con Digital Equipment, Sperry Univac y Mary Poppins. También estará presente un micro portátil fabricado por el gigante japonés Matushita.

Si IBM los vende, **¿por qué yo no?**

LOS PROBLEMAS DE APPLE (Y II)

Como comentábamos hace poco, Apple está que trina ante la dimisión de Jobs, exfundador de la compañía, y «huido» con los mejores cerebros de la misma.

En el tribunal de Santa Clara *(California)*, Apple Computer ha presentado una demanda judicial contra el señor Jobs, acusándole de **«haber tramado en secreto»** su escapada para fundar una empresa rival de Apple, y, además, de haber **«seducido»** a colaboradores muy importantes para que le acompañaran, con idea de aprovecharse de sus conocimientos en los proyectos más secretos de Apple.

Considerándose injustamente perjudicada, la compañía americana exige como indemnización la bagatela de 5 millones de dólares. A la espera de las declaraciones de Jobs, el **«acusado»**, la prensa del corazón del sector informático americano se encontraba sobre ascuas.

El impasible Jobs ha dicho que se siente **«sorprendido»** de la querella de Apple. Algo es algo.

To be continued.

TORA, TORA, TORA

Las fábricas japonesas creadoras del sistema MSX no se resignan al relativo fracaso que su standard está teniendo hasta ahora. En efecto, las cifras de ventas no corresponden a lo esperado ni mucho menos.

Ha nacido el MSX2, un sistema compatible con el MSX «a secas», formado por máquinas de 64 Kbytes de RAM y nada menos, asómbrese, que 128 Kbytes de memoria para gráficos, lo cual implica una resolución en pantalla comparable a la fotográfica. Los responsables del MSX2 parecen pensar que el futuro del mercado de los **«home computers»** descansa en los juegos, o al menos en aquellos programas que posean gráficos por todo lo alto.

No cabe duda de que juegos o lo que sea que use 128 K de gráficos será de armas tomar; esperemos que las casas comerciales de software respondan pronto a la iniciativa nipona.

BULL: EL TORO

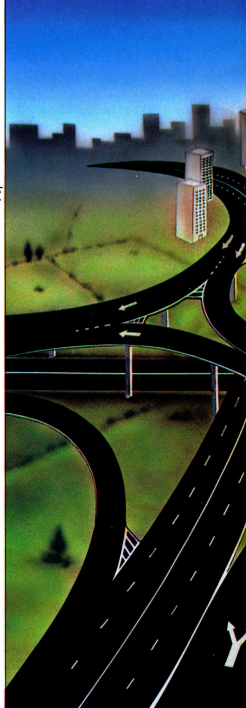
El primer fabricante europeo de informática *(junto con Olivetti)*, registrará este año un considerable aumento de beneficios después de perder hasta la camisa en los últimos tres años.

Jacques Stern, su presidente, aun fue más lejos augurando un ejercicio *(contable, claro)* **«muy positivo»** para el 86.

Se espera que el volumen de negocios alcance este año la respetable cifra de 16.000 millones de francos.

PROGRAMAS INTELIGENTES CON WHILE ... WEND

Esta vez no hay introducciones. Vamos a examinar, sin más, el programa número 1.



Pejo

Es idéntico al programa XII publicado la semana pasada, saca en pantalla todos los números a partir del 1 hacia arriba.

Es un programa bastante interesante, y una buena noticia es que, una vez tecleado, algunos de los próximos programas se basarán en él.

Quizá lo más chocante del programa 1 es el bucle WHILE...WEND. Como ve, WHILE y WEND actúan como **«paréntesis»** de una sección *(o cuerpo)* de instrucciones, que se desea repetir. En este caso las líneas que se están repitiendo son la 70 y 80.

Por supuesto, no puede haber un WHILE donde no se conoce, el micro necesita saber qué es WHILE.

Así que asociamos a WHILE lo que conocemos como una condición, mientras **«que»**...

Aquí la condición es sencilla: 1.° Como ve, da al micro el código numérico para **«verdadero»**. Por tanto la condición es WHILE verdadero, y después nuestro micro lo comprenderá todo, hasta que se diga otra cosa, todo se reduce siempre a WHILE.

En otras palabras, se repite el bucle hasta que lo interrumpimos pulsando la tecla de Escape dos veces.

Otra característica interesante del programa es la línea 70.

```
70 número=número+1
```

El efecto de esta línea es aumentar el valor de la variable «número» en la unidad.

Recuerde que primero se ejecuta lo que está a la derecha del signo igual, así que la línea 70 se lee: **: «Coge el valor de lo etiquetado 'número', súmale 1, coloca este nuevo valor en la etiqueta 'número'.»**

El signo igual no es en realidad el signo de igualdad que conocemos, **¿cómo puede ser un número igual a sí mismo más uno?** Es una asignación que puede leerse como **«convertirse»**.

Entonces, la línea 70 se lee: «'número' se convierte en lo etiquetado como 'número' más uno.»

De este modo la línea 70 aumenta el valor de la variable «número» en uno y la 80 saca en pantalla este nuevo valor. Mientras estas dos líneas están en el interior del bucle indefinido WHILE...WEND, el proceso se mantiene repitiéndose a sí mismo.

Una última cosa, aunque demos a la variable «número» el valor 0 en la línea 50, el primer número que sale en pantalla es el 1. **¿Saben por qué?**

Realmente la línea 50 es superflua puesto que el **Amstrad** supone que una variable numérica es cero a menos que se le haya dicho otra cosa.

El mes pasado le desafiamos a cambiar la línea 70 para que los números que salen aumenten en dos.

El programa II hace este truco:

Una de prácticas

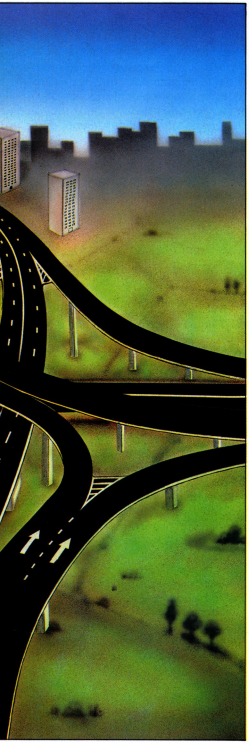
¿A que es fácil? Ahora le invitamos a que vea si es capaz de hacerlo incrementando en intervalos de 4 y 10. Aunque no tenga soltura todavía, creemos que será capaz ahora, utilizando el programa III como guía.

Otra propuesta que le hacemos, es empezar sacando en pantalla desde el 1000 para ir disminuyendo en intervalos de uno. Aquí, en vez de sumar uno, en la línea 70 tendrá que restarlo. El programa III demuestra cómo se hace:

Observe que damos a la variable **«número»** el valor 1001 en la línea 50, porque queremos que el primer número en pantalla sea 1000. Antes que se imprima la variable «número» *(línea 80)* este valor decrece en uno *(línea 70)*. Comenzando desde 1001 se nos permite esta sustracción.

Además nos encontramos con el primero de un interesante grupo, las desigualdades. Llamadas en el mundo de los ordenadores **«operadores relacionales»**, son interesantes porque nos indican cuál de los números es mayor o menor, si son iguales o distintos y así sucesivamente.

Primeros pasos



Probablemente ya los encontró en el colegio. Por ejemplo:

X + Y

significa que el valor de X es mayor que el valor de Y.

Del mismo modo:

X < Y

significa que el valor de X es menor que el valor de Y.

Desde luego, se tiende a olvidar en qué sentido deben ir los signos de las desigualdades, + . La manera de recordarlo es pensar que el signo tiene dos lados: el lado abierto o grande y el afilado o pequeño. El número **«grande»** va próximo al lado grande, y el número **«pequeño»** al lado pequeño.

Lo que quiere decir que:

7 + 4,

que se lee siete es mayor que cuatro es cierto, mientras que:

250 < 150,

que se lee doscientos cincuenta es menor que ciento cincuenta, ¡es claramente mentira!

El programa IV utiliza una desigualdad unida a la condición WHILE. La línea 60 se lee:

60 WHILE número < 23

Quiere decir que mantenemos la repetición del bucle limitado por WHILE...WEND mientras que el valor de la variable «número» es menor que 23.

Tan pronto como la línea 60 se encuentre un valor de la variable «número» que no sea más pequeño que 23, interrumpe el bucle— dice que se retira.

Es bastante sencillo, pero algo que podría preocuparle es que el propio número 23 se imprima. Podría pensarse que puesto que el bucle se está repitiendo: **«mientras la variable 'numero' sea menor que 23»** el propio 23 nunca se imprimirá.

De cualquier manera, recuerde el efecto de la línea 70 que suma 1 a la variable «número» antes de imprimirla. Después que el programa ha alcanzado e impreso el valor 23, el valor de la variable **«número»** es 22, así que impreso permite repetir el bucle, ya que 22 es menor que 23.

La línea 70 ha aumentado el valor de la variable **«número»** en 1 obteniendo 23 y la línea 80 lo saca en pantalla. Por supuesto, ya que la variable «número» es ahora 23, el bucle no se continuará.

Intente correr el programa con la línea 60 cambiando:

60 WHILE número + 23

No parece que ocurra nada, ¿verdad?. Es porque mientras la línea 50 coloque cero en la variable **«número»**, la condición del bucle (en la línea 60) no se cumple.

Salir de un bucle

Por lo tanto el bucle no se hace, lo hemos saltado u omitido. Y ahora ya que ahí no hay más programa sencillamente se termina.

La misma línea de razonamiento puede explicar los resultados que se obtienen de cambiar la línea 60 a:

60 WHILE número < 0

De la misma forma que hay desigualdades también hay igualdades, el signo = . Podemos combinarlas en expresiones tales como:

número < = 23

Que se lee: **«el valor de la variable 'número' es menor o igual que 23.»**

De igual manera:

número + = 23

se lee: «el valor de la variable 'número' es mayor o igual que 23.»

Recuerde que, la semana pasada, habíamos encontrado una combinación de desigualdades: el par < + , siendo su significado: **«distinto a.»**

Para ver cómo trabaja en la práctica una combinación semejante echemos una mirada al programa V. En la línea 60 se lee:

60 WHILE número < = 23

creemos que no tendrá ningún problema de interpretación.

Cuando corramos el programa descubriremos que en este caso 24 es el último número que aparece en pantalla.

Veamos este caso, a diferencia de programa IV, puede entrar en el bucle con una variable **«número»** igual a 23. La línea 70 le incrementa uno, y la línea 80 saca en pantalla el resultado 24.

Cuando se intenta hacer el bucle otra vez, como la variable «número» no es menor o igual que 23, se alta el bucle.

Como último paso, intentaremos utilizar mayor que para la condición del bucle WHILE... WEND —en una modificación del programa IV— no hemos conseguido hacer nada demasiado espectacular. El programa VI demuestra ahora cómo puede servir en este sentido.

Antes de correrlo intente encontrar qué número será el último que saldrá en pantalla.

Bien, desde la línea 60 sabemos que el bucle sólo se realizará si el valor de la variable **«número»** de entrada es mayor que 30. *(A propósito, la línea 50 asegura que empezamos con un número mayor que 30, y la línea 70 nos dice que descienden de uno en uno.)*

El último valor posible que **«activará»** el bucle será 31. No puede ser 30 o menor, ya que entonces la variable **«número»** no sería mayor que 30.

Por tanto, la última vez que se entra en el bucle WHILE... WEND, la variable **«número»** es 31. La línea 70 le disminuye en uno y la línea 80 escribe el valor resultante, 30.

Si el micro intenta hacer el bucle otra vez no lo conseguirá, ya que 30 no es mayor que 30.

La sentencia IF... THEN

El bucle WHILE... WEND no son las únicas palabras Basic que pueden usarse con condiciones. Esta sentencia IF... THEN que también es válida.

El uso de IF... THEN no puede ser más fácil ya que feflejan muy bien el lenguaje inglés.

Por ejemplo, hay una frase típicamente paternal:

IF no te vas a la cama
THEN me enfadaré

o la típicamente matrimonial:

IF lo dices otra vez
THEN estalláre

La idea es que de una condición después de IF, y que después de THEN especifique las calamitosas consecuencias si se ha cumplido esta condición.

Por su puesto que si la condición no se cumple, entonces ya no se realiza lo que está detrás de THEN.

(¿Sabe si la última sentencia es una instrucción IF... THEN?)

A menudo las condiciones que especificamos en nuestros programas suponen igualdades o desigualdades. Por ejemplo:

IF número + 1000 THEN PRINT
«Es un número grande.»
IF número < + supuesto THEN PRINT
«Su suposición es equivocada.»

El programa VII utiliza estas ideas para decirnos si el número es mayor que, igual a o menor que 10. Metamos el número en la línea 50, entonces las líneas 60, 70 y 80 escriben el resultado de la comparación para cada uno de los tres casos o condiciones.

Si la variable «número» cumple una de las condiciones *(es decir, es mayor que, igual a o menor que 10)*, se imprime el mensaje apropiado.

Observe que es imposible para cualquier valor de la variable **«número»** provocar más de un mensaje.

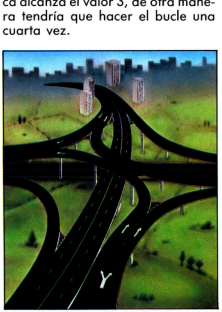
El programa VIII desarrolla las ideas que estamos discutiendo. Las líneas 70 a 100 hacen el trabajo del programa VII comparando el número con 10.

Ambas condiciones pueden mezclarse

La novedad es que en este caso la comparación está envuelta por un bucle WHILE... WEND. Vamos ha hacer nuestras comparaciones en tres tiempos. *(Conozcamos el cielo.)* Así que vamos a introducir la variable apropiadamente llamada **«contador»** que se mantendrá al tanto de cuantas veces hacemos WHILE... WEND.

Al principio la variable «contador» está marcando cero *(línea 50)*, cada vez que se ejecute el bucle será incrementada en uno *(línea 120)*. Así que, la primera vez que se ejecute el bucle, la variable **«contador»** es cero, la segunda vez es uno, la tercera es dos.

Ya que necesitamos hacer el bucle sólo tres veces, sería mejor asegurarnos que la variable **«contador»** nunca alcanza el valor 3, de otra manera tendría que hacer el bucle una cuarta vez.



Para lo cual agregaremos la condición a nuestro WHILE.

60 WHILE contador < 3

Puede que parezca un poco extraño que valga menos que 3, pero recuerde, empezó contando cero.

Si cree que esto está claro, puede empezar contando desde uno, cambiando la línea 50 a:

50 contador = 1

En este caso necesita cambiar la condición del bucle para alterar la línea 60 a:

60 WHILE contador < =3

A propósito, ¿sabe sí:

60 WHILE contador < + 4

y

60 WHILE contador < 4

son equivalentes a esta nueva versión de la línea 60? Personalmente, nos parece que nuestra primera alteración era clara, ya que necesita el valor 3.

A propósito, la línea 110 es, precisamente, un espacio en blanco entre los grupos de mensajes del bucle, y la línea 140 nos hace saber que el programa ha terminado.

No olvide que puede utilizar la combinación < < en el sentido de **«no es igual que»** con cadenas y caracteres así como con variables numéricas. El programa IX lo demuestra, mejor dicho, es un ejemplo tonto.

Es muy similar a un programa que tuvimos el mes pasado, de modo que no debe causarle ningún problema.

Ahora veamos el programa X. Lo crea o no, es un equivalente exacto al programa IX. En lugar de utilizar WHILE... WEND, vamos a usar la combinación IF... THEN y GOTOs.

Hasta el momento, hemos utilizado solamente PRINT después de THEN. Sin embargo, se puede utilizar cualquier palabra clave Basic, incluyendo GOTO.

GOTO, con el que ya nos hemos encontrado, hace que el micro salte al número de línea especificado y sigue operando allí.

La línea 60 da entrada al valor de una variable **«respuestas$»** en respuesta a la pregunta: **«¿lo repetiremos?»**

La línea 70 deja una línea en blanco como espacio, así saltamos inmediatamente a la línea 50 vía el GOTO de la línea 80.

Entonces la línea 50 analiza nuestra respuesta a la pregunta. Si respondemos de otra manera que

«no», se continuará con el *(más bien insignificante)* programa, si respondemos «no» se deja el programa.

La línea 50 utiliza GOTO acompañando el final de una sentencia IF... THEN.

En efecto, dice, IF el valor de **«respuestas$»** es **«no»** THEN GOTO a la línea 90.

Ya que hemos dado entrada a un «no», el efecto de la línea 50 nos haría un GOTO a la línea 90. La línea 90 contiene la palabra clave BASIC END que, como su nombre indica, provoca el final de la ejecución del programa.

Por otra parte, cualquier otra respuesta no cumple la condición, por lo tanto no haremos lo que viene después de THEN, sino que sencillamente se continuará el programa.

Es asombroso que el micro encuentre lo primero la línea 50. Al fin y al cabo, no hemos dado entrada a ningún valor para la variable **«respuesta$»**.

Bueno, el **Amstrad** lo supone, hasta que le demos a la variable un valor, ésta no tiene ninguno cualquiera que sea la comparación. Verdaderamente no puede tener el valor **«no»**, así que el programa no hace lo que viene después de THEN, pero continúa normalmente.

Creemos que estará de acuerdo en que es una suerte tremenda el poder llevar a cabo este trabajo con un manejo tan simple como el bucle WHILE... WEND en el programa IX, y este último está lejos de ser fácil de comprender.

Los GOTOs, con su tendencia a saltar sobre los programas, que parecen liebres de marzo, son una primera causa de oscuridad y esconden errores en los programas.

Los buenos programadores raramente los utilizan *—casi nunca—* mientras haya alternativas claras y sencillas. Esperamos que usted también los evite.

PROGRAMAS

```
10 REM ****************************
20 REM            Programa I
30 REM ****************************
****
40 MODE 1
50 numero=0
60 WHILE -1
70 numero=numero+1
80 PRINT "El numero es ahora ";nume
ro
90 WEND
```

```
10 REM ****************************
20 REM            Programa II
30 REM ****************************
****
40 MODE 1
50 numero=0
60 WHILE -1
70 numero=numero+2
80 PRINT "El numero es ahora ";nume
ro
90 WEND
```

```
10 REM ****************************
20 REM            Programa III
30 REM ****************************
40 MODE 1
50 numero=100:
60 WHILE -1
70 numero=numero-1
80 PRINT "El numero es ahora ";nume
ro
90 WEND
```

```
10 REM ****************************
20 REM            Programa IV
30 REM ****************************
**
40 MODE 1
50 numero=0
60 WHILE NUMERO<23
70 numero=numero+1
80 PRINT "El numero es ahora ";nume
ro
90 WEND
```

```
10 REM ****************************
20 REM            Programa V
30 REM ****************************
**
40 MODE 1
50 numero=0
60 WHILE NUMERO<=23
70 numero=numero+1
80 PRINT "El numero es ahora ";nume
ro
90 WEND
```

```
10 REM ****************************
20 REM            Programa VI
30 REM ****************************
40 MODE 1
50 numero=51
60 WHILE NUMERO)30
70 numero=numero-1
80 PRINT "El numero es ahora ";nume
ro
90 WEND
```

```
10 REM ****************************
20 REM            Programa VII
30 REM ****************************
40 MODE 1
50 INPUT "Teclee un numero";numero
60 IF numero)10 THEN PRINT"El numer
o es mayor que 10"
70 IF numero=10 THEN PRINT"El numer
o es igual a 10"
80 IF numero<10 THEN PRINT"El numer
o es menor que 10"
```

```
10 REM ****************************
20 REM            Programa VIII
30 REM ****************************
40 MODE 1
50 contador=0
60 WHILE contador<3
70 INPUT"Teclee un numero";numero
80 IF numero)10 THEN PRINT"El numer
o es mayor que 10"
90 IF numero=10 THEN PRINT"El numer
o es igual a 10"
100 IF numero<10 THEN PRINT"El nume
ro es menor que 10"
110 PRINT
120 contador=contador+1
130 WEND
140 PRINT "Se acabo"
```

```
10 REM ****************************
20 REM            Programa IX
30 REM ****************************
40 MODE 1
50 WHILE respuesta$<>"no"
60 INPUT"Me lo repite";respuesta$
70 PRINT
80 WEND
```

```
10 REM ****************************
20 REM            Programa X
30 REM ****************************
**
40 MODE 1
50 IF respuesta$="no" THEN GOTO 90
60 INPUT"Me lo repite";respuesta$

70 PRINT
80 GOTO 50
90 END
```

INTRODUCCION AL PROGRAMA "GRAFICAS"

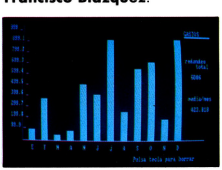
No siempre resulta fácil escribir un prólogo, así que voy a recurrir a un tópico muy conocido para «coger el hilo». Ahí va: una imagen vale más que mil palabras.

COMPATIBLE
CPC 464
CPC 664

No, el Serie Oro de este número no va de fotografía, sino que se trata de un programa escrito íntegramente en Basic pensado para representar gráficamente información, en forma de diagramas de barras, tridimensionales, de tarta *(que el autor llama cilindros)*, etc.

Creo que su utilidad resulta clara para cualquier persona que necesite ver rápidamente cómo evoluciona una serie de datos con el tiempo, acerca de conceptos tales como nóminas, ingresos, gastos, en fin, cualquier partida que uno maneje en su negocio. En mi opinión, gráficas combina perfectamente utilidad con estética en la presentación de los datos en pantalla; tanto el que sólo le interese usarlo como herramientas, como el programador deseoso de investigar acerca de los gráficos, encontrarán en el programa una gran ayuda.

Pero basta ya de prólogos. Cedo la palabra al padre de la criatura: **Francisco Blázquez**.



Explicación General

Gráficas I es un programa escrito íntegramente en Basic cuya utilidad es la representación gráfica de cualquier tipo de tablas de valores.

Incluye varias clases distintas de diagrama, desde el simple de barras de dos dimensiones al complejo de dos líneas en contraste.

Comienza el programa con dos pantallas de instrucciones en las cuales se expone, de un modo reducido, todo lo que a continuación se va a explicar.

Seguidamente se accede al menú principal presionando cualquier tecla. Este consta de seis opciones distintas de gráficas más la posibilidad de otra doble opción en las tres primeras. Son las siguientes:

1. Barras en dos dimensiones *(sólo valores positivos). %*

2. Barras en dos dimensiones *(sólo valores negativos). %*

3. Barras en dos dimensiones *(combinación valores pos. y neg.) %*

4. Barras en dos dimensiones *(para valores positivos).*

5. Diagramas de cilindros *(4 bloques como máximo).*

6. Gráficas de líneas.

7. Instrucciones.

Nota: Las opciones con el símbolo «%» constan a su vez de la opción Picos como variante de las barras.

1. Barras 2D *(valores positivos)*

Tras haber escogido esta opción y al igual que en la dos y la tres, aparecerá en pantalla una sucesión de preguntas las cuales te irán pidien-

do distintos valores y datos que deberás ir introduciendo. Después del concepto general, te pedirá el máximo valor, del cual dependerá que aparezca o no la media/mes y el total de los valores introducidos *(aparecerá siempre que sea menor de 1001)*. Habiendo respondido a esta pregunta, el programa pasará a pedirte los valores de cada barra en concepto de meses. Esto se debe, a que las opciones 1,2 y 3 están diseñadas para tablas anuales. Es obvio indicar que a pesar de esta predisposición del programa, puedes usar esta opción sin tener en cuenta ni los meses ni la media si aparece.

Y después de la entrada de datos, se nos plantea otra opción: **¿Barras** (b) **o Picos?** De este modo, y en las tres primeras opciones, podemos variar las gráficas consiguiendo que aparezcan cuñas en vez de barras. La elección depende ya del gusto de cada uno.

Una vez obtenida la gráfica podemos borrarla de pantalla pulsando cualquier tecla, con lo que volveremos al menú principal.

2. Barras 2D *(valores negativos)*

Esta opción es muy similar a la anterior. El interrogatorio para la entrada de datos es el mismo. Cuando te pida el máximo valor, has de introducirlo en valor absoluto, o sea prescindiendo del signo aunque sea negativo. Del mismo modo, el resto de las entradas deberás efectuarlas sin tener en cuenta el signo. También se te presentará la opción barras/picos.

Ahora el color del fondo será el rojo en vez del negro. La escala de referencia llevará el signo «—» delante de cada cifra.

3. Barras 2D *(combinación de valores positivos y negativos)*

Aparecerá el mismo interrgatorio que en las opciones 1 y 2. El máximo valor debes introducirlo en forma absoluta, pero no así los demás valores, los cuales habrán de llevar el signo (pos. o neg.) según corresponda. La pantalla está dividida en dos partes iguales madiante una línea horizontal, que dentro del eje de coordenadas, marca el valor **0** en la escala comparativa.

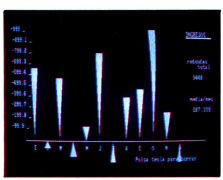
Serie Oro

De esta línea y hacia arriba, saldrán los valores positivos en forma de barra o pico *(según se haya escogido)*, y hacia abajo, los negativos. El fondo es de color azul y la gráfica está hecha a una escala menor en tamaño.

La media/mes y el total dependen del máximo valor, al igual que en anteriores opciones.

4. Barras en tres dimensiones *(sólo valores positovos)*

Esta opción introduce una variante realmente práctica: el número de barras es opcional, pudiendo elegir de 2 a 15 barras. Esto posibilita la representación de tablas que no tengan 12 separaciones o conceptos, completando de este modo, a las opciones 1,2 y 3. El grosor de las barras depende de su número, a mayor cantidad de barras, más finas son, y viceversa.



Las barras se nos muestran en tres colores y sobre un eje de coordenadas tridimensional. A la derecha, en color blanco, aparece una escala comparativa con el máximo valor como tope. Bajo ella, saldrá el concepto general de la tabla representada. Tanto en esta opción como en las anteriores, si una barra tiene un valor mayor al del máx. val., esta barra se saldrá por la parte superior del eje de gráficas.

5. Cilindros

Es otra de las alternativas que cuenta con representación tridimensional.

PRINCIPALES LINEAS DEL PROGRAMA

LINEAS	FUNCION
60	Comienzo del programa. Dos pantallas de instrucciones.
260	Menú principal: inicio.
270	Clear: pone a 0 todas las variables. Selección y llamada de las distintas opciones.
550	Final del menú. Retorno a la línea 260 si se desea.
560	Inicio de la opción 4, *(barras 3D)*. Toma de datos y conversión de los valores a proporciones de pantalla según el máximo valor introducido.
1000	Ejecución tras la toma de datos.
3040	Final opción 4 y vuelta al menú principal.
3050	Comienzo de la opción 5 *(cilindros)*: toma de datos.
3460	Conversión a escala de los valores y representación de la gráfica.
3910	Final y vuelta al menú principal.
3920	Inicio de la opción 6 *(líneas)*. Toma de datos y conversión de valores.
4680	Representación de la gráfica.
5240	Fin de opción 6. Vuelta al menú.
5250	Comienzo de las opciones 1, 2 y 3, *(barras 2D)*. Toma de datos, conversión de valores y llamadas a las distintas opciones.
5620	Representación de ejes y pantalla. Llamada a PICOS si se elige esta opción. Ejecución de barras si se eligen.
6260	Fin barras opción 3 y retorno al menú.
6270	Representación de ejes y pantallas de opciones 1 y 2. Llamadas a PICOS si se eligen, si no, representación de barras.
6880	Fin barras opciones 1 y 2. Retorno al menú.
6890	Ejecución de opción PICOS.
7070	Retorno a la línea 6250 para regreso al menú de la opción 3.
7080	Regreso a la línea 6870 para retorno al menú de las opciones 1 y 2.

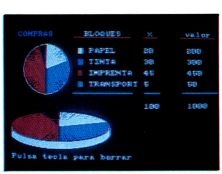
Como anteriormente ha sucedido en el resto de las opciones, comienza con un interrogatorio para la entrada de datos. Esta vez, el programa se sale de la tónica marcada por los interrogatorios anteriores.

En primer lugar nos pide el concepto general, el total, y luego intercala las preguntas de valores y conceptos *(por bloques)*.

Esta opción es utilizada con la finalidad de apreciar el valor relativo de cualquier dato dentro de un total previamente fijado. Resalta el valor de cada bloque como formante del total. Por ello, aunque te pedirá el concepto de los cuatro bloques, no pasará así con los valores respectivos, ya que tras meter el tercer valor y concepto del cuarto bloque, el valor de este último bloque se deduce al sumar los tres primeros y restárselo al total que anteriormente has introducido. Si te equivocas y la suma de los bloques resulta mayor que el total, el mismo programa te devuelve al principio de la entrada de datos para que rectifiques. Del mismo modo no admite ningún valor que sea superior al total.

La representación es bastante completa: Sobre un fondo negro y con tres colores, se nos presenta una doble visión del cilindro. Una visión desde arriba, y la otra en perspectiva caballera tridimensional. Cada bloque se irá formando simultáneamente en ambas perspectivas, sobre dos moldes que aparecen al terminar la entrada de datos. Los colores de los tres primeros bloques son blanco, azul y rojo respectivamente, quedando el cuarto bloque representado por la falta de color.

A la derecha de la visión aérea, se configura una tabla en la cual aparece el concepto de cada bloque asociado a su color en la gráfica, el



tanto por ciento que supone del total y su valor real.

No tienes por qué usar forzosamente esta opción con cuatro bloques, sino que se puede usar para dos y tres bloques, ya que la suma de los valores de los primeros bloques igualará al total y el valor de los bloques no usados será **0**.

6. Diagramas de líneas

Esta representación nos muestra la evolución de dos líneas a partir de los valores que se introduzcan. Su contraste resulta muy claro por lo que esta opción es propia de representaciones de tablas de valores opuestas.

Al igual que las barras 3D, cuentas con la posibilidad de elegir el número de valores por línea, aunque ahora oscilará entre 3 y 13.

Admite valores positivos y negativos por lo que el signo es fundamental a la hora de introducir datos. Excepto el máximo valor, que deberá ser introducido en forma absoluta, todos los demás habrán de llevar su signo, ya que admite la combinación de valores pos. y neg.

Tras introducir todos los valores, se pasa a la representación: sobre un eje de coordenadas plano, con valores por encima y por debajo de **0**, según una escala comparativa de la derecha, aparecen dos líneas de distinto color (cada color a un concepto).

Si algún valor de cualquiera de las dos líneas fuera superior en forma absoluta al máximo valor, la línea se saldría del eje de coordenadas y desaparecería, volviendo a verse si el siguiente valor de la misma línea no sobrepasa al máximo.

Pulsando una tecla borras y vuelves al menú principal.

7. Instrucciones

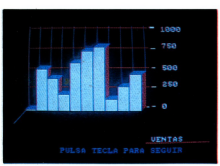
Con esta opción reaparecerán en pantalla las dos hojas de instrucciones siempre que lo desees.

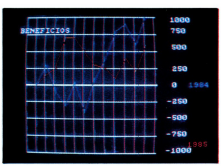
VARIABLES PRINCIPALES

VARIABLE	FUNCION
X	Selección opción.
C$	Retorna al principio de la toma de datos.
CG$	Concepto general en barras 3D.
mv,mva	Máximo valor, usado para establecer la escala a tamaño de la pantalla mediante una regla de tres.
nb	Número de barras 3D (opcional): usado también para delimitar el reparto de separaciones en los ejes de coordenadas.
h1...h15	Valor o altura de las barras 3D.
t	Total en gráficas de cilindros: se usa para obtener otros datos como el valor relativo de los bloques.
c1$,c2$,c3$,c4$	Conceptos de los bloques de las representaciones de cilindros.
v1,v2,v3,v4	Valores de los bloques de la opción 5.
nb$	Concepto general de diagramas de líneas.
n1$	Concepto línea r.
n2$	Concepto línea s.
nv	Número de valores por línea: viena a funcionar como el nb en las barras 3D.
r1...r13	Valores línea r.
s1...s13	Valores línea s.
e,f,m,a,my,j	
j1,ag,s,o,n,d	Valores en forma de meses de las barras 2D.
gb$	Opción barras/picos en las barras 2D.
db,j,d,p,a,b,c, f,l,kl,k,w,...etc	Variables de uso general.

FORMULAS CLAVE

FORMULA	FUNCION
ha1=*((ha1×230)/mv)*	Al igual que ha2....ha15, lo que hace es poner el valor introducido en función del máximo valor y del espacio en la pantalla *(n.º de pixels)*.
step*(350/nb)*	Delimita el número de separaciones o el n.º de barras verticales en función del n.º de barras escogido.
v1=*(v1×360)/t*	Traduce el valor introducido a grados en proporción al total escogido. *(También v2...v4)*.
s1=*(s1×199)/mva*	Al igual que s2...s13 y que r1...r13, su finalidad es proporcionar el valor parcial de la recta en un valor proporcional al máximo valor y en función al n.º de pixels en pantalla destinados a la gráfica.
e=CINT *((e×k1)/mv)*	Como los anteriores, su función es hallar el valor proporcional al máximo valor y al espacio para la representación de cualquiera de los valores introducidos como parciales. En las opciones 1 y 2 el valor de k1=330 y en la n.º 3 su valor es de 165.

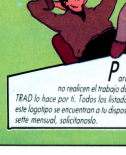




Serie Oro

```
10 '*************************
**********
20 '***    GRAFICAS/Fco J. BLAZQUEZ
 ***
30 '***          1985'
 ***
40 '*************************
*********
60 MODE 1:INK 1,13:INK 2,11:INK 3,6
:INK 0,0:BORDER 0
70 CLS
80 LOCATE 16,3:PEN 3:PRINT "GRAFICA
S"
90 LOCATE 4,5:PEN 1:PRINT "    El p
rograma sirve para hacer varios tip
os de diagramas.Hay cuatro opciones
  principales y cada una se ajusta
a una   determinada finalidad."
100 PRINT
110 PRINT "     Si lo que se preten
de es apreciar  la participacion de
 una cifra dentro de un total dado,
o sea el valor relativo,lo mas conv
eniente es elegir la opcion de cili
ndros( cuatro conceptos distintos
 como max)."
120 PRINT
130 PRINT "     Otra de las opcione
s,las barras en  dos dimensiones,vi
ene programada para  diagramas anua
les,de tal modo que la en_trada de
datos se hara por meses."
140 LOCATE 9,24:PEN 2:PRINT "PULSA
TECLA PARA SEGUIR"
150 WHILE INKEY$="":WEND
160 CLS
170 PEN 1
180 LOCATE 16,3:PEN 3:PRINT "GRAFIC
AS"
190 LOCATE 1,5:PEN 1:PRINT "    Den
tro de esta misma alternativa se pu
ede escoger entre dos tipos de barr
as; normales o de ( picos>.Ademas h
ay tres posibilidades de diagrama;
para valores positivos,para negativ
os y para combina_cion de ambos."
200 PRINT
210 PRINT "    Una variacion de est
as graficas son las barras tridimen
sionales,que gracias a la posibilid
ad de elegir el numero de barras,pe
rmiten su aplicacion a tablas  de
muy distinta composicion."
220 PRINT
230 PRINT "   Al igual que las barr
as(3D),la opcionde lineas incluye l
a eleccion del numero de divisiones
.Las representaciones por  medio de
 lineas facilitan el contraste de l
a evolucion de dos funciones distin
_tas,(los gastos e ingresos,por eje
mplo)."
240 LOCATE 9,25:PEN 2:PRINT "PULSA
TECLA PARA SEGUIR"
250 WHILE INKEY$="":WEND
260 MODE 1
270 CLEAR
280 INK 0,0:BORDER 0:INK 1,13:INK 2
,11:INK 3,6
290 LOCATE 10,2:PEN 2:PRINT "MENU P
RICIPAL"
300 LOCATE 9,3:PEN 3:PRINT STRING$(
15,"-")
310 LOCATE 3,5:PEN 2:PRINT "BARRAS
:"
320 LOCATE 1,7:PEN 1:PRINT"    '
- Valores positivos.              2D
'     - Valores negativos.
'     - Valores positivos y neg
ativos. 3D'    -(Solo valores posit
ivos)."
330 PRINT
340 PEN 2:PRINT "  CILINDROS :"
350 PRINT
```

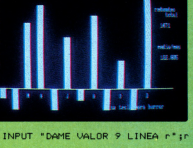
```
360 PEN 1:PRINT "        - 4 bloque
s de conceptos(max)."
370 PRINT
380 PEN 2:PRINT" LINEAS :"
390 PRINT
400 PEN 1:PRINT"        - 2 lineas
en contraste (r y s)."
410 LOCATE 8,7:PEN 3:PRINT "1"
420 LOCATE 8,8:PRINT "2"
430 LOCATE 8,9:PRINT "3"
440 LOCATE 8,10:PRINT "4"
450 LOCATE 8,14:PRINT "5"
460 LOCATE 8,18:PRINT "6"
470 LOCATE 2,20:PEN 2:INPUT "Para v
olver a instrucciones pulsa (?), p
ara elegir opcion el numero + ENTER
"isx$
480 IF sx$="1" THEN GOTO 5250
490 IF sx$="2" THEN GOTO 5250
500 IF sx$="3" THEN GOTO 5250
510 IF sx$="4" THEN GOTO 560
520 IF sx$="5" THEN GOTO 3050
530 IF sx$="6" THEN GOTO 3920
540 IF sx$="7" THEN GOTO 60
550 GOTO 260
560 MODE 1:'$$$$$$$$$$ BARRAS $$$$
$$$$$$
570 CLS
580 INK 0,0:BORDER 0:INK 1,13:INK 2
,2:INK 3,6:PEN 1:PRINT:PRINT
590 INPUT "        INTRODUCE CONCEPTO(m
ax.7 letras)          ";cg$
600 INPUT "  INTRODUCE n. DE BARR
AS(2-15)";nb:IF nb<2 OR nb>15 THEN
GOTO 570
610 INPUT "  INTRODUCE MAXIMO VAL
OR";mv
620 INPUT "  INTRODUCE VALOR BARR
A 1";ha1:LET ha1=((ha1*230)/mv)+1F
ha1=0 THEN GOTO 620
630 INPUT "  INTRODUCE VALOR BARR
A 2";ha2:LET ha2=((ha2*230)/mv)+1F
ha2=0 THEN GOTO 630
640 IF nb=2 THEN GOTO 980
650 INPUT "  INTRODUCE VALOR BARR
A 3";ha3:LET ha3=((ha3*230)/mv)+1F
ha3=0 THEN GOTO 650
660 IF nb=3 GOTO 980
670 INPUT "  INTRODUCE VALOR BARR
A 4";ha4:LET ha4=((ha4*230)/mv)+1F
ha4=0 THEN GOTO 670
680 IF nb=4 GOTO 980
690 INPUT "  INTRODUCE VALOR BARR
A 5";ha5:LET ha5=((ha5*230)/mv)+1F
ha5=0 THEN GOTO 690
700 IF nb=5 GOTO 980
710 INPUT "  INTRODUCE VALOR BARR
A 6";ha6:LET ha6=((ha6*230)/mv)+1F
ha6=0 THEN GOTO 710
720 IF nb=6 GOTO 980
730 INPUT "  INTRODUCE VALOR BARR
A 7";ha7:LET ha7=((ha7*230)/mv)+1F
ha7=0 THEN GOTO 730
740 IF nb=7 GOTO 980
750 INPUT "  INTRODUCE VALOR BARR
A 8";ha8:LET ha8=((ha8*230)/mv)+1F
ha8=0 THEN GOTO 750
760 IF nb=8 GOTO 980
770 INPUT "  INTRODUCE VALOR BARR
A 9";ha9:LET ha9=((ha9*230)/mv)+1F
ha9=0 THEN GOTO 770
780 IF nb=9 GOTO 980
790 INPUT "  INTRODUCE VALOR BARR
A 10";ha10:LET ha10=((ha10*230)/mv)
:IF ha10=0 THEN GOTO 790
800 IF nb=10 GOTO 980
810 INPUT "  INTRODUCE VALOR BARR
A 11";ha11:LET ha11=((ha11*230)/mv)
:IF ha11=0 THEN GOTO 810
820 IF nb=11 GOTO 980
830 INPUT "  INTRODUCE VALOR BARR
A 12";ha12:LET ha12=((ha12*230)/mv)
:IF ha12=0 THEN GOTO 830
840 IF nb=12 GOTO 980
850 INPUT "  INTRODUCE VALOR BARR
A 13";ha13:LET ha13=((ha13*230)/mv)
:IF ha13=0 THEN GOTO 850
860 IF nb=13 GOTO 980
870 INPUT "  INTRODUCE VALOR BARR
A 14";ha14:LET ha14=((ha14*230)/mv)
:IF ha14=0 THEN GOTO 870
880 IF nb=14 GOTO 980
890 INPUT "  INTRODUCE VALOR BARR
A 15";ha15:LET ha15=((ha15*230)/mv)
:IF ha15=0 THEN GOTO 890
900 IF ha<1 THEN ha=1:IF ha2<1 THEN
 ha2=2
910 IF ha3<1 THEN ha3=1:IF ha4<1 TH
EN ha4=1
920 IF ha5<1 THEN ha5=1:IF ha6<1 TH
EN ha6=1
930 IF ha7<1 THEN ha7=1:IF ha8<1 TH
EN ha8=1
940 IF ha9<1 THEN ha9=1:IF ha10<1 T
HEN ha10=1
950 IF ha11<1 THEN ha11=1:IF ha12<1
 THEN ha12=1
960 IF ha13<1 THEN ha13=1:IF ha14<1
 THEN ha14=1
970 IF ha15<1 THEN ha15=1
980 INPUT "Si quieres cambiar algo
pulsa c + enter ";c$
990 IF c$="c" THEN GOTO 570
1000 CLS
1010 INK 1,24
1020 LOCATE 27,22:PRINT cg$
1030 LOCATE 26,23:PEN 3:PRINT STRIN
G$(12,CHR$(208)):PEN 1
1040 FOR db=50 TO 400 STEP (350/nb)

1050 ORIGIN db,150:DRAW -50,-50,3
1060 ORIGIN db,150:DRAW 0,235,2:NEX
T db
1070 ORIGIN 50,150:DRAW 0,235,1
1080 ORIGIN 50,150:DRAW 360,0,1
1090 ORIGIN 50,150:DRAW -50,-50,1
1100 FOR d=57.5 TO 230 STEP 57.5
1110 ORIGIN 50,150+d:DRAW 360,0,3
1120 ORIGIN 50,150+d:DRAW -50,-50.3
:NEXT d
1130 IF mv>1000 THEN k1=3:IF mv<100
1 THEN k1=4
1140 IF mv>10000 THEN k1=2
1150 LOCATE 27,2:PRINT CHR$(208);mv
1160 LOCATE 27,5:PRINT CHR$(95);(mv
/4)*3
1170 LOCATE 27,9:PRINT "-";(mv/4)*2
1180 LOCATE 27,12:PRINT CHR$(95);mv
/4
1190 LOCATE 27,16:PRINT "- 0"
1200 '********** BARRA 1 **********
*****
1210 FOR c=1 TO ha1
1220 ORIGIN 38,140+c:DRAW (350/nb),
0,1
1230 ORIGIN 38+(350/nb),140:DRAW 12
,10,1
1240 ORIGIN 38+(350/nb),140+c:DRAW
12,10,3:NEXT c
1250 ORIGIN 38+(350/nb),140+ha1:DRA
W 12,10,1
1260 ORIGIN 38,140+ha1:DRAW 12,10,1

1270 ORIGIN 38+(350/nb),140:DRAW 0,
ha1,2
1280 ORIGIN 38,140:DRAW 0,ha1,2
1290 FOR j=0 TO 350/nb
1300 ORIGIN 38+j,140+ha1:DRAW 12,10
,2:NEXT j
1310 ' ********** BARRA 2 **********
******
1320 FOR c=1 TO ha2
1330 LET k=(350/nb)
1340 ORIGIN 38+k,140+c:DRAW (350/nb
),0,1
1350 ORIGIN 38+(2*k),140:DRAW 12,10
,1
1360 ORIGIN 38+(2*(350/nb)),140+c:D
RAW 12,10,3:NEXT c
1370 ORIGIN 38+(2*(350/nb)),140+ha2
:DRAW 12,10,1
1380 ORIGIN 38+k,140+ha2:DRAW 12,10
,1
1390 ORIGIN 38+(2*(350/nb)),140:DRA
W 0,ha2,2
1400 ORIGIN 38+k,140:DRAW 0,ha2,2
1410 FOR j=0 TO 350/nb
1420 ORIGIN 39+k+j,140+ha2:DRAW 12,
10,2:NEXT j
1430 IF nb=2 THEN GOTO 3020
1440 ' ******* 3 Barra *****
1450 FOR c=1 TO ha3
1460 LET k=2*(350/nb)
1470 ORIGIN 38+k,140+c:DRAW (350/nb
),0,1
1480 ORIGIN 38+(2*k),140:DRAW 12,10
,1
1490 ORIGIN 38+(3*(350/nb)),140+c:D
RAW 12,10,3:NEXT c
1500 ORIGIN 38+(3*(350/nb)),140+ha3
:DRAW 12,10,1
1510 ORIGIN 38+k,140+ha3:DRAW 12,10
,1
1520 ORIGIN 38+(3*(350/nb)),140:DRA
W 0,ha3,2
1530 ORIGIN 38+k,140:DRAW 0,ha3,2
1540 FOR j=0 TO 350/nb
1550 ORIGIN 39+k+j,140+ha3:DRAW 12,
10,2:NEXT j
1560 IF nb=3 THEN GOTO 3020
1570 ' ***** 4 BARRA *****
1580 FOR c=1 TO ha4
1590 LET k=3*(350/nb)
1600 ORIGIN 38+k,140+c:DRAW (350/nb
),0,1
1610 ORIGIN 38+(3*k),140:DRAW 12,10
,1
1620 ORIGIN 38+(4*(350/nb)),140+c:D
RAW 12,10,3:NEXT c
1630 ORIGIN 38+(4*(350/nb)),140+ha4
:DRAW 12,10,1
1640 ORIGIN 38+k,140+ha4:DRAW 12,10
,1
1650 ORIGIN 38+(4*(350/nb)),140:DRA
W 0,ha4,2
1660 ORIGIN 38+k,140:DRAW 0,ha4,2

1670 FOR j=0 TO 350/nb
1680 ORIGIN 39+k+j,140+ha4:DRAW 12,
10,2:NEXT j
1690 IF nb=4 THEN GOTO 3020
1700 ' ****** 5 barra *****
1710 FOR c=1 TO ha5
1720 LET k=4*(350/nb)
1730 ORIGIN 38+k,140+c:DRAW (350/nb
),0,1
1740 ORIGIN 38+(5*(350/nb)),140+c:D
RAW 12,10,3:NEXT c
1750 ORIGIN 38+(5*(350/nb)),140+ha5
:DRAW 12,10,1
1760 ORIGIN 38+k,140+ha5:DRAW 12,10
,1
1770 ORIGIN 38+(5*(350/nb)),140:DRA
W 0,ha5,2
1780 ORIGIN 38+k,140:DRAW 0,ha5,2

1790 FOR j=0 TO 350/nb
1800 ORIGIN 39+k+j,140+ha5:DRAW 12,
10,2:NEXT j
1810 IF nb=5 THEN GOTO 3020
1820 ' ****** 6 BARRA *******
1830 FOR c=1 TO ha6
1840 LET k=5*(350/nb)
1850 ORIGIN 38+k,140+c:DRAW (350/nb
),0,1
1860 ORIGIN 38+(6*(350/nb)),140+c:D
RAW 12,10,3:NEXT c
1870 ORIGIN 38+(6*(350/nb)),140+ha6
:DRAW 12,10,1
1880 ORIGIN 38+k,140+ha6:DRAW 12,10
,1
```



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

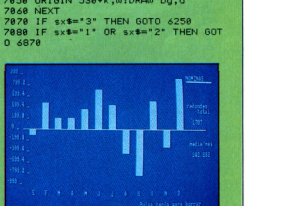
```
1890 ORIGIN 38+(&#(350/nb)),140:DRA
W 0,ha6,2
1900 ORIGIN 38+k,140:DRAW 0,ha6,2
1910 FOR j=0 TO 350/nb
1920 ORIGIN 39+k+j,140+ha6:DRAW 12,
10,2:NEXT j
1930 IF nb=6 THEN GOTO 3020
1940 ' ******* 7 BARRA *****
1950 FOR c=1 TO ha7
1960 LET k=6*(350/nb)
1970 ORIGIN 38+k,140+c:DRAW (350/nb
),0,1
1980 ORIGIN 38+(7*(350/nb)),140+c:D
RAW 12,10,3:NEXT c
1990 ORIGIN 38+(7*(350/nb)),140+ha7
:DRAW 12,10,1
2000 ORIGIN 38+k,140+ha7:DRAW 12,10
,1
2010 ORIGIN 38+(7*(350/nb)),140:DRA
W 0,ha7,2
2020 ORIGIN 38+k,140:DRAW 0,ha7,2
2030 FOR j=0 TO 350/nb
2040 ORIGIN 39+k+j,140+ha7:DRAW 12,
10,2:NEXT j
2050 IF nb=7 THEN GOTO 3020
2060 ' ***** 8 BARRA *****
2070 FOR c=1 TO ha8
2080 LET k=7*(350/nb)
2090 ORIGIN 38+k,140+c:DRAW (350/nb
),0,1
2100 ORIGIN 38+(8*(350/nb)),140+c:D
RAW 12,10,3:NEXT c
2110 ORIGIN 38+(8*(350/nb)),140+ha8
:DRAW 12,10,1
2120 ORIGIN 38+k,140+ha8:DRAW 12,10
,1
2130 ORIGIN 38+(8*(350/nb)),140:DRA
W 0,ha8,2
2140 ORIGIN 38+k,140:DRAW 0,ha8,2

2150 FOR j=0 TO 350/nb
2160 ORIGIN 39+k+j,140+ha8:DRAW 12,
10,2:NEXT j
2170 IF nb=8 THEN GOTO 3020
2180 ' ****** 9 BARRA *****
2190 FOR c=1 TO ha9
2200 LET k=8*(350/nb)
2210 ORIGIN 38+k,140+c:DRAW (350/nb
),0,1
2220 ORIGIN 38+(9*(350/nb)),140+c:D
RAW 12,10,3:NEXT c
2230 ORIGIN 38+(9*(350/nb)),140+ha9
:DRAW 12,10,1
2240 ORIGIN 38+k,140+ha9:DRAW 12,10
,1
2250 ORIGIN 38+(9*(350/nb)),140:DRA
W 0,ha9,2
2260 ORIGIN 38+k,140:DRAW 0,ha9,2

2270 FOR j=0 TO 350/nb
2280 ORIGIN 39+k+j,140+ha9:DRAW 12,
10,2:NEXT j
2290 IF nb=9 THEN GOTO 3020
2300 ' ***** 10 barra ****
2310 FOR c=1 TO ha10
2320 LET k=9*(350/nb)
2330 ORIGIN 38+k,140+c:DRAW (350/nb
),0,1
2340 ORIGIN 38+(10*(350/nb)),140+c:
DRAW 12,10,3:NEXT c
2350 ORIGIN 38+(10*(350/nb)),140+ha
10:DRAW 12,10,1
2360 ORIGIN 38+k,140+ha10:DRAW 12,1
0,1
2370 ORIGIN 38+(10*(350/nb)),140:DR
AW 0,ha10,2
2380 ORIGIN 38+k,140:DRAW 0,ha10,2

2390 FOR j=0 TO 350/nb
2400 ORIGIN 39+k+j,140+ha10:DRAW 12
,10,2:NEXT j
2410 IF nb=10 THEN GOTO 3020
2420 ' ****** 11 BARRA *****
2430 FOR c=1 TO ha11
2440 LET k=10*(350/nb)
2450 ORIGIN 38+k,140+c:DRAW (350/nb
),0,1
2460 ORIGIN 38+(11*(350/nb)),140+c:
DRAW 12,10,3:NEXT c
2470 ORIGIN 138+(11*(350/nb)),140+h
a11:DRAW 12,10,1
2480 ORIGIN 38+k,140+ha11:DRAW 12,1
0,1
2490 ORIGIN 38+(11*(350/nb)),140:DR
AW 0,ha11,2
2500 ORIGIN 38+k,140:DRAW 0,ha11,2

2510 FOR j=0 TO 350/nb
2520 ORIGIN 39+k+j,140+ha11:DRAW 12
,10,2:NEXT j
2530 IF nb=11 THEN GOTO 3020
2540 ' ***** 12 barra ******
2550 FOR c=1 TO ha12
2560 LET k=11*(350/nb)
2570 ORIGIN 38+k,140+c:DRAW (350/nb
),0,1
2580 ORIGIN 38+(12*(350/nb)),140+c:
DRAW 12,10,3:NEXT c
2590 ORIGIN 38+(12*(350/nb)),140+ha
12:DRAW 12,10,1
2600 ORIGIN 38+k,140+ha12:DRAW 12,1
0,1
2610 ORIGIN 38+(12*(350/nb)),140:DR
AW 0,ha12,2
2620 ORIGIN 38+k,140:DRAW 0,ha12,2

2630 FOR j=0 TO 350/nb
2640 ORIGIN 39+k+j,140+ha12:DRAW 12
,10,2:NEXT j
2650 IF nb=12 THEN GOTO 3020
2660 ' ***** 13 barra ****
2670 FOR c=1 TO ha13
2680 LET k=12*(350/nb)
2690 ORIGIN 38+k,140+c:DRAW (350/nb
),0,1
2700 ORIGIN 38+(13*(350/nb)),140+c:
DRAW 12,10,3:NEXT c
2710 ORIGIN 38+(13*(350/nb)),140+ha
13:DRAW 12,10,1
2720 ORIGIN 38+k,140+ha13:DRAW 12,1
0,1
2730 ORIGIN 38+(13*(350/nb)),140:DR
AW 0,ha13,2
2740 ORIGIN 38+k,140:DRAW 0,ha13,2

2750 FOR j=0 TO 350/nb
2760 ORIGIN 39+k+j,140+ha13:DRAW 12
,10,2:NEXT j
2770 IF nb=13 THEN GOTO 3020
2780 ' **** 14 BARRA *****
2790 FOR c=1 TO ha14
2800 LET k=13*(350/nb)
2810 ORIGIN 38+k,140+c:DRAW (350/nb
),0,1
2820 ORIGIN 38+(14*(350/nb)),140+c:
DRAW 12,10,3:NEXT c
2830 ORIGIN 38+(14*(350/nb)),140+ha
14:DRAW 12,10,1
2840 ORIGIN 38+k,140+ha14:DRAW 12,1
0,1
2850 ORIGIN 38+(14*(350/nb)),140:DR
AW 0,ha14,2
2860 ORIGIN 38+k,140:DRAW 0,ha14,2

2870 FOR j=0 TO 350/nb
2880 ORIGIN 39+k+j,140+ha14:DRAW 12
,10,2:NEXT j
2890 IF nb=14 THEN GOTO 3020
2900 ' ***** 15 BARRA *****
2910 FOR c=1 TO ha15
2920 LET k=14*(350/nb)
2930 ORIGIN 38+k,140+c:DRAW (350/nb
),0,1
2940 ORIGIN 38+(15*(350/nb)),140+c:
DRAW 12,10,3:NEXT c

2950 ORIGIN 38+(15*(350/nb)),140+ha
15:DRAW 12,10,1
2960 ORIGIN 38+k,140+ha15:DRAW 12,1
0,1
2970 ORIGIN 38+(15*(350/nb)),140:DR
AW 0,ha15,2
2980 ORIGIN 38+k,140:DRAW 0,ha15,2

2990 FOR j=0 TO 350/nb
3000 ORIGIN 39+k+j,140+ha15:DRAW 12
,10,2:NEXT j

3010 IF nb=15 THEN GOTO 3020
3020 LOCATE 10,24:PEN 2:PRINT "PULS
A TECLA PARA SEGUIR"
3030 WHILE INKEY$="":WEND
3040 GOTO 2e0
3050 ' *$*$*$*$*$*$*$*MQUESOS *$*$*$*
$*$*$*M
3060
3070 MODE 1:PEN 1
3080 PRINT
3090 INPUT "Dame concepto general";
cg$
3100 PRINT
3110 INPUT "Introduce total";t
3120 PRINT
3130 INPUT "Dame concepto bloque 1"
;c1$
3140 PRINT
3150 INPUT "Dame valor bloque 1";v1
3160 PRINT
3170 IF v1<0 OR v1>t THEN GOTO 3150
3180 INPUT "Dame concepto bloque 2"
;c2$
3190 PRINT
3200 LET v1=(v1*360)/t
3210 INPUT "Dame valor bloque 2";v2

3220 PRINT
3230 IF v2<0 OR v2>t THEN GOTO 3210
3240 INPUT "Dame concepto bloque 3"
;c3$
3250 PRINT
3260 LET v2=(v2*360)/t
3270 INPUT "Dame valor bloque 3";v3
3280 PRINT
3290 IF v3<0 OR v3>t THEN GOTO 3270
3300 LET v3=(v3*360)/t
3310 INPUT "Dame concepto bloque 4"
;c4$
3320 PRINT
3330 PRINT
3340 PRINT " Se asume que el valor
del cuarto bloque es el resto hasta
 el total."
3350 PRINT
3360 INPUT "Si quieres cambiar algo
 pulsa /c/+ENTER";c$
3370 CLS
3380 PRINT
3390 IF c$="c" THEN GOTO 3070
3400 PRINT
3410 IF v1+v2+v3>360 THEN PRINT "
Has sobrepasado el total...Rectific
a  pulsando cualquier  tecla."
3420 PRINT
3430 LOCATE 10,12:PEN 2:PRINT "Puls
a tecla para seguir":PEN 1
3440 WHILE INKEY$="":WEND
3450 IF v1+v2+v3>360 THEN GOTO 3070
3460 INK 0,0:BORDER 0:INK 1,26:INK
2,2:INK 3,6:CLS
3470 LOCATE 2,2:PEN 2:PRINT cg$
3480 LOCATE 13,2:PEN 1:PRINT"  BLOO
UES      %      valor
3490 LOCATE 13,3:PEN 3:PRINT STRING
$(11,CHR$(208));"  ";STRING$(3,CHR$
(208));"    ";STRING$(7,CHR$(208))
3500 LOCATE 14,5:PEN 1:PRINT CHR$(1
43);" "jc1$
3510 LOCATE 14,7:PEN 2:PRINT CHR$(1
43)
3520 LOCATE 14,9:PEN 3:PRINT CHR$(1
43)
3530 LOCATE 14,11:PEN 1:PRINT CHR$(
207)
3540 LOCATE 16,7:PEN 1:PRINT c2$
3550 LOCATE 16,9:PEN 1:PRINT c3$
3560 LOCATE 16,11:PEN 1:PRINT c4$
3570 LOCATE 25,5:PEN 1:PRINT ROUND(
((v1*100)/360),1)
3580 LOCATE 25,7:PRINT ROUND(((v2*1
00)/360),1)
```

```
3590 LOCATE 25,9:PRINT ROUND(((v3*1
00)/360),1)
3600 IF t>9999 THEN j1=0
3610 LOCATE 25,11:PRINT ROUND(100-(
((v1*100)/360)+((v2*100)/360)+((v3*
100)/360)),2)
3620 LOCATE 33,5:PRINT (v1/360)*t
3630 LOCATE 33,7:PRINT (v2/360)*t
3640 LOCATE 33,9:PRINT (v3/360)*t
3650 LOCATE 33,11:PRINT ROUND((t-((
(v1/360)*t)+((v2/360)*t)+((v3/360)*
t)))/2)
3660 LOCATE 13,13:PEN 2:PRINT STRIN
G$(11,CHR$(208));"  ";STRING$(3,CHR
$(208));"    ";STRING$(7,CHR$(208))
3670 LOCATE 26,15:PEN 1:PRINT "100"
3680 LOCATE 33,15:PEN 1:PRINT t
3690 FOR p=1 TO 360 STEP 1.5
3700 DEG
3710 CPIGIN 110,275:PLOT 72*COS(p),
72*SIN(p),1:ORIGIN 150,110:PLOT 152
*COS(p),42*SIN(p),3
3720 ORIGIN 150,98:PLOT 150*COS(p/2
),40*SIN(p/2),3
3730 ORIGIN 150+152*COS(-p/2),98+42
*SIN(-p/2):DRAW 0,18,1
3740 ORIGIN 0,98:DRAW 0,20,1
3750 ORIGIN 300,98:DRAW 0,20,1
3760 NEXT
3770 FOR a= 1 TO v1
3780 DEG
3790 ORIGIN 110,275:DRAW 70*SIN(a),
70*COS(a),1:ORIGIN 150,110:DRAW 150
*SIN(a),40*COS(a),1
3800 NEXT
3810 FOR b=v1 TO v1+v2
3820 DEG
3830 ORIGIN 110,275:DRAW 70*SIN(b),
70*COS(b),2:ORIGIN 150,110:DRAW 150
*SIN(b),40*COS(b),2
3840 NEXT
3850 FOR c=v1+v2 TO v1+v2+v3
3860 DEG
3870 ORIGIN 110,275:DRAW 70*SIN(c),
70*COS(c),3:ORIGIN 150,110:DRAW 150
*SIN(c),40*COS(c),3
3880 NEXT
3890 LOCATE 20,23:PRINT "Pulsa tecl
a para borrar"
3900 WHILE INKEY$="":WEND
3910 GOTO 260
3920 '************ LINEAS *********
*
3930 MODE 1
3940 CLS
3950 INK 0,0:INK 1,13:BORDER 0:INK
2,2:INK 3,6:PEN 1
3960 INPUT "DAME CONCEPTO GENERAL";
nb$
3970 INPUT "DAME CONCEPTO LINEA r";
n1$
3980 INPUT "DAME CONCEPTO LINEA s";
n2$
3990 INPUT "ELIGE N. DE VALORES POR
 LINEA(3-13)";nv
4000 IF nv<3 OR nv>13 GOTO 3990
4010 INPUT "DAME MAXIMO VALOR";mva
4020 IF mva<=0 THEN GOTO 4010
4030 INPUT "DAME VALOR 1 LINEA r";r
1
4040 INPUT "DAME VALOR 1 LINEA s";s
1
4050 INPUT "DAME VALOR 2 LINEA r";r
2
4060 INPUT "DAME VALOR 2 LINEA s";s
2
4070 INPUT "DAME VALOR 3 LINEA r";r
3
4080 INPUT "DAME VALOR 3 LINEA s";s
3
4090 LET s1=(s1*199)/mva
4100 LET r1=(r1*199)/mva
4110 LET s2=(s2*199)/mva
4120 LET r2=(r2*199)/mva
4130 LET s3=(s3*199)/mva
4140 LET r3=(r3*199)/mva
4150 IF nv=3 THEN GOTO 4660
4160 INPUT "DAME VALOR 4 LINEA r";r
4
4170 INPUT "DAME VALOR 4 LINEA s";s
4
4180 LET r4=(r4*199)/mva
4190 LET s4=(s4*199)/mva
4200 IF nv=4 THEN GOTO 4660
4210 INPUT "DAME VALOR 5 LINEA r";r
5
4220 INPUT "DAME VALOR 5 LINEA S";S
5
4230 LET r5=(r5*199)/mva
4240 LET s5=(s5*199)/mva
4250 IF nv=5 THEN GOTO 4660
4260 INPUT "DAME VALOR 6 LINEA r";r
6
4270 INPUT "DAME VALOR 6 LINEA s";s
6
4280 LET r6=(r6*199)/mva
4290 LET s6=(s6*199)/mva
4300 IF nv=6 THEN GOTO 4660
4310 INPUT "DAME VALOR 7 LINEA r";r
7
4320 INPUT "DAME VALOR 7 LINEA s";s
7
4330 LET r7=(r7*199)/mva
4340 LET s7=(s7*199)/mva
4350 IF nv=7 THEN GOTO 4660
4360 INPUT "DAME VALOR 8 LINEA r";r
8
4370 INPUT "DAME VALOR 8 LINEA s";s
8
4380 LET r8=(r8*199)/mva
4390 LET s8=(s8*199)/mva
4400 IF nv=8 THEN GOTO 4660
```



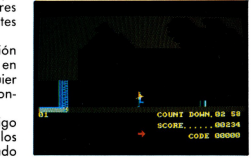
```
4410 INPUT "DAME VALOR 9 LINEA r";r
9
4420 INPUT "DAME VALOR 9 LINEA s";s
9
4430 LET r9=(r9*199)/mva
4440 LET s9=(s9*199)/mva
4450 IF nv=9 THEN GOTO 4660
4460 INPUT "DAME VALOR 10 LINEA r";
r10
4470 INPUT "DAME VALOR 10 LINEA s";
s10
4480 LET r10=(r10*199)/mva
4490 LET s10=(s10*199)/mva
4500 IF nv=10 THEN GOTO 4660
4510 INPUT "DAME VALOR 11 LINEA r";
r11
4520 INPUT "DAME VALOR 11 LINEA s";
s11
4530 LET r11=(r11*199)/mva
4540 LET s11=(s11*199)/mva
4550 IF nv=11 THEN GOTO 4660
4560 INPUT "DAME VALOR 12 LINEA r";
r12
4570 INPUT "DAME VALOR 12 LINEA s";
s12
4580 LET r12=(r12*199)/mva
4590 LET s12=(s12*199)/mva
4600 IF nv=12 THEN GOTO 4660
4610 INPUT "DAME VALOR 13 LINEA r";
r13
4620 INPUT "DAME VALOR 13 LINEA s";
s13
4630 LET r13=(r13*199)/mva
4640 LET s13=(s13*199)/mva
4650 IF nv=13 THEN GOTO 4660
4660 INPUT "PARA RECTIFICAR PULSA (
c)+ENTER";fc$
4670 IF c$="c" THEN 3940
4680 CLS:INK 1,26
4690 FOR f=0 TO 200 STEP 50
4700 ORIGIN 20,199+f:DRAW 400,0,1
4710 ORIGIN 20,200-f:DRAW 400,0,1:N
EXT
4720 FOR s=20 TO 440 STEP (395/(nv+
1))
4730 ORIGIN s,0:DRAW 0,400,1:NEXT
4740 '
4750 LET l=(395/(nv+1))
4760 ORIGIN 25+l,199+r1:DRAW l,r2-r
1,2
4770 ORIGIN 25+l,199+s1:DRAW l,s2-s
1,3
4780 ORIGIN 25+(2*l),199+r2:DRAW l,
r3-r2,2
4790 ORIGIN 25+(2*l),199+s2:DRAW l,
s3-s2,3
4800 IF nv=3 THEN GOTO 5110
4810 ORIGIN 25+(3*l),199+r3:DRAW l,
r4-r3,2
4820 ORIGIN 25+(3*l),199+s3:DRAW l,
s4-s3,3
4830 IF nv=4 THEN GOTO 5110
4840 ORIGIN 25+(4*l),199+r4:DRAW l,
r5-r4,2
4850 ORIGIN 25+(4*l),199+s4:DRAW l,
s5-s4,3
4860 IF nv=5 THEN GOTO 5110
4870 ORIGIN 25+(5*l),199+r5:DRAW l,
r6-r5,2
4880 ORIGIN 25+(5*l).199+s5:DRAW l,
s6-s5,3
4890 IF nv=6 THEN GOTO 5110
4900 ORIGIN 25+(6*l),199+r6:DRAW l,
r7-r6,2
4910 ORIGIN 25+(6*l),199+s6:DRAW l,
s7-s6,3
4920 IF nv=7 THEN GOTO 5110
4930 ORIGIN 25+(7*l),199+r7:DRAW l,
r8-r7,2
4940 ORIGIN 25+(7*l),199+s7:DRAW l,
s8-s7,3
4950 IF nv=8 THEN GOTO 5110
4960 ORIGIN 25+(8*l),199+r8:DRAW l,
r9-r8,2
4970 ORIGIN 25+(8*l),199+s8:DRAW l,
s9-s8,3
4980 IF nv=9  THEN GOTO 5110
4990 ORIGIN 25+(9*l),199+r9:DRAW l,
r10-r9,2
5000 ORIGIN 25+(9*l),199+s9:DRAW l,
s10-s9,3
5010 IF nv=10 THEN GOTO 5110
5020 ORIGIN 25+(10*l),199+r10:DRAW
l,r11-r10,2
5030 ORIGIN 25+(10*l),199+s10:DRAW
l,s11-s10,3
5040 IF nv=11 THEN GOTO 5110
5050 ORIGIN 25+(11*l),199+r11:DRAW
l,r12-r11,2
5060 ORIGIN 25+(11*l),199+s11:DRAW
l,s12-s11,3
5070 IF nv=12 THEN GOTO 5110
5080 ORIGIN 25+(12*l),199+r12:DRAW
l,r13-r12,2
5090 ORIGIN 25+(12*l),199+s12:DRAW
l,s13-s12,3
5100 IF nv=13 THEN GOTO 5110
5110 LOCATE 29,1:PEN 1:PRINT mva
5120 LOCATE 29,3:PEN 1:PRINT mva*(3
/4)
5130 LOCATE 29,6:PEN 1:PRINT mva/2
5140 LOCATE 29,10:PRINT mva/4
5150 LOCATE 29,13:PRINT " 0"
5160 LOCATE 29,16:PRINT -mva/4
5170 LOCATE 29,19:PRINT -mva/2
5180 LOCATE 29,22:PRINT -mva*(3/4)
5190 LOCATE 29,25:PRINT -mva
5200 LOCATE 33,2:PEN 1:PRINT nb$
5210 LOCATE 33,13:PEN 2:PRINT n1$
5220 LOCATE 33,24:PEN 3:PRINT n2$
5230 WHILE INKEY$="":WEND
5240 GOTO 260
5250 '********* BARRAS 2D *********
***
```

```
5260 INK 2,26:INK 3,26
5270 CLS:INK 1,13:PEN 1:INPUT "Intr
oduce concepto de datos( 10  letras
 max.)";cd$:PRINT:INPUT "  Introd
uce maximo valor(sin tener en cuent
a el signo): De ser menor de 1000
aparecera media y total";mv

5280 PRINT
5290 INPUT "Introduce valor Enero";
e
5300 PRINT
5310 INPUT "Introduce valor Febrero
";f
5320 PRINT
5330 INPUT "Introduce valor Marzo";
m
5340 PRINT
5350 INPUT "Introduce valor Abril";
a
5360 PRINT
5370 INPUT "Introduce valor Mayo";m
y
5380 PRINT
5390 INPUT "Introduce valor Junio";
j
5400 PRINT
5410 INPUT "Introduce valor Julio";
jl
5420 PRINT
5430 INPUT "Introduce valor Agosto"
;ag
5440 PRINT
5450 INPUT "Introduce valor Septiem
bre";s
5460 PRINT
5470 INPUT "Introduce valor Octubre
";o
5480 PRINT
5490 INPUT "Introduce valor Noviemb
re";n
5500 PRINT
5510 INPUT "Introduce valor Diciemb
re";d
5520 PRINT
5530 IF mv>1000 THEN PRINT "No apar
ecera media ni total "
5540 PRINT
5550 INPUT "BARRAS(b) O PICOS";gb$

5560 PRINT
5570 INPUT "Para rectificar pulsa (
c)+ENTER";c$
5580 IF c$="C" THEN GOTO 5250
5590 IF sx$="3" THEN GOTO 5630
5600 IF sx$="1" OR sx$="2" THEN GOT
O 6280
5610 END
5620 ' ************** POS Y NEG ***
**************
5630 k!=165
5640  IF mv>k! THEN 5650
5650 LET e=CINT((e*k!)/mv)
5660 LET f=CINT((f*k!/mv))
5670 LET m=CINT((m*k!)/mv)
5680 LET a=CINT((a*k!)/mv)
5690 LET my=CINT((my*k!)/mv)
5700 LET j=CINT((j*k!)/mv)
5710 LET jl=CINT((jl*k!)/mv)
5720 LET ag=CINT((ag*k!)/mv)
5730 LET s=CINT((s*k!)/mv)
5740 LET o=CINT((o*k!)/mv)
5750 LET n=CINT((n*k!)/mv)
5760 LET d=CINT((d*k!)/mv)
5770 CLS:MODE 2
5780 INK 0,2:INK 1,26:BORDER 2
5790 IF mv>1000 GOTO 5820
5800 LOCATE 71,10:PRINT CINT(((e+f+
m+a+my+j+jl+ag+s+o+n+d)/k!)*mv)
5810 LOCATE 71,16:PRINT ROUND((((((
e+f+m+a+my+j+jl+ag+s+o+n+d)*mv)/k!)
/12),3)
5820 LOCATE 70,2:PRINT cd$
5830 LOCATE 70,3:PRINT STRING$(10,C
HR$(208))
5840 LOCATE 74,8:PRINT "total":LOCA
TE 70,7:PRINT "redondeo"
5850 LOCATE 71,14:PRINT "media/mes"
5860 LOCATE 2,1:PRINT 10*(mv/10);"_
":LOCATE 3,3:PRINT 8*(mv/10);" "
5870 LOCATE 3,5:PRINT 6*(mv/10);"_
:LOCATE 3,7:PRINT 4*(mv/10);" "
5880 LOCATE 3,9:PRINT 2*(mv/10);"_"
:LOCATE 3,11:PRINT 0*(mv/10);"_"
5890 LOCATE 3,13:PRINT 2*(mv/(-10))
;"_":LOCATE 3,15:PRINT 4*(mv/(-10))
;"_"
5900 LOCATE 3,17:PRINT 6*(mv/(-10))
;"_":LOCATE 3,19:PRINT 8*(mv/(-10))
;"_"
5910 LOCATE 2,21:PRINT 10*(mv/(-10)
);"_"
5920 ORIGIN 60,60:DRAWR 0,350
5930 ORIGIN 60,225:DRAWR 550,0
5940 LOCATE 12,23:PRINT "E"
5950 LOCATE 17,23:PRINT "F"
5960 LOCATE 22,23:PRINT "M"
5970 LOCATE 27,23:PRINT "A"
5980 LOCATE 32,23:PRINT "M"
5990 LOCATE 37,23:PRINT "J"
6000 LOCATE 42,23:PRINT "j"
6010 LOCATE 47,23:PRINT "A"
6020 LOCATE 52,23:PRINT "S"
6030 LOCATE 57,23:PRINT "O"
6040 LOCATE 62,23:PRINT "N"
6050 LOCATE 67,23:PRINT "D"
6060 LOCATE 60,24:PRINT "
                Pulsa tecla para borrar"
6070 IF gb$="b" OR gb$="B" THEN GOT
O 6100
6080 IF NOT gb$="b" OR NOT gb$="B"
THEN GOTO 6980
6090 END
6100 FOR k=-10 TO 10
6110 LET w=225
6120 ORIGIN 90+k,w:DRAW 0,e
6130 ORIGIN 130+k,w:DRAW 0,f
6140 ORIGIN 170+k,w:DRAW 0,m
6150 ORIGIN 210+k,w:DRAW 0,m
6160 ORIGIN 250+k,w:DRAW 0,my
6170 ORIGIN 290+k,w:DRAW 0,j
6180 ORIGIN 330+k,w:DRAW 0,jl
6190 ORIGIN 370+k,w:DRAW 0,ag
6200 ORIGIN 410+k,w:DRAW 0,s
6210 ORIGIN 450+k,w:DRAW 0,o
6220 ORIGIN 490+k,w:DRAW 0,n
6230 ORIGIN 530+k,w:DRAW 0,d
6240 NEXT
6250 WHILE INKEY$="" :WEND
6260 GOTO 260
6270 '*****************  pos o neg
*****
6280 LET e=CINT((e*330)/mv)
6290 LET f=CINT((f*330/mv))
6300 LET m=CINT((m*330)/mv)
6310 LET a=CINT((a*330)/mv)
6320 LET my=CINT((my*330)/mv)
6330 LET j=CINT((j*330)/mv)
6340 LET jl=CINT((jl*330)/mv)
6350 LET ag=CINT((ag*330)/mv)
6360 LET s=CINT((s*330)/mv)
6370 LET o=CINT((o*330)/mv)
6380 LET n=CINT((n*330)/mv)
6390 LET d=CINT((d*330)/mv)
6400 CLS:MODE 2:IF sx$="2" THEN INK
 0,3:IF sx$="2" THEN BORDER 3:IF s
x$="2" THEN INK 1,26
6410 IF mv>1000 GOTO 6440
6420  LOCATE 71,10:PRINT CINT(((e+f+
m+a+my+j+jl+ag+s+o+n+d)/330)*mv)
6430 LOCATE 71,16:PRINT ROUND(((((e
+f+m+a+my+j+jl+ag+s+o+n+d)*mv)/330)
/12),3)
6440 LOCATE 70,2:PRINT cd$
6450 LOCATE 70,3:PRINT STRING$(10,C
HR$(208))
6460 LOCATE 74,8:PRINT "total":LOCA
TE 70,7:PRINT "redondeo"
6470 LOCATE 71,14:PRINT "media/mes"
6480 IF sx$="2" THEN mv=(-1)*mv
6490 LOCATE 2,1:PRINT 10*(mv/10);"_
":LOCATE 3,3:PRINT 9*(mv/10);"_"
6500 LOCATE 3,5:PRINT 8*(mv/10);"_"
:LOCATE 3,7:PRINT 7*(mv/10);"_"
6510 LOCATE 3,9:PRINT 6*(mv/10);"_"
:LOCATE 3,11:PRINT 5*(mv/10);"_"
6520 LOCATE 3,13:PRINT 4*(mv/10);"_
":LOCATE 3,15:PRINT 3*(mv/10);"_"
6530 LOCATE 3,17:PRINT 2*(mv/10);"_
":LOCATE 3,19:PRINT 1*(mv/10);"_"
6540 ORIGIN 60,60:DRAWR 0,350
6550 ORIGIN 60,60:DRAWR 550,0
6560 LOCATE 12,23:PRINT "E"
6570 LOCATE 17,23:PRINT "F"
6580 LOCATE 22,23:PRINT "M"
6590 LOCATE 27,23:PRINT "A"
6600 LOCATE 32,23:PRINT "M"
6610 LOCATE 37,23:PRINT "J"
6620 LOCATE 42,23:PRINT "j"
6630 LOCATE 47,23:PRINT "A"
6640 LOCATE 52,23:PRINT "S"
6650 LOCATE 57,23:PRINT "O"
6660 LOCATE 62,23:PRINT "N"
6670 LOCATE 67,23:PRINT "D"
6680 LOCATE 60,24:PRINT "
                Pulsa tecla para borrar"
6690 IF gb$="b" OR gb$="B" THEN GOT
O 6710
6700 IF NOT gb$="b" OR NOT gb$="B"
THEN GOTO 6980
6710 FOR k=-10 TO 10
6720 LET w=60
6730 ORIGIN 90+k,w:DRAW 0,e
6740 ORIGIN 130+k,w:DRAW 0,f
6750 ORIGIN 170+k,w:DRAW 0,m
6760 ORIGIN 210+k,w:DRAW 0,a
6770 ORIGIN 250+k,w:DRAW 0,my
6780 ORIGIN 290+k,w:DRAW 0,j
6790 ORIGIN 330+k,w:DRAW 0,jl
6800 ORIGIN 370+k,w:DRAW 0,ag
6810 ORIGIN 410+k,w:DRAW 0,s
6820 ORIGIN 450+k,w:DRAW 0,o
6830 ORIGIN 490+k,w:DRAW 0,n
6840 ORIGIN 530+k,w:DRAW 0,d
6850 ORIGIN 530,w:DRAW k,d
6860 NEXT
6870 WHILE INKEY$="" :WEND
6880 GOTO 260
6890 '********** PICOS **********
*****
6900 LET k=0
6910 IF sx$="1" OR sx$="2" THEN LET
 w=60
6920 IF sx$="3" THEN LET w=225
6930 FOR bg=-10 TO 10
6940 ORIGIN 90+k,w:DRAW bg,e
6950 ORIGIN 130+k,w:DRAW bg,f
6960 ORIGIN 170+k,w:DRAW bg,m
6970 ORIGIN 210+k,w:DRAW bg,a
6980 ORIGIN 250+k,w:DRAW bg,my
6990 ORIGIN 290+k,w:DRAW bg,j
7000 ORIGIN 330+k,w:DRAW bg,jl
7010 ORIGIN 370+k,w:DRAW bg,ag
7020 ORIGIN 410+k,w:DRAW bg,s
7030 ORIGIN 450+k,w:DRAW bg,o
7040 ORIGIN 490+k,w:DRAW bg,n
7050 ORIGIN 530+k,w:DRAW bg,d
7060 NEXT
7070 IF sx$="3" THEN GOTO 6250
7080 IF sx$="1" OR sx$="2" THEN GOT
O 6870
```



A VIEW TO A KILL

A View To A Kill, vive las emociones de la última aventura de James Bond en tres escenarios diferentes; París, Silicon Valley y The City Hall en San Francisco.



La música de Duran-Duran invade el ambiente, llenando nuestros oídos;

Dance into the fire... The final kiss is always me... so dance into the fire.

Mientras el inefable Roger Moore, en una de sus mejores actuaciones de toda la historia del 007, vive y deja morir. James Bond, ama a las mujeres más bellas, conduce los mejores deportivos y lucha contra las mentes más retorcidas.

Todo desarrollado en una acción trepidante, en la que su vida está en constante peligro, y donde cualquier agente de la CIA o del KGB, encontraría una muerte segura.

En esta aventura nuestro enemigo es el temible Max Zorin, uno de los más grandes magnates del mercado europeo de la microelectrónica.

Max, pretende hacerse con la hegemonía del mercado de los ordenadores, pero sin seguir el notable ejemplo de Tramiel (*presidente de Atari*), cuya política es la reducción notable de los precios, sino de una forma mucho más drástica y efectiva; eliminando a la competencia.

Max ha tomado lo de eliminar a la competencia en su sentido más elemental, no pretende arruinarles ni quitarles el mercado, su objetivo destruir la primera factoría mundial de chips, con una bomba nuclear.

El juego de ordenador está dotado de multicarga, lo que permite que su acción se desarrolle en tres escenarios diferentes: The Silicon Valley, City Hall y las calles de París.

Compatible CPC 464/CPC 664

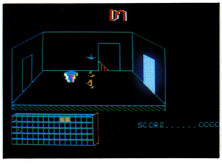
The Silicon Valley

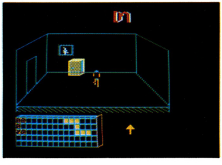
James Bond, descubre que Max pretende destruir el Silicon Valley, colocando una bomba nuclear en una mina abandonada.

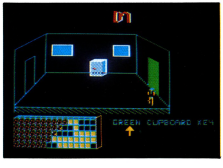
Al estar el Silicon Valley situado sobre una falla geológica, una explosión de este calibre haría que toda la zona se precipitase al Pacífico, consiguiendo Max de esta forma tan sencilla, hacerse con el mercado internacional del chip.

Para destactivar ésta, 007 debe intentar descubrir en los cientos de galerías, el lugar donde ha sido colocada la bomba y desactivarla.

James tiene que contar con la ayuda de May Day, papel encarnado por la supermaciza Grace Jones, adepta a Max y que caerá fácilmente ante el atractivo irresistible de James.







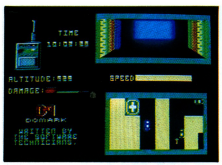
Solamente con su ayuda podremos localizar y desactiva el mecanismo de relojería que detonará la bomba.

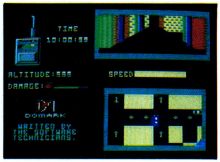
En nuestro recorrido por las distintas galerías encontraremos diversos objetos que nos ayudarán en nuestra misión: pistola lanza cabos, con ella podemos fijar un extremo de la cuerda de escalar a cualquier pared, moviéndonos de esta manera por sitios que serían inalcanzables si no la lleváramos encima.

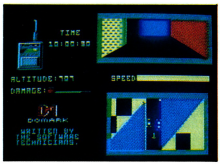
Dinamita y detonadores, utilizándolos en los sitios adecuados, podemos eliminar rocas y otros obstáculos que impiden nuestro paso a otras cavidades de los subterráneos.

Números código, con ellos podremos desactivar la bomba, pero de todos los que se encuentran dispersos en los corredores, solamente sirven cinco y en un determinado orden, descubrirlo es tarea de un gran agente secreto.

Indicador direccional, en nuestro recorrido por las profundidades disponemos de un indicador de la posición de la bomba, éste aparece en la parte inferior de la pantalla y está representado por una flecha roja.







City Hall

Cumplida con éxito la primera misión, 007 viaja a San Francisco en busca de Max Zorin, gracias a la información obtenida a base de seducción de May Day, James le localiza en el City Hall, pero el astuto Max, consigue encerrar a 007 y a su nueva compañera Stacey en el ascensor, donde quedan atrapados entre dos plantas. Después de esto prende fuego a los sótanos del edificio y huye dejando a nuestro protagonista y a su bella compañera en verdadero peligro.

Mr. Joystick

Las calles de París

May Day, después de asesinar a un agente amigo de Bond, se lanza en paracaídas de lo alto de la torre Eiffel, 007 baja de la torre donde tomaba el almuerzo con el agente, que ya no tiene problemas de ningún tipo, y coge un taxi en persecución del paracaídas.

Esta se realiza por las calles de París, repletas de tráfico y con policía municipal que no permite infracciones al código. Su misión es atrapar a May Day en el momento del aterrizaje.

Un programa realizado aprovechando la fama de la película, sin grandes maravillas, pero que tiene el aliciente de que está dividido en tres partes, cada una de las cuales se puede jugar por separado, siendo cualquiera de ellas completamente distinta de las otras.

Tres juegos por el precio de uno.

CEDRIC

Cedric es un sencillo programa, bellamente presentado, que nos introduce en uno de los juegos de inteligencia y educación más conocidos, en el que se trata de ejercitar la memoria y retentiva visuales.

La historia del juego es como sigue:

Cedric, el inevitable personaje, ha perdido sus juguetes, mejor dicho, ha olvidado dónde está cada uno de ellos diseminados en un grupo de cajas.

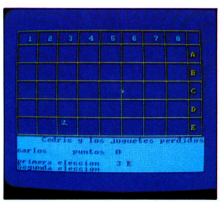
Nosotros, inevitablemente, también debemos ayudarle a encontrarlos todos mediante el simple método de escoger una casilla dentro de un tablero que aparece en la pantalla y que representa todas las cajas en las que están distribuidos los juguetes de Cedric.

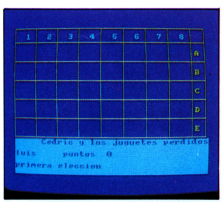
Se juega entre dos personas y se trata de recordar dónde van apareciendo los jueguetes de forma tal, que cuando uno se descubre debemos decirle al ordenador en qué casilla se encuentra su pareja, idéntica a él.

La elección de la casilla se realiza tecleando un número y una letra, de forma similar al conocido juego de los barquitos.

Naturalmente, gana el que más juguetes encuentra hasta el final del juego.

COMPATIBLE
CPC 464
CPC 664





ESTRUCTURA DEL PROGRAMA

LINEAS	COMETIDO
40	Selección de colores.
50-470	Definición de caracteres.
480-690	Definición de gráficos.
700	Factor aleatorio.
710	Títulos.
720-790	Lugares de los juguetes en la matriz.
800-820	Comienzo del juego?
840-850	Pide los nombres de los jugadores.
860-1010	Dibuja tablero.
1030	Letras y números del tablero.
1050-1070	Bucle principal.
1080-1150	Fin del juego.
1160-1410	Pide coordenadas y muestra los juguetes.
1420-1440	Fin del turno.
1450-1490	Borra gráficos.
1500-1540	Mueve a las coordenadas correctas.
1550-1760	Examina matrices e imprime juguetes.
1770-1870	Títulos e instrucciones.
1890-1930	Sonido.
1940-1960	Actualiza puntuaciones.

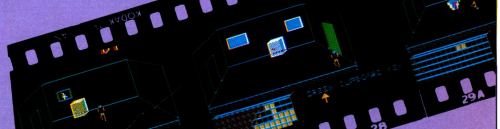
```
10 REM ** Ce
dric **
20 REM ** by Steve Lucas **
30 REM (c) Amstrad Semanal
40 MODE 1:INK 0,1:INK 1,24:INK 2,20
:INK 3,6
50 REM ** define caracteres **
60 SYMBOL AFTER 200
70 SYMBOL 215,128,128,255,3,255,128
,128,128
80 SYMBOL 216,1,1,255,64,255,1,1,1
90 SYMBOL 217,156,156,144,144,144,2
55,128
100 SYMBOL 218,0,0,0,0,0,0,128,128
110 SYMBOL 219,7,3,1,1,1,1,3,7
120 SYMBOL 220,240,224,192,192,192,
192,224,240
130 SYMBOL 221,15,24,63,5,13,9,18,2
0
140 SYMBOL 222,248,140,254,40,236,3
6,18,10
150 SYMBOL 223,14,31,55,127,63,3,14
,0
160 SYMBOL 224,0,8,152,248,152,24,8
,0
170 SYMBOL 225,1,1,1,1,1,1,39,31
180 SYMBOL 226,192,192,192,192,192,
196,245,254
190 SYMBOL 227,15,63,255,25,31,25,3
1,31
200 SYMBOL 228,240,252,255,216,248,
248,216,216
210 SYMBOL 229,0,96,112,88,204,252,
0,192
220 SYMBOL 230,103,159,89,185,95,18
7,44,71
230 SYMBOL 231,230,249,154,157,250,
221,52,226
240 SYMBOL 232,192,128,156,191,255,
255,65,113
250 SYMBOL 233,1,255,255,255,255,0,
0,0
260 SYMBOL 234,24,24,48,255,255,20,
20,60
270 SYMBOL 235,0,2,2,250,254,48,40,
120
280 SYMBOL 236,0,0,15,11,15,255,127
,63
290 SYMBOL 237,128,128,240,240,240,
255,255,255
300 SYMBOL 238,0,254,222,142,222,25
4,254,12
310 SYMBOL 239,0,15,9,9,127,127,127
,48
320 SYMBOL 240,0,0,255,253,253,220,
20,60
330 SYMBOL 241,136,112,127,127,127,
14,10,30
340 SYMBOL 242,192,224,224,224,248,
204,14,174
350 SYMBOL 243,1,3,3,3,15,25,56,58
360 SYMBOL 244,255,133,255,5,255,25
5,12,12
370 SYMBOL 245,255,136,255,136,255,
255,24,24
380 SYMBOL 246,0,160,224,160,235,25
5,63,63
390 SYMBOL 247,0,2,3,2,127,255,170,
254
400 SYMBOL 248,240,156,248,72,72,16
,144,224
410 SYMBOL 249,7,28,15,9,9,4,4,3
420 SYMBOL 250,0,0,192,48,252,255,2
4,24
430 SYMBOL 251,0,0,15,24,127,225,24
,24
440 SYMBOL 252,0,0,0,8,232,252,8,8
450 SYMBOL 253,0,128,143,241,255,12
7,3,2
460 SYMBOL 254,128,224,0,0,255,254,
252,248
470 SYMBOL 255,1,1,1,1,255,127,63,3
1
480 REM ** define los grapicos de l
os juguetes**
490 aa$=CHR$(255)+CHR$(254)
500 ab$=CHR$(253)+CHR$(252)
510 ac$=CHR$(251)+CHR$(250)
520 ad$=CHR$(245)+CHR$(244)
530 ae$=CHR$(243)+CHR$(242)
540 af$=CHR$(249)+CHR$(248)
550 ag$=CHR$(247)+CHR$(246)
560 ah$=CHR$(241)+CHR$(240)
570 ai$=CHR$(239)+CHR$(238)
580 aj$=CHR$(236)+CHR$(237)
590 ak$=CHR$(234)+CHR$(235)
600 al$=CHR$(232)+CHR$(229)
610 am$=CHR$(230)+CHR$(231)
620 an$=CHR$(227)+CHR$(228)
630 ao$=CHR$(225)+CHR$(226)
640 ap$=CHR$(223)+CHR$(224)
650 ai$=" "
660 aq$=CHR$(221)+CHR$(222)
670 ar$=CHR$(219)+CHR$(220)
680 as$=CHR$(217)+CHR$(218)
690 at$=CHR$(215)+CHR$(216)
700 RANDOMIZE TIME
710 GOSUB 1780:REM ** instrucciones
 **
720 REM ** OBJECTOS A LA MATRIZ **
730 DIM ob%(10,10)
740 z%=1
750 FOR y=1 TO 8:FOR x=1 TO 5
760 a=INT(RND(1)*5)+1:b=INT(RND(1)*
8)+1
770 IF ob%(a,b)<>0 THEN 760
780 ob%(a,b)=z%:z%=z%+1:IF z%>20 TH
EN z%=1
790 NEXT x,y
800 REM ** comienzo del juego **
810 PRINT:PRINT"   Pulsa (Space) pa
ra empezar."
820 a$=INKEY$:IF a$<>" "THEN 820
830 REM ** nombre de los jugadores
 **
840 CLS:PEN 1:INPUT"Nombre del juga
dor 1          ";name1$:PRINT CHR$(7
)
850 INPUT"Nombre del jugador 2
  ";name2$:PRINT CHR$(7)
860 REM ** dibuja el tablero **
870 CLS
880 ORIGIN 1,100
890 FOR x=0 TO 8
900 MOVE x*64,1: DRAWR 0,350,1
910 NEXT x
920 WINDOW #1,1,35,20,25:PAPER #1,2
:CLS#1
930 MOVE 0,299:DRAWR 550,0,1
940 TAG
950 MOVE 0,260:DRAWR 550,0,1
960 MOVE 550,0:DRAWR 0,299,1
970 FOR y=0 TO 4
980 MOVE 0,y*52: DRAWR 550,0,1
990 MOVER -30,30:PRINT CHR$(69-y);
1000 NEXT
1010 TAGOFF
1020 REM ** nombres de los ejes **
1030 PEN 2:FOR x= 0 TO 7: LOCATE x*
4+2,2:PRINT x+1;:NEXT x
1040  REM ** bucle principal....repe
tido hasta que se encuentren todos
 los juguetes**
1050 WHILE j%<20
1060 GOSUB 1170
1070 WEND
1080 REM ** fin **
1090 CLS#1:PEN#1,0:PRINT#1,"Has enc
ontrado todos los objetos.Bien hech
o."
1100 PEN #1,3:PRINT#1,name1$;" hall
ados ";i1%;" juguetes"
1110 PRINT#1,name2$;" found ";i2%;"
 toys"
1120 PRINT#1:PRINT#1,"  Pulsa (Spac
e) para continuar"
1130 aa$=INKEY$:IF aa$<>" " THEN 11
30
1140 RUN
1150 END
1160 REM ** coordenadas de la casil
la elegida **
1170 CLS#1:PEN #1,0:PRINT#1,SPC(5);
"Cedric y los juguetes perdidos":PE
N #1,3
1180 PEN #1,0:IF p1%=0 THEN PRINT#1
,name1$;SPC(5);"puntos ";i1% ELSE P
```

VARIABLES

NOMBRE	SIGNIFICADO
name1$, name2$	Nombres de los jugadores.
s1%, s2%	Puntuaciones de ambos.
j%	Número de juguetes encontrados.
z%	Usado como un flag (indicador).
aa$–at$	Gráficos para los jugadores.
ob% (x, y)	Matriz que almacena los juguetes.
a, b	Distribución aleatoria de los juguetes.
x, y	Coordenadas para los gráficos.
t$, s$, t, s	Celdillas escogidas por los jugadores.
fa%	Juguete encontrado?
qa, qb, pa, pb	Coordenadas de los juguetes visibles en pantalla.

```
RINT#1,name2$;SPC(5);"puntos ";s2%
1190 PRINT#1
1200 PRINT#1,"primera eleccion    ";
:PEN #1,3
1210 s$=INKEY$:IF s$="" THEN 1210
1220 s$=UPPER$(s$):IF ASC(s$)>56 OR
 ASC(s$)<49 THEN 1210
1230 PRINT#1,s$;" ";
1240 t$=INKEY$:IF t$="" THEN 1240
1250 t$=UPPER$(t$):IF ASC(t$)>69 OR
 ASC(t$)<65 THEN 1240
1260 PRINT#1,t$
1270 GOSUB 1510
1280 IF fa%=1 THEN CLS#1:GOTO 1170:
REM ** mira si ya se ha elegido **
1290 p=ob%(t,s):pb=t:pa=s
1300 PEN #1,0:PRINT#1,"Enter your s
econd guess  ";:PEN #1,3
1310 s$=INKEY$:IF s$="" THEN 1310
1320 s$=UPPER$(s$):IF ASC(s$)>56 OR
 ASC(s$)<49 THEN 1310
1330 PRINT#1,s$;" ";
1340 t$=INKEY$:IF t$="" THEN 1340
1430 CLS#1:LOCATE #1,1,2:PRINT#1,"P
ulsa (Space) para continuar."
1440 a$=INKEY$:IF a$<>" " THEN 1440
1450 REM ** borra grapicos **
1460 LOCATE 2*(pa-1)*4,5+(pb-1)*3:P
RINT"  ";
1470 LOCATE 2*(qa-1)*4,5+(qb-1)*3:P
RINT"  ";
1480 p1%=p1%+1:IF p1%>1 THEN p1%=0
1490 RETURN
1500 REM ** pinta grapicos en las c
oordenadas correctas **
1510 t=ASC(t$)-64:s=ASC(s$)-48
1520 fa%=0
1530 LOCATE 2*(s-1)*4,5+(t-1)*3
1540 IF ob%(t,s)=0 THEN fa%=1:PRINT
 CHR$(7):RETURN
1550 REM ** mira el contenido de la
 matriz para ver que juguete esta a
ll! **
1560 IF ob%(t,s)=1 THEN PRINT aa$
1570 IF ob%(t,s)=2 THEN PRINT ab$
1580 IF ob%(t,s)=3 THEN PRINT ac$
1590 IF ob%(t,s)=4 THEN PRINT ad$
1600 IF ob%(t,s)=5 THEN PRINT ae$
1610 IF ob%(t,s)=6 THEN PRINT af$
1620 IF ob%(t,s)=7 THEN PRINT ag$
1630 IF ob%(t,s)=8 THEN PRINT ah$
1640 IF ob%(t,s)=9 THEN PRINT ai$
1650 IF ob%(t,s)=10 THEN PRINT aj$
1660 IF ob%(t,s)=11 THEN PRINT ak$
1670 IF ob%(t,s)=12 THEN PRINT al$
1680 IF ob%(t,s)=13 THEN PRINT am$
1690 IF ob%(t,s)=14 THEN PRINT an$
1700 IF ob%(t,s)=15 THEN PRINT ao$
1710 IF ob%(t,s)=16 THEN PRINT ap$
1720 IF ob%(t,s)=17 THEN PRINT aq$
1730 IF ob%(t,s)=18 THEN PRINT ar$
1740 IF ob%(t,s)=19 THEN PRINT as$
1750 IF ob%(t,s)=20 THEN PRINT at$
1760 RETURN
1770 REM ** titulos e instrucciones
 **
1780 CLS:WINDOW #2,4,36,1,3:PEN #2,
3:PAPER #2,1:CLS#2
1790 LOCATE #2,6,2:PRINT#2,"Cedric
```

```
y los juguetes perdidos"
1800 LOCATE 6,6:PRINT"Un juego para
 Amstrad CPC464/664"
1810 LOCATE 11,8:PEN 2:PRINT"por  S
teve W. Lucas"
1820 LOCATE 2,10:PEN 1:PRINT"Cedric
 ha perdido sus juguetes y no sabed
onde encontrarlos."
1830 PRINT"Puedes ayudarle? Estan e
scondidos en el tablero ,el cual es
ta etiquetado de la Aa la E y del 1
al 8."
1840 PRINT"Tecleando las coordenada
s (numero prime-ro) podras ver que
juguete hay en una casilla.";
1850 PRINT"Si los dos juguetes son
el mismo, obten-dras un punto y amb
os permaneceran en la pantalla. De
lo contrario, se borraran."
1860 GOSUB 1890
1870 RETURN
1880 REM ** sonido  **
1890 RESTORE:FOR x=1 TO 6
1900 READ d:SOUND 1,d,d,10
1910 NEXT
1920 DATA 478,478,426,478,358,426
1930 RETURN
1940 REM ** actualiza puntuacioni s
i se ha acertado **
1950 IF p1%=0 THEN s1%=s1%+1 ELSE s
2%=s2%+1
1960 RETURN
1350 t$=UPPER$(t$):IF ASC(t$)>69 OR
 ASC(t$)<65 THEN 1340
1360 PRINT#1,t$
1370 GOSUB 1510
1380 IF fa%=1 THEN CLS#1:GOTO 1300
:REM ** mira si ya se eligio antes
**
1390 q=ob%(t,s):qb=t:qa=s
1400 IF qa=pa AND qb=pb THEN PRINT
CHR$(7):GOTO 1300
1410 IF p=q THEN CLS#1:j%=j%+1:ob%(
qb,qa)=0:ob%(pb,pa)=0:GOSUB 1890:GO
SUB 1950:RETURN
1420 REM ** fin del turno **
```



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

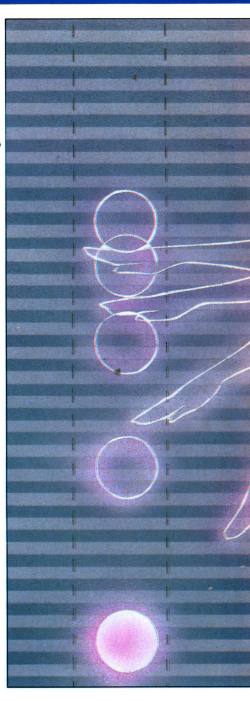
... TAMBIEN PERIFERICOS PARA AMSTRAD

Productos distribuidos por LSB S.A.
c/ Sanchez Pacheco, 78 - 28002 Madrid - Tf.: 413 92 68

Segunda parte de la serie Animación en Basic para ***Amstrad***

ANIMACION EN COLOR

J. Igual



Anteriormente vimos cómo LOCATE y PRINT pueden utilizarse para hacer animaciones sencillas. Ahora vamos a concentrarnos en el tipo de animación que el usuario puede obtener utilizando el comando INK.

Como usted ya sabe, el CPC 464 tiene tres modos gráficos diferentes —0,1 y 2— cada uno de ellos con sus características propias. Por ejemplo, MODE 0 puede tener hasta 16 colores diferentes al mismo tiempo, MODE 1 tiene cuatro colores y MODE 2 tiene dos colores.

Es, precisamente, el muestrario de colores disponibles. Cada uno puede utilizar cualquier combinación de los 27 colores expuestos en la Tabla I.

No obstante, a pesar de que pueda utilizarse cualquiera de los 27 colores en cada uno de los modos, el máximo número de colores simultáneos en la pantalla es constante para cada modo. Lo que se hace es elegir 2, 4 ó 16 colores de una paleta de 27.

Y no sólo puede elegir qué colores tiene en pantalla, sino que puede intercambiar cualquier color por otro. Por ejemplo, si tiene un círculo azul en la pantalla puede cambiarlo a rojo con un sencillo comando Basic:

```
INK X, Y
```

donde X es el número del fondo o de la pluma (paper o pen) que se va a cambiar (azul en el ejemplo) e Y es el nuevo color (rojo en el ejmplo).

Todos los programas de lo ejemplos de este mes utilizan MODE 0, el modo de los 16 colores.

Advertirá que al final de los tres primeros programas están los comandos.

```
MODE 1
CALL &BC02
PEN 1
PAPER 0
END
```

Pulsando la barra espaciadora, todos los colores vuelven a su estado original al conectar el ordenador. Esto es nacesario porque los programas de los ejmplos cambian los colores (tanto de PEN como de PAPER) al color del fondo, azul. Y escribiendo en azul sobre fondo azul ¡hace muy difícil la lectura!

Programa uno

El programa I salva la situación. Coloca 10 bolas escritas con diferentes plumas *(PEN)* en medio de la pantalla. Lo hace con las líneas 60 a 110.

Sin embargo, si se para el programa en la línea 110 nunca podrá verlas. Cada bola es invisible porque la línea 80 cambia de color en las plumas 1 a 10 *(PEN 1 a PEN 10)* haciéndole azul *(INK 1)*. Las bolas se dibujan una a continuación de otra, la primera con PEN 1, la segunda con PEN 2 y así sucesivamente.

Como cada una de estas plumas *(PEN)* se ha rellenado con tinta azul, y puesto que el color del fondo es azul, no aparecerá nada en la pantalla. Las bolas se confunden con él, pero, creednos, están allí.

Las líneas 120 a 190 son las responsables de la animación. La llevan a cabo enseñando una bola, escondiéndola y enseñando la siguiente.

La variable **«numbola»** contine el número de la bola que está saliendo en pantalla, y su valor está comprendido entre 1 y 10. 1 es la que está colocada más a la izquierda y 10 la que está más a la derecha.

El programa selecciona la primera bola inicializando la varialbe **«numbola»** a 1. Esta es la bola, indicada por la variable **«numbola»**, cuyo color cambia a magenta. Y ocurre porque la tinta de PEN 1 se cambia por tinta magenta, INK 1,4.

Cuando la bola fue dibujada original con la pluma 1 *(PEN 1)* se rellenó con tinta azul. Ahora la línea 140 la rellena con tinta magenta. El resultado es que vemos una bola magenta.

Dese cuenta cómo el bucle formado por las líneas 120 a 160, todas las demás bolas están a la derecha y con el mismo color del fondo, azul.

Sólo se cambia la tinta en la primera pluma M*(PEN)* y por lo tanto sólo la primera bola aparece en magenta.

La línea 150 proporciona un corto retardo y la línea 170 comprueba si se está pulsando la barra espaciadora. Cuando esto ocurre el programa termina y los colores se restablecen *(líneas 200 a 230)*. Si no es así la línea 180 selecciona la siguiente bola que aparecerá, incrementando en **«numbola»**. Esta nueva bola está colocada a la derecha de la anterior.

La línea 190 comprueba si la anterior bola está en la última posición de la fila. Si es así se comienza de nuevo por la primera bola cambiando **«numbola»** a 1 *(GOTO 120)*.

Recuerde, sólo se ve una bola en la pantalla, debido al bucle FOR... NEXT. La línea 140 se asegura de ello haciendo magenta la bola correspondiente al valor de **«numbola»** y volviendo azules todas las demás.

Si lo piensa, sólo una de las bolas necesita cambiarse desde el magenta al color del fondo. En cualquier caso sigue siendo una programación sencilla.

El resultado final de todo esto es una bola cruzando la pantalla de izquierda a derecha. En realidad, está viendo diez bolas, una detrás de otra, en lugares ligeramente desplazados. Como las bolas hacen un flash en magenta y vuelven al color del fondo una tras otra, los ojos se engañan y parece que solo hay una bola.

¡Es bastante curioso considerar el hecho de que la parte de animación se realiza sin la instrucción PRINT en la pantalla!

Para mover la bola de derecha a izquierda cambie las líneas siguientes:

```
120 numbola=10
180 numbola=numbola—1
190 IF numbola< THEN GOTO 120
                ELSE GOTO 130
```

N.º de tinta	Color
0	Negro
1	Azul
2	Azul brillante
3	Rojo
4	Magenta
5	Malva
6	Rojo brillante
7	Púrpura
8	Magenta brillante
9	Verde
10	Cyan
11	Azul cielo
12	Amarillo
13	Blanco
14	Azul pastel
15	Naranja
16	Rosa
17	Magenta pastel
18	Verde brillante
19	Verde mar
20	Cyan brillante
21	Verde lima
22	Verde pastel
23	Cyan pastel
24	Amarillo brillante
25	Amarillo pastel
26	Blanco brillante

Tabla 1 **Colores de tinta**

Programa dos

El programa II usa la instrución INK de forma parecida al progama I. En este caso cambia el color de los sectores de un círculo. El resultado es un sector que gira —más bien parece un molino de viento con un solo aspa.

Las líneas 50 a 100 dibujan un círculo en medio de la pantalla. El círculo está formado por 10 sectores, cada uno de un color diferente. Los sectores numerados del 1 al 10 están colocados en el sentido de las agujas del reloj en intervalos de 36 grados desde la parte superior del círculo. La variable **«numsector»** indica cuándo debe aparecer en pantalla cada uno de los 10 sectores —algo parecido a **«numbola»** en el programa I.

Todos los sectores restantes se cambian al color del fondo, azul.

La línea 110 inicializa «numsector» a 1, coloreando el primer sector, y las líneas 120 a 140 devuelven a todos los sectores el color del fondo exceptuando el indicado por **«numsector»**.

La línea 150 examina la barra espaciadora. Si se pulsa finalizará el programa.

La línea 160 aumenta el número de sector que ha aparecido en pantalla. La línea 170 comprueba si han aparecido los 10 sectores. Si han aparecido, se selecciona el primero otra vez y el programa salta a la línea 110. Si no han aparecido el salto se hace a la línea 120.

Como se puede ver el resultado es impresionante.

Programa tres

Teclee el programa III y pulse RUN. Esta es otra demostración del intercambio de colores de la paleta. Queremos decir que cambiaremos los colores de las tintas (INK) en las las plumas *(PEN)*.

El programa dibuja una gran estrella con pluma (PEN) 1. Después dibuja otra estrella, esta vez con PEN 2, gira dos grados en sentido de las agujas del reloj. Esto se repite hasta dibujar 10 estrellas diferentes, utilizando una pluma diferente para cada una de ellas.

Cuando se ha completado el dibujo las estrellas aparecen en pantalla utilizando el mismo procedimiento que en los programas I y II.

Las tintas rellenan cada pluma por turno simultaneando entre las tintas 4 y 1 —*esto es, entre el magenta y el color del fondo*.

Le dejamos que descubra cómo funciona el programa. Es muy similar a los otros dos, de modo que no encontará ninguna dificultad. Una vez metido de lleno en el programa, puede hacer que las estrellas giren en sentido contrario agujas del reloj.

Programa cuatro

Las funciones seno *(SIN)* y coseno *(COS)* usan mucho tiempo calculando sus resultados. Los mismos calculos usando estas tediosas funciones se han repetido en el programa original. Por eso es tan lento.

Este problema se ha solucionado en la línea 100 que es la principal responsable del aumento de velocidad.

Y esto es todo por el momento. Le hacemos unas sugerencias: ¿Por qué no intenta cambiar los programas I a III? ¿Debe ser siempre azul el color del fondo? ¿Deben ser siempre magenta las bolas? ¿Y por qué no tiene dos bolas de diferentes colores yendo en sentidos contrarios?

PROGRAMAS

```
10 REM ******************************

20 REM            PROGRAMA I

30 REM ******************************
****
40 ON ERROR GOTO 200
50 MODE 0
60 LOCATE 1,14
70 FOR x=1 TO 10
80 INK x,1
90 PEN x
100 PRINT CHR$(231);CHR$(32);
110 NEXT
120 numbola=1
130 FOR bola=1 TO 10
140 IF bola=numbola THEN INK bola,4
 ELSE INK bola,1
150 FOR retardo=1 TO 10: NEXT retar
do
160 NEXT bola
170 IF INKEY(47)=0 THEN GOTO 200
180 numbola=numbola+1
190 IF numbola>10 THEN GOTO 120 ELS
E GOTO 130
200 MODE 1
210 CALL &BC02
220 PEN 1
230 PAPER 0
240 END
```

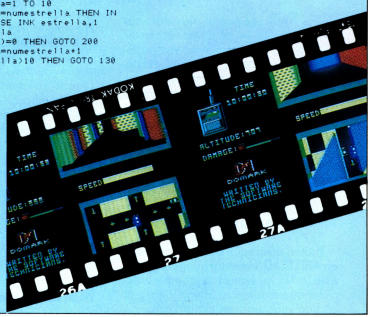
```
10 REM ******************************

20 REM            PROGRAMA II

30 REM ******************************
****
40 ON ERROR GOTO 180
50 DEG
60 MODE 0
70 FOR angulo=0 TO 359
80 MOVE 320,200
90 DRAW 200+SIN(angulo)*320,175+COS
(angulo)*200,INT(angulo/36)+1
100 NEXT
110 numsector=1
120 FOR sector=1 TO 10
130 IF sector=numsector THEN INK se
ctor,4 ELSE INK sector,1
140 NEXT sector
150 IF INKEY(47)=0 THEN GOTO 180
160 numsector=numsector+1
170 IF numsector>10 THEN GOTO 110 E
LSE GOTO 120
180 MODE 1
190 CALL &BC02
200 PEN 1
210 PAPER 0
220 END
```

```
10 REM ******************************

20 REM            PROGRAMA III

30 REM ******************************

40 ON ERROR GOTO 200
50 DEG
60 MODE 0
70 FOR fuga=0 TO 9
80 FOR angulo=fuga*2 TO 359 STEP 20
90 MOVE 320,200
100 DRAW 200+SIN(angulo)*320,175+CO
S(angulo)*200,fuga+1
110 NEXT angulo
120 NEXT fuga
130 numestrella=1
140 FOR estrella=1 TO 10
150 IF estrella=numestrella THEN IN
K estrella,4 ELSE INK estrella,1
160 NEXT estrella
170 IF INKEY(47)=0 THEN GOTO 200
180 numestrella=numestrella+1
190 IF numestrella>10 THEN GOTO 130
 ELSE GOTO 140
```

```
200 MODE 1
210 CALL &BC02
220 PEN 1
230 PAPER 0
240 END
```

```
10 REM ******************************

20 REM            PROGRAMA IV

30 REM ******************************
****
40 MODE 0
50 CALL &BC02
60 DEG
70 ORIGIN 320,200
80 MOVE 0,100
90 FOR i%=0 TO 360
100 s=SIN(i%):c=COS(i%)
110 PLOT s*150,c*150,i% MOD 15+1:PL
OT s*75,c*150:PLOT s*150,c*75
120 NEXT
130 FOR i%=0 TO 15
140 INK i%,0
150 NEXT
160 WHILE 1=1
170 FOR i%=1 TO 15
180 INK i%,26
190 FOR x=1 TO 100:NEXT
200 INK i%,0
210 NEXT
220 WEND
```

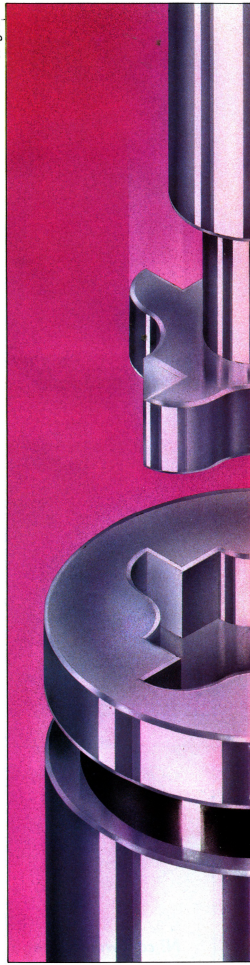
Para que tus dedos, no realicen el trabajo duro, M.H. AMS TRAD lo hace por ti. Todos los listados que incluyan este logotipo se encuentran a tu disposición en un cassette mensual, solicítanoslo.



HERRAMIENTAS DEL PROGRAMADOR

J. Igual

A lo largo de esta serie de artículos hemos intentado revisar y aprender, paso a paso, unas cuantas técnicas y particularidades del código máquina. Sabemos cargar y manipular registros, llamar a subrutinas de máquina desde un programa, indexar zonas de memoria usando los registros dobles y un montón de cosas.



Tal vez sea el momento de hacer un alto en el aprendizaje de nuevas técnicas, practicar exhaustivamente lo que ya sabemos, hasta dominarlo, y hacer uso de las herramientas que todo programador en código máquina debe tener.

Nos referimos, en primer lugar, al cargador Hexadecimal que denominamos Hexer, y un programa ensamblador que veremos posteriormente.

Potenciar el Hexer

Hexer apareció por vez primera en **AMSTRAD** Semanal número 3 y en este artículo vamos a completarlo para que podamos salvar y cargar código máquina en cinta/disco.

. Para ello, tenemos el programa número 1, que muestra las líneas que hay que añadir al Hexer original.

El ensamblador *(programa 3)* también se unirá al Hexer de la siguiente manera:

1. Cargue el Hexer original *(LOAD «HEXER»)*.

2. Introduzca el programa 1.

3. Renumere el programa completo a partir del número de línea 10000 *(RENUM 10000)*.

4. Sálvelo como ASCII *(SAVE «HEXER», A.*

5. Limpie la memoria *(NEW)*.

6. Teclee el ensamblador *(programa número 3)*.

7. Con el ensamblador en memoria, haga un merge con el Hexer desde cinta/disco *(MERGE «HEXER»)*.

8. Teclee el programa número 2, respetando los números de línea.

A estas alturas tenemos en la memoria del **Amstrad** el Hexer modificado, el ensamblador y el programa **«menú»** *(número 2)* juntos.

Queda solamente hacer una pequeña modificación, esto es, cambiar todas las referencias a &3000 ó 3000 por &8000 ó 8000, en la parte del programa correspondiente al Hexer, con idea de permitirles a ambos que coexistan en memoria. También sería conveniente cambiar &2FF8 ó 2FF8 por &7FF8 ó 7FF8.

Una vez hechas estas modificaciones, salve el conjunto en cinta/disco de forma normal y el programa completo puede utilizarse.

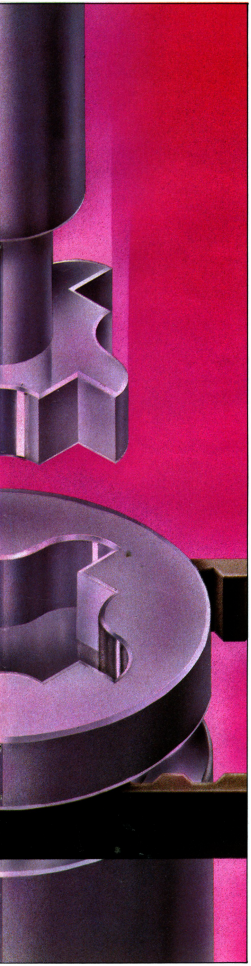
No más ensamblar a mano

Vamos a ocuparnos ahora del ensamblador.

Un programa en lenguaje máquina es una serie de números escritos en el sistema binario: bien. Las series de unos y ceros no tienen ningún sentido para las personas: no tan bien. Por eso se creó el lenguaje ensamblador, cuya función es, mediante el uso de mnemónicos, facilitar la programación.

Recordad: tiene más sentido LD H, &2A que &26, &2A, y aún tendría más sentido si pudiéramos decir:

.puntos=&2A
LD H, puntos



mediante un ensamblador, sí podemos porque estos programas se encargan de la ingrata tarea de traducir menmónicos, texto a números hexa y a la postre a binario, para que el ordenador pueda entenderlos.

Nuestro ensamblador está escrito íntegramente en Basic, por lo que ocupa la parte inferior de la memoria. Es conveniente, por tanto, que el código máquina ocupe la parte superior.

HIMEM puede variarse hacia abajo para hacerle sitio si es necesario.

Código fuente y código objeto

El código en lenguaje ensamblador, conocido como código fuente, puede salvarse en cinta/disco con o sin el propio programa ensamblador.

Una vez ensamblado, el código objeto que se crea puede ser llamado desde Basic mediante una instrucción CALL, seguido del número, en hexa o decimal, a partir del cual se ensambló el código fuente.

Todas las instrucciones y etiquetas de éste deben ir colocadas en sentencias DATA. Sólo se permite una instrucción por línea y pueden tener 1, 2 ó 3 partes.

El tipo de letra usado deben ser mayúsculas, con un espacio entre la primera y segunda parte *(si la hay)*, y una coma entre las partes 2 y 3 *(si la hay)*. No se permiten espacios extra.

Las etiquetas deben ir escritas en minúsculas.

Un ejemplo:

RET	instrucción con una sola parte;
DJNZ Loop	2 partes separadas por un espacio;
LD A, (IX+4)	3 partes: un espacio, una coma.

Para reservar espacio con HIMEM, se usa MEMORY de la forma habitual:

```
10 MEMORY &7FFF: REM el código objeto a partir de &8000.
```

El ensamblador cumple su función en dos pasadas y, mediante la variable printer, podemos dirigir su salida a la impresora. Si dicha variable es igual a 1, la impresora se activa; si es igual a 0, se desactiva:

```
20 printer=0: REM impresora desactivada
```

La primera instrucción de código fuente en las DATAS debe ser ORG, de ORiGen, seguida de un número:

```
30 DATA ORG &8000
```

en donde se colocará el código máquina ensamblado.

Al definir una etiqueta, debe ir precedida por un punto, como en:

```
30 DATA .puntos=5
```

y puede hacerse igual a una dirección de memoria determinada según el procedimiento del programa 4 *(línea 70)*.

También se puede asignar a una etiqueta un valor positivo o negativo mediante el signo igual *(líneas 20 y 30)*.

```
20 DATA     .print=&BB5A
30 DATA     .offset=5
60 REM
70 DATA     .loop
80 DATA     LD, (IX+offset)
90 DATA     CALL print
95 DATA     JR, loop
```

Programa 4

```
100 DATA    LD A, 27
110 DATA    LD B, &1B
120 DATA    LD C, %11011
130 DATA    .number=27
140 DATA    LD D, number
```

Programa 5

Los números pueden ser en decimal, hexadecimal o binario, como en el programa 5.

El programa 6 permite observar cómo se usan los directivos de otros ensambladores DEFB *(DEFine Byte), (DEFW, define palabra, del inglés DEFine Word), DEF$ (para colocar cadenas de caracteres)* y DEFS *(DEFine Storage, define almacenamiento de datos)*.

```
150 DATA    DEFB &FF
160 DATA    DEFW number
170 DATA    DEF$ «Assembler»
190 DATA    DEFS 150
200 DATA    .number=1000
```

Programa 6

Podemos incluir comentarios en el código fuente, en líneas que comiencen con apóstrofe *(tecla 7+Shift)*.

Si arrancamos el ensamblador, supuesto que todavía no se ha unido al Hexer, el programa se detendrá con el mensaje **«Origen?»,** indicado, bien que no hay código fuente en DATAS, bien que la primera de ellas no es un ORG.

Teclee AUTO e introduzca las líneas de ensamblador, procediéndolas con la palabra Basic DATA (use la tecla ENTER pequeña cuando sea necesario teclear Enter).

Creemos que sería una buena idea introducir todos los listados en ensamblador (con menmónicos) dados en anteriores artículos, comprobar que funcionan y familiarizarse con el ensamblador.

Salvar y grabar código fuente

El código fuente puede ser grabado con el propio ensamblador o éste ser borrado con DELETE y entonces salvar el resto.

Para recuperarlo, basta con Load el ensamblador y luego Merge con el código fuente grabado anteriormente.

Para salvar el código objeto generado, se puede hacer algo parecido a:

save **«código»,** b, &8000, &100

en donde los 2 últimos números indican la dirección de ensamblado *(la de ORG)* y la longitud en bytes.

Las subrutinas del programa están separadas para mayor claridad. Comienza en la 5000 para dejar sitio a las líneas DATA de código fuente, y el programa se escribió en Mode 2, por lo que tendrá más sentido hacerlo funcionar en ese modo.

Los tipos de instrucciones, los mnemónicos, se colocan en una matriz y se mira a ver si concuerda con la primera parte de la instrucción, textI$.

Si es así, el programa bifurca a la rutina adecuada mediante ON GOSUB.

Los nombres de las etiquetas se guardan en label *$(50)* y su valor en valúe *(50)*.

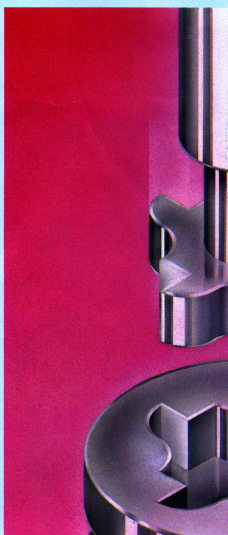
Algunas instrucciones son ensambladas por la misma rutina, debido a la similitud binaria de su código de operación, como las BIT, RES y SET.

Por lo demás, el programa está bastante bien estructurado, y pensamos que, una vez tecleado y comprobado, será de gran utilidad al programador. No queremos ensamblar a mano nunca más.

```
150 PRINT "5. Fin del programa"
155 PRINT "6. Salvar codigo"
156 PRINT "7. Cargar codigo"
160 b$ = INKEY$:IF b$="" GOTO 160
170 IF INSTR("1234567",b$)=0 GOTO 1
60
180 b = VAL(b$) : ON b GOSUB 210,40
0,550,350,200,700,800
700 INPUT "Direccion de comienzo";
start$
710 IF start$ = "" THEN start$ = "3
000"
720 start = VAL( "&" + start$ )
730 INPUT "Nombre";name$
740 IF name$ = "" THEN GOTO 730
750 INPUT "Longitud"; length$
760 IF length$ = "" THEN length$ =
"400"
770 length = VAL( "&" + length$ )
780 SAVE name$,b,start,length
790 RETURN
800 INPUT "Nombre";name$
810 LOAD name$
820 RETURN
```

```
4000 REM Programa 2
4010 MODE 1:INK 0,0:BORDER 0
4020 LOCATE 10,5:PEN 3:PAPER 1:PRIN
T"Ensamblador y Hexer"
4030 LOCATE 15,10:PAPER 0
4040 PEN 1:PRINT"1. ";:PEN 2:PRINT"
Hexer"
4050 LOCATE 15,12
4060 PEN 1:PRINT"2. ";:PEN 2:PRINT"
Ensamblador"
4070 LOCATE 15,16
4080 PEN 3:PRINT"Pulse 1 o 2"
4090 IF INKEY(64)>-1 THEN CLS:GOTO
10000
4100 IF INKEY(65)=-1 GOTO 4090
4110 PEN 1
```

```
5000 REM ======= ensamblador ======
==
5010 REM By R.A.Waddilove
5020 REM(c)Amstrad Semanal
5030 MODE 1:INK 0,0:BORDER 0
5040 GOSUB 5330:inicializacion
5050 FOR pass=1 TO 2
5060 IF pass=2 AND printer=1 THEN l
ist=0
5070 CLS:RESTORE:READ text1$
5080 IF LEFT$(text1$,3)<>"ORG" THEN
 PRINT "Origen ?":END
5090 x=VAL("&"+MID$(text1$,INSTR(text1$,"
")+1):GOSUB 7570:pc=x
5100 PRINT #list,"RAM Assembler V.
3":PRINT #list:PRINT #list,"Paso.
.";pass;TAB(20);text1$:PRINT #list
o
5110 WHILE text1$<>"END"
5120 PEN 1:READ text1$:textr$=""":IF
 LEFT$(text1$,1)="'" THEN PRINT #li
st,TAB(20)MID$(text1$,2):GOTO 5120
5130 PRINT #list,,HEX$(pc);":";:PEN
 3
5140 byte1=-1:byte2=-1:byte3=-1:byt
e4=-1:index=-1
5150 i%=-1
5160 FOR j%=0 TO 69
5170 IF INSTR(text1$,type$(j%))=1 T
HEN i%=j%
5180 NEXT
5190 IF i%=-1 THEN PRINT:PRINT text
1$;"???":END
5200 IF i%<36 THEN ON i% GOSUB 6860
,7010,7230,6460,7110,6160,6380,6500
,6420,6420,6420,6420,6380,6380,6760
,6380,5980,6380,5900,6380,6380,6010
,6060,6760,6420,6420,6420,6420,6260
,6210,5590,6420,6420,6420,6420
5210 IF i%>35 THEN i%=i%-35:ON i% G
OSUB 6420,6380,6510,6420,6420,6110,
6420,6420,5520,5520,7130,6340,6420,
6420,6650,6380,6380,6640,6420,6670,
6380,6660,6380,6420,5490,6990,6380,
7140,6600,6620,6630,6480,6490,5420
5220 IF index>-1 THEN x=index:GOSUB
 7350
5230 IF byte1>-1 THEN x=byte1:GOSUB
 7350
5240 IF byte2>-1 THEN x=byte2:GOSUB
 7350
5250 IF byte3>-1 THEN x=byte3:GOSUB
 7350
5260 IF byte4>-1 THEN x=byte4:GOSUB
 7350
5270 PEN 2:PRINT #list,TAB(20);tex
t1$;:IF textr$="" THEN PRINT #list
 ELSE PRINT #list,",";textr$
5280 WEND
5290 NEXT pass
5300 PEN.1
5310 END
5320 :
5330 REM === inicializa ===
5340 DIM type$(69),label$(50),value
(50),cond$(7)
5350 RESTORE 7810:FOR i%=0 TO 69:RE
AD type$(i%):NEXT
5360 RESTORE 7830:FOR i%=0 TO 7:REA
D cond$(i%):NEXT
```



```
5370 KEY 139,CHR$(13)+"DATA  "
5380 KEY 138,CHR$(13)+"REM   "
5390 num!=8
5400 RETURN
5410 :
5420 REM ==== def ====
5430 IF MID$(text$,4,1)="S" THEN x
$=MID$(text$,6):GOSUB 7570:pcmpcxx
:RETURN
5440 IF MID$(text$,4,1)="F" THEN F
OR j%=INSTR(text$,"/")+1 TO LEN(te
xt$)-1:POKE pc,ASC(MID$(text$,j%))
:pcmpc+1:NEXT:RETURN
5450 x$=MID$(text$,INSTR(text$,"
")+1):GOSUB 7570
5460 IF MID$(text$,4,1)="B" THEN b
yte!= ELSE GOSUB 7480:byte1=lbibyt
e2=hb
5470 RETURN
5480 :
5490 REM ==== rst ====
5500 x$=MID$(text$,5):x=VAL("&"+x$
)*8:byte1=VAL("&X11"+BIN$(x,3)+"111
"):RETURN
5510 :
5520 REM ==== push/pop ====
5530 x$=RIGHT$(text$,2)
5540 IF x$="IX" OR x$="IY" THEN ind
ex=-&DD*(x$="IX")-&FD*(x$="IY"):x$=
"HL"
5550 IF x$="AF" THEN byte1=-&F5*(IN
STR(text$,"U"))0)-&F1*(INSTR(text$
,"O"))0):RETURN
5560 GOSUB 7730:IF INSTR(text$,"PO
P") THEN byte1=VAL("&X11"+x$+"0001"
) ELSE byte1=VAL("&X11"+x$+"0101")
5570 RETURN
5580 :
5590 REM ==== ld ====
5600 READ text$
5610 IF INSTR(text$,"IX")+INSTR(te
xt$,"IX") THEN index=&DD
5620 IF INSTR(text$,"IY")+INSTR(te
xt$,"IY") THEN index=&FD
5630 IF INSTR(text$,"(")+INSTR(tex
t$,")")=0 GOTO 5810:'no brackets
5640 IF INSTR(text$,"(") GOTO 5720
:'left bracket
5650 IF text$="(HL)" THEN x$=RIGHT
$(text$,1):GOSUB 7690:byte1=VAL("&
X01"+x$+"110"):RETURN
5660 x$=text$:IF x$="(BC)" OR x$="
(DE)" THEN byte1=-&A*(x$="(BC)")-&l
A*(x$="(DE)"):RETURN
5670 IF INSTR(text$,"HL")+INSTR(te
xt$,"I") THEN byte1=&2A:x$=MID$(te
xt$,2,LEN(text$)-2):GOSUB 7570:GO
SUB 7480:byte2=lbibyte3=hb:RETURN
5680 IF INSTR(text$,"I") THEN"x$=R
IGHT$(text$,1):GOSUB 7690:byte1=VA
L("&X01"+x$+"110"):x$=MID$(text$,5
,LEN(text$)-5):GOSUB 7570:byte2=x:
RETURN
5690 IF LEN(text$)=4 THEN byte1=&3
A:x$=MID$(text$,2,LEN(text$)-2):G
OSUB 7570:GOSUB 7480:byte2=lbibyte3
=hb:RETURN
5700 byte1=&ED:x$=RIGHT$(text$,2):
GOSUB 7730:byte2=VAL("&X01"+x$+"101
1"):x$=MID$(text$,2,LEN(text$)-2)
:GOSUB 7570:GOSUB 7480:byte3=lbibyt
e4=hb:RETURN
5710 :
5720 IF LEFT$(text$,1)="I" OR text
r$="HL" THEN byte1=&22:x$=MID$(text
1$,5,LEN(text1$)-5):GOSUB 7570:GOSU
B 7480:byte2=lbibyte3=hb:RETURN
5730 IF INSTR(text$,"(I") THEN GOT
O 5790
5740 x$=MID$(text$,5,1):IF x$="B"
OR x$="D" THEN byte1=-&2*(x$="B")-&
12*(x$="D"):RETURN
5750 x$=text$:IF x$="BC" OR x$="DE
" OR x$="SP" THEN byte1=&ED:GOSUB 7
730:byte2=VAL("&X01"+x$+"001"):x$=
MID$(text$,5,LEN(text$)-5):GOSUB
7570:GOSUB 7480:byte3=lbibyte4=hb:R
ETURN
5760 IF INSTR(text$,"H")=0 THEN by
te1=&32:x$=MID$(text$,5,LEN(text$
)-5):GOSUB 7570:GOSUB 7480:byte2=lb
ibyte3=hb:RETURN
5770 x$=text$:IF x$>="A" AND x$<="
L" THEN GOSUB 7690:byte1=VAL("&X011
10"+x$):RETURN
5780 byte1=&36:GOSUB 7570:byte2=x:R
ETURN
5790 IF text$="A" AND text$<="L"
 THEN x$=text$:GOSUB 7690:byte1=VA
L("&X01110"+x$):x$=MID$(text$,8,LE
N(text$)-8):GOSUB 7570:byte2=x:RET
URN
5800 byte1=&36:x$=MID$(text$,8,LEN
(text$)-8):GOSUB 7570:byte2=x:x$=t
ext$:GOSUB 7570:byte3=x:RETURN
5810 IF RIGHT$(text$,1)="I" OR tex
t$="R" THEN byte1=&ED:byte2=-&57*(
text$="I")-&47*(text$="A"):RETURN
5820 IF RIGHT$(text$,1)="R" OR tex
t$="R" THEN byte1=&ED:byte2=-&5F*(
text$="R")-&4F*(text$="A"):RETURN
5830 IF INSTR(text$,"I") THEN byte
1=&21:x$=text$:GOSUB 7570:GOSUB 74
80:byte2=lbibyte3=hb:RETURN
5840 IF text$="HL" OR LEFT$(text$
,1)="I" THEN byte1=&F9:RETURN
5850 IF MID$(text$,LEN(text$)-1,1
))" " THEN x$=RIGHT$(text$,2):GOSU
B 7730:byte1=VAL("&X00"+x$+"000");
x$=text$:GOSUB 7570:GOSUB 7480:byt
e2=lbibyte3=hb:RETURN
5860 x$=RIGHT$(text$,1):GOSUB 7690
5870 IF text$="A" AND text$<="L"
 THEN byte1=VAL("&X01"+x$):x$=textr
$:GOSUB 7690:byte1=VAL("&X"+BIN$(by
te1,5)+x$):RETURN
5880 byte1=VAL("&X00"+x$+"110"):x$=
textr$:GOSUB 7570:byte2=x:RETURN
5890 :
5900 REM ==== ex ====
5910 READ textr$
5920 IF textr$="IX" THEN index=&DD
5930 IF textr$="IY" THEN index=&FD
5940 IF INSTR(text1$,"S") THEN byte
1=&E3:RETURN
5950 IF text1$="HL" THEN byte1=&EB
ELSE byte1=&8
5960 RETURN
5970 :
5980 REM ==== djnz ====
5990 byte1=&10:x$=MID$(text$,INSTR
(text$," ")+1):GOSUB 7570:GOSUB 75
20:byte2=x:RETURN
6000 :
6010 REM ==== im ====
6020 byte1=&ED:x=VAL(RIGHT$(text1$,
1))
6030 byte2=-&46*(x=0)-&56*(x=1)-&5E
*(x=2)
6040 RETURN
6050 :
6060 REM ==== in ====
6070 READ textr$
6080 IF textr$="(C)" THEN byte1=&ED
:x$=RIGHT$(text1$,1):GOSUB 7690:byt
e2=VAL("&X01"+x$+"000"):RETURN
6090 byte1=&DB:x$=MID$(textr$,2,INS
TR(textr$,")")-2):GOSUB 7570:byte2=
x:RETURN
6100 :
6110 REM ==== out ====
6120 READ textr$
6130 IF INSTR(text1$,"C") THEN byte
1=&ED:x$=textr$:GOSUB 7690:byte2=VA
L("&X01"+x$+"001"):RETURN
6140 byte1=&D3:x$=MID$(textr$,6,INS
TR(text1$,")")-6):GOSUB 7570:byte2=
x:RETURN
6150 :
6160 REM ==== call ====
6170 x$=RIGHT$(text1$,2):GOSUB 7410
6180 IF x=-1 THEN x$=MID$(text1$,4)
:GOSUB 7570:GOSUB 7480:byte1=&CD:by
te2=lbibyte3=hb:RETURN
6190 byte1=VAL("&X11"+BIN$(x,3)+"10
0"):READ textr$:x$=textr$:GOSUB 757
0:GOSUB 7480:h:ce2=lbibyte3=hb:RETU
RN
6200 :
6210 REM ==== jr ====
6220 x$=RIGHT$(text1$,2):GOSUB 7410
6230 IF x=-1 THEN x$=MID$(text1$,4)
:GOSUB 7570:GOSUB 7520:byte1=&18:by
te2=x:RETURN
6240 byte1=VAL("&X001"+BIN$(x,2)+"0
00"):READ textr$:x$=textr$:GOSUB 75
70:GOSUB 7520:byte2=x:RETURN
6250 :
6260 REM ==== jp ====
6270 IF INSTR(text1$,"X") THEN inde
x=&DD
6280 IF INSTR(text1$,"Y") THEN inde
x=&FD
6290 IF INSTR(text1$,"(") THEN byte
1=&E9:RETURN
6300 x$=RIGHT$(text1$,2):GOSUB 7410
6310 IF x=-1 THEN x$=MID$(text1$,4)
:GOSUB 7570:GOSUB 7480:byte1=&C3:by
te2=lbibyte3=hb:RETURN
6320 byte1=VAL("&X11"+BIN$(x,3)+"01
0"):READ textr$:x$=textr$:GOSUB 757
0:GOSUB 7480:byte2=lbibyte3=hb:RETU
RN
6330 :
6340 REM ==== ret ====
6350 IF text1$="RET" THEN byte1=&C9
 ELSE x$=RIGHT$(text1$,2):GOSUB 741
0:byte1=VAL("&X11"+BIN$(x,3)+"000")
6360 RETURN
6370 :
6380 REM ==== finales ====
6390 x$=text1$:byte1=-&IF*(x$="RRA"
)-&FF*(x$="RRCA")-&37*(x$="SCF")-&3F
*(x$="CCF")-&2F*(x$="CPL")-&27*(x$=
```

```
"DAA")~&FD#(x$="DI")~&FB#(x$="EI")~
&D9#(x$="EXX")~&76#(x$="HALT")~&17#
(x$="RLA")~&7#(x$="RLCA")
6400 RETURN
6410 :
6420 x$=text1$:x=~&A9#(x$="CPD")~&
7#(x$="RRD")~&B9#(x$="CPDR")~&A1#(x
$="CPI")~&B1#(x$="CPIR")~&AA#(x$="I
ND")~&BA#(x$="INDR")~&A2#(x$="INI")
~&B2#(x$="INIR")~&A8#(x$="LDD")~&BE
#(x$="LDDR")~&A0#(x$="LDI")~&B0#(x$
="LDIR")
6430 byte2=x~&44#(x$="NEG")~&B8#(x$
="OTDR")~&B3#(x$="OTIR")~&AB#(x$="O
UTD")~&A3#(x$="OUTI")~&4D#(x$="RETI
")~&45#(x$="RETN")~&6F#(x$="RLD")
6440 byte1=&ED:RETURN
6450 :
6460 REM === and/sub/xor ===
6470 byte=&A6:GOTO 6530:'and
6480 byte=&96:GOTO 6530:'sub
6490 byte=&AE:GOTO 6530:'xor
6500 byte=&BE:text1$="CP "+MID$(tex
t1$,3):GOTO 6530:'cp
6510 byte=&B6:text1$="OR "+MID$(tex
t1$,3):GOTO 6530:'or
6520 :
6530 IF INSTR(text1$,"I") GOTO 6570
6540 IF INSTR(text1$,"HL") THEN byt
e1=byte:RETURN
6550 x=RIGHT$(text1$,1):IF (x$)="A
" AND x$<>"L") AND INSTR(text1$,"&"
)=0 THEN GOSUB 7690:byte1=(byte AND
&X11111000)+VAL("&X"+x$):RETURN
6560 byte1=byte OR &X1000000:x$=MID
$(text1$,5):GOSUB 7570:byte2=x:RETU
RN
6570 IF INSTR(text1$,"IX") THEN ind
ex=&DD ELSE index=&FD
6580 byte1=byte:x$=MID$(text1$,9,IN
STR(text1$,")")-9):GOSUB 7570:byte2
=x:RETURN
6590 :
6600 REM === sla/sra/srl/rlc/rl/rrc
/rr ===
6610 byte=&26:GOTO 6690:'sla
6620 byte=&2E:GOTO 6690:'sra
6630 byte=&3E:GOTO 6690:'srl
6640 byte=&6:GOTO 6690:'rlc
6650 byte=&16:text1$="RL "+MID$(tex
t1$,3):GOTO 6690:'rl
6660 byte=&E:GOTO 6690:'rrc
6670 byte=&1E:text1$="RR "+MID$(tex
t1$,3):GOTO 6690:'rr
6680 :
6690 IF INSTR(text1$,"I") GOTO 6730
6700 byte1=&CB
6710 IF INSTR(text1$,"HL") THEN byt
e2=byte:RETURN
6720 x=RIGHT$(text1$,1):IF x$>"A"
 AND x$<"L" THEN GOSUB 7690:byte2=
(byte AND &X11111000)+VAL("&X"+x$):
RETURN
6730 IF INSTR(text1$,"IX") THEN ind
ex=&DD ELSE index=&FD
6740 byte1=&CB:x=MID$(text1$,9,INS
TR(text1$,")")-9):GOSUB 7570:byte2=
x:byte3=byte:RETURN
6750 :
6760 REM === inc/dec ===
6770 IF INSTR(3,text1$,"H") THEN GO
TO 6820
6780 byte=-1*(text1$="I")
6790 IF INSTR(text1$,"(H") THEN byt
e1=&34 OR byte:RETURN
6800 IF RIGHT$(text1$,2)="1" THEN x
$=RIGHT$(text1$,1):GOSUB 7690:byte1
=byte OR VAL("&X00"+x$+"100"):RETUR
N
6810 byte=-8*(text1$="I"):x$=RIGHT$
(text1$,2):GOSUB 7730:byte1=byte OR
 VAL("&X00"+x$+"011"):RETURN
6820 IF INSTR(text1$,"IX") THEN ind
ex=&DD ELSE index=&FD
6830 IF INSTR(text1$,"(") THEN byte
=-1*(text1$="I"):byte1=&34 OR byte:
x$=MID$(text1$,9,INSTR(text1$,")")-
9):GOSUB 7570:byte2=x:RETURN
6840 byte=-8*(text1$="I"):byte1=&23
 OR byte:RETURN
6850 :
6860 REM === etiquetas ===
6870 PEN 1:kX=-1:x$=MID$(text1$,2)
6880 IF INSTR(x$,"=")>0 THEN x$=LEF
T$(x$,INSTR(x$,"=")-1)
6890 FOR j%=0 TO num1
6900 IF x$=label$(j%) THEN k%=j%
6910 NEXT
6920 IF k%=-1 THEN k%=num1:num1=num
1+1
6930 IF num1>50 THEN PRINT:PRINT "T
oo many labels":STOP
6940 label$(k%)=x$
6950 IF INSTR(text1$,"=")=0 THEN va
lue(k%)=pc:RETURN
6960 x$=MID$(text1$,INSTR(text1$,"=
")+1):GOSUB 7570:value(k%)=x
6970 RETURN
6980 :
6990 REM === sbc/adc ===
7000 byte=&H10011110:GOTO 7020:'sbc
7010 byte=&X10001110:GOTO 7020:'adc
7020 READ textr$
7030 IF INSTR(textr$,"HL") THEN x$=
textr$:GOSUB 7730:byte1=&ED:byte2=V
AL("&X01"+x$+"0010")+(8 AND text1$(
"S"):RETURN
7040 IF textr$>="A" AND textr$<="L"
 THEN byte=byte AND &X11111000:x$=t
extr$:GOSUB 7690:byte1=byte+VAL("&X
"+x$):RETURN
7050 IF textr$="(HL)" THEN byte1=by
te:RETURN
7060 IF INSTR(textr$,"I") GOTO 7080
7070 byte1=byte OR &X10000000:x$=tex
tr$:GOSUB 7570:byte2=x:RETURN
7080 IF INSTR(textr$,"IX") THEN ind
ex=&DD ELSE index=&FD
7090 byte1=byte:x$=MID$(textr$,5,IN
STR(textr$,")")-5):GOSUB 7570:byte2
=x:RETURN
7100 :
7110 REM === bit/set/res ===
7120 byte=&X01000000:GOTO 7150:'bit
7130 byte=&X10000000:GOTO 7150:'res
7140 byte=&X11000000:GOTO 7150:'set
7150 byte1=byte:x=VAL(RIGHT$(text1
$,1)):byte2=byte+VAL("&X"+BIN$(x,3)
+"000")
7160 IF INSTR(text$,"I") GOTO 7200
:'ix/iy
7170 byte1=&CB
7180 IF textr$="(HL)" THEN byte2=by
te2+VAL("&X"+x$):RETURN
7190 x$=textr$:GOSUB 7690:byte2=byt
e2+VAL("&X"+x$):RETURN
7200 IF INSTR(textr$,"IX") THEN ind
ex=&DD ELSE index=&FD
7210 byte3=byte2+VAL("&X110"):byte1
=&CB:x$=MID$(textr$,5,INSTR(textr$,
")")-5):GOSUB 7570:byte2=x:RETURN
7220 :
7230 REM === add ===
7240 READ textr$
7250 IF INSTR(textr$,"Y") THEN inde
x=&FD
7260 IF INSTR(textr$,"X") THEN inde
x=&DD
7270 IF INSTR(textr$,"L") OR INSTR(
text$,"I") THEN x$=textr$:GOSUB 77
30:byte1=VAL("&X00"+x$+"1001"):RETU
RN
7280 IF textr$>="A" AND textr$<="L"
 THEN x$=textr$:GOSUB 7690:byte1=VA
L("&X10000"+x$):RETURN
7290 IF textr$="(HL)" THEN byte1=&8
6:RETURN
7300 IF INSTR(textr$,"I") GOTO 7320
:'adc a,(ix/iy+n)
7310 byte1=&CB:x$=textr$:GOSUB 7570
:byte2=x:RETURN
7320 IF INSTR(textr$,"Y") THEN inde
x=&FD ELSE index=&DD
7330 byte1=&86:x$=MID$(textr$,5,INS
TR(textr$,")")-5):GOSUB 7570:byte2=
x:RETURN
7340 :
7350 REM === print hex$(x) ===
7360 hex$(x)=hex$(pc+x)
7370 IF x<16 THEN PRINT #list0,"0";
7380 PRINT #list0,HEX$(x);" ";
7390 RETURN
7400 :
7410 REM === get code for condition
 x$ -> x ===
7420 x=-1
7430 FOR j%=0 TO 7
7440 IF x$=cond$(j%) THEN x=j%
7450 NEXT
7460 RETURN
7470 :
7480 REM === x -> bytebajo,bytealto
 ===
7490 lb=x-256*INT(x/256):hb=INT(x/2
56)
7500 RETURN
7510 :
7520 REM === calcula salto ===
7530 IF x>255 THEN x=x-pc-2:IF x>25
5 THEN PRINT text1$;" Too far":END
7540 IF x<0 THEN x=x+256
7550 RETURN
7560 :
7570 REM === x$,number/label -> x,n
umber ===
7580 WHILE x$("!"):x$=MID$(x$,2):WEN
D
7590 IF LEFT$(x$,1)="&" THEN x=VAL(
x$)-65536*(VAL(x$)<0):RETURN
7600 IF LEFT$(x$,1)="%" AND LEFT$(
x$,1)<>"9" THEN x=VAL(x$):RETURN
7610 IF LEFT$(x$,1)="%" THEN x=VAL(
"&X"+MID$(x$,2)):RETURN
7620 IF pass=2 THEN x=-1 ELSE x=0:R
ETURN
7630 FOR j%=0 TO num1
7640 IF label$(j%)=x$ THEN x=value(
j%)
7650 NEXT
7660 IF x=0 THEN PRINT text1$;" Lab
el???":END
7670 RETURN
7680 :
7690 REM === x$,register -> x$,bina
ry code ===
7700 x$=BIN$(ASC(x$)-66-8*(x$="A")+
2*(x$="H")+5*(x$="L"),3)
7710 RETURN
7720 :
7730 REM === x$,register pair -> x$
,binary code ===
7740 IF x$="BC" THEN x$="00"
7750 IF x$="DE" THEN x$="01"
7760 IF x$="HL" OR x$="IX" OR x$="I
Y" THEN x$="10"
7770 IF x$="SP" THEN x$="11"
7780 RETURN
7790 :
7800 REM === mnemonics ===
7810 DATA END,,,,ADC,ADD,AND,BIT,CAL
L,CCF,CP,CPD,CPDR,CPI,CPIR,CPL,DAA,
DEC,DI,DJNZ,EI,EX,EXX,HALT,IM,IN,IN
C,IND,INDR,INI,INIR,JP,JR,LD,LDD,LD
DR,LDI,LDIR,NEG,NOP,OR,OTDR,OTIR
7820 DATA OUT,OUTD,OUTI,POP,PUSH,RE
S,RET,RETI,RETN,RL,RLA,RLCA,RLC,RLD
,RR,RRA,RRC,RRCA,RRD,RST,SBC,SCF,SE
T,SLA,SRA,SRL,SUB,XOR,DEF
7830 DATA "NZ","Z","NC","C","PO",
"PE","P","M"
```

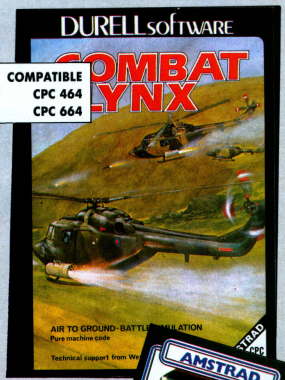
Suscríbete... y uno de estos tres sensacionales juegos será tuyo... ¡GRATIS!

M.H. AMSTRAD *te da a elegir entre tres de los mejores juegos existentes en el mercado para AMSTRAD;* ***COMBAT LYNX, DALEY THOMPSON'S DECATHLON y BEACH HEAD,*** *cualquiera de los cuales puede ser tuyo solamente con suscribirte a nuestra revista.* ***Aprovecha esta ocasión excepcional*** *y ahorra 2.100 pesetas (precio de venta del programa) más el importante descuento que se produce en el precio de cada número, por el hecho de ser suscriptor. Disfruta de las ventajas que supone recibir cómodamente tu revista a domicilio y de la seguridad de tener tu ejemplar aunque se haya agotado en los quioscos.*

Envíanos tu boletín de suscripción y no le des más vueltas, el número de juegos para regalos de suscripción, aunque grande, es limitado, y estos se podrían agotar mientras lo estás pensando.

Utiliza el cupón adjunto a la revista o suscríbete por teléfono (91) 733 50 12 (91) 733 50 16

OFERTA VALIDA SOLAMENTE PARA ESPAÑA



BEACH HEAD producido por U.S. GOLD es una misión de desembarco en una costa fuertemente defendida por las fuerzas aeronavales enemigas. Debes conducir tu flota hacia la bahía y repeler el ataque aéreo, si lo consigues tu siguiente obstáculo será una flotilla de destructores y acorazados, superada la cual desembarcarás tus anfibios en las arenas de la bahía, estos deben superar las defensas costeras y llegar a la fortaleza que es el objetivo final.

COMBAT LYNX simula una misión de defensa de unas bases atacadas por una división acorazada. Disponemos para enfrentarnos a ellos de un modernísimo helicóptero.

Este juego podría incluirse dentro del catálogo de los de estrategia, y su complejidad le dota de una gran dosis de adicción y belleza.

DALEY THOMPSON'S DECATHLON con este juego OCEAN enciende la llama olímpica y te reta a superar los récords de los campeonatos de todos los tiempos, el decathlon se desarrolla en dos días de competición y se compone de las siguientes pruebas:

PRIMER DIA: 100 m lisos, salto de longitud, lanzamiento de peso, salto de altura y 400 m lisos.

SEGUNDO DIA: 110 m vallas, lanzamiento de disco, salto con pértiga, lanzamiento de jabalina y los 1.500 m.

GENERADOR DE PALABRAS CLAVE

Análisis

Una de las primeras cosas que debemos intentar al crear nuestra primera obra maestra, es protegerla de intrusos por medio de una palabra clave.
Esta palabra, debe ser fácilmente reconocible por nosotros mismos e incomprensible para los demás.
El generador de palabras clave, es un medio idóneo para encontrar estas series de letras que para los demás no tienen sentido alguno y que nosotros somos capaces de recordar fácilmente; ayudándonos a introducirnos de paso en el mundo del tratamiento de cadenas alfanuméricas.

10,20 Título del programa.

30-180 Contiene el ciclo principal WHILE...WEND, que controla el programa.

La variable ESTADO, actúa como controlador del ciclo; éste continúa rotando mientras aquélla es falsa (valor 0), produciendo cada vez una clave distinta.

40 Inicializa las dos variables alfanuméricas, dándoles a cada una el valor nulo o cadena vacía, cada vez que se repite el ciclo principal. Sería interesante suprimirla y observar los resultados.

50 LETRAS$, contiene todas las letras del abecedario y es el lugar donde el programa elige las letras que componen las retorcida palabra clave.

En nuestro caso hemos puesto el abecedario, pero mentes más pueden colocar cualquier combinación de los caracteres que contiene el teclado.

60-80 La primera trampa, con ésta obligamos al usurario a introducir el número de letras que va a contener la palabra clave.

El WHILE....WEND que lo controla, solamente admite respuestas que contengan de uno a diez caracteres.

90-120 El ciclo FOR....NEXT, se repite una vez por cada una de las letras que componen la palabra clave.

100 Constituye la línea que selecciona aleatoriamente, una letra de la variable LETRAS$, almacenándola en CARACTER$.

110 Cada vez que se repite el ciclo, la última letra contenida en CARACTER$ es añadida al final de la variable CLAVE$, que al concluir el FOR....NEXT, contendrá la palabra clave completa.

130 Muestra en pantalla, la nueva clave formada.

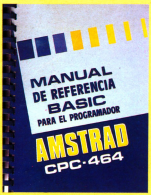
140-160 La segunda trampa, pregunta si el usuario acepta la palabra clave y guarda el resultado en RESPUESTA$.

Las condiciones del ciclo WHILE....WEND, se ocupan de asegurar que solamente las respuestas correctas sean aceptadas.

170 Si la respuesta es sí (S o s), la variable ESTADO toma el valor —1, finalizando el ciclo principal. En caso de que la respuesta sea negativa (N o n), ESTADO sigue teniendo el valor 0, lo que produce que el ciclo se repita de nuevo.

190 Cuando el programa sale del ciclo principal, al tener ESTADO el valor —1, la palabra clave elegida es mostrada en pantalla.

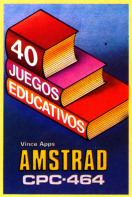
LIBROS EN CASTELLANO PARA TU AMSTRAD



Manual de Referencia Basic para el Programador
La más autorizada y completa guía para programar en Locomotive Basic.
3.400.— Pts.

Juegos Sensacionales para AMSTRAD
Listados completos de 27 estupendos juegos de muy diversos estilos.
1.950.— Pts.

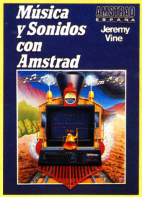
Programando con AMSTRAD
Fundamental para el usuario principiante.
Ameno y repleto de ejemplos.
2.400.— Pts.



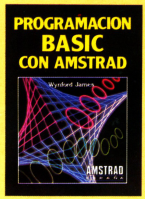
40 Juegos Educativos
Listados completos (matemáticas, geografía, música, etc.) para aprender divirtiéndose.
1.950.— Pts.

Código máquina para principiantes con AMSTRAD
Ideal para iniciarse en el código máquina del Z80 y en el sistema operativo del AMSTRAD.
2.100.— Pts.

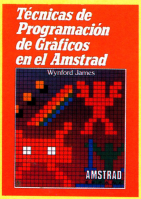
Hacia la Inteligencia Artificial con AMSTRAD
Convierta su AMSTRAD en un compañero inteligente
1.500.— Pts.



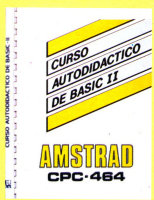
Música y Sonidos con AMSTRAD
Programe música y efectos sonoros y convierta su AMSTRAD en un sintetizador.
1.200.— Pts.



Programación de Basic con AMSTRAD.
Imprescindible para el principiante y eficaz herramienta para el programador avanzado.
2.100.— Pts.



Técnicas de Programación de Gráficos en el AMSTRAD
Este libro enseña a aprovechar las excelentes funciones gráficas del AMSTRAD, con múltiples ejemplos.
1.950.— Pts.



Curso Autodidáctico de Basic I y II
Un completo y estructurado Curso de Basic apoyado con numerosos ejemplos y acompañado de cassettes.
2.900.— Pts. cada volumen

indescomp
PUBLICACIONES

Avd. del Mediterráneo, 9
Telfs.: 433 45 48 — 433 48 76
28007 MADRID

Delegación en Cataluña:
C/. Tarragona, 110 — Telf. 325 10 58
08015 BARCELONA

DE VENTA EN EL CORTE INGLES
Y TIENDAS ESPECIALIZADAS

AMSTRAD CPC-6128

- MICROPROCESADOR Z80A.
- 128 K DE MEMORIA RAM.
- 48 K DE MEMORIA ROM QUE INCLUYEN EL LOCOMOTIVE BASIC Y EL SISTEMA OPERATIVO.
- 76 TECLAS, TECLADO NUMERICO Y DE CURSOR INDEPENDIENTE.
- TEXTO EN MONITOR DE 20, 40 U 80 COLUMNAS Y GRAFICOS CON DEFINICION DE HASTA 640 × 200 PUNTOS. 27 COLORES DISPONIBLES.
- HASTA 8 VENTANAS EN PANTALLA.
- GENERACION DE SONIDOS EN 3 VOCES Y 8 OCTAVAS.
- UNIDAD DE DISCO DE 3" (169 K BYTES).
- SISTEMAS OPERATIVOS AMS-DOS Y CPM/PLUS.
- CONECTORES PARA IMPRESORA, JOYSTICKS, CASSETTE, SEGUNDA UNIDAD DE DISCO, ETC.

SISTEMA COMPLETO CON MONITOR EN FOSFORO VERDE, MANUAL EN CASTELLANO, GARANTIA OFICIAL AMSTRAD ESPAÑA, DISCO CON SISTEMA OPERATIVO CP/M 2.2 Y LENGUAJE DR. LOGO, DISCO CON SISTEMA OPERATIVO CP/M PLUS (CP/M 3.0) Y UTILIDADES, DISCO CON SIETE PROGRAMAS DE OBSEQUIO

109.500 Pts.

SISTEMA COMPLETO IGUAL AL ANTERIOR PERO CON MONITOR EN COLOR.

134.500 Pts.

AMSTRAD (tm)
ESPAÑA

Avd. de Mediterráneo, 9, 28007 MADRID.
Tels. 433 45 48 - 433 48 76
Delegación Cataluña: C/. Tarragona, 110, 08015 BARCELONA - Tel. 325 10 58