AÑO I N.º 9

MICROHOBBY AMSTRAD Semanal

REVISTA INDEPENDIENTE PARA USUARIOS DE ORDENADORES AMSTRAD

AÑO I N.º 9

150 Ptas.

Canarias 160 ptas.

Base de datos RPA Systems, organiza la información para un acceso fácil y rápido

Convierte tu Amstrad en una máquina tragaperras con FRUTAS.

GENERADOR DE QUINIELAS ALEATORIAS: HASTA 8 APUESTAS

Código máquina

DE LA TEORIA A LA PRACTICA

SOFTWARE

Herber, el travieso hijo de Wally, perdido en los grandes almacenes

COMPUTIQUE

Te da más

AMSTRAD ESPAÑA

GARANTIA

Al comprar tu Amstrad te regalamos

- **Estuche con ocho programas originales**
  - Fruit Machine
  - Procesador texto
  - Almirante Graf
  - Oh Mummy
  - Plaga Galáctica
  - Amsdraw
  - Laberinto Sultan
  - Animal, Vegetal, Mineral
- **Un estupendo libro de Basic**
  - Guía de referencia del programador
  - y además te obsequiamos con un curso de introducción al Basic.

Venta a plazos hasta 36 meses.

Servimos tiendas
Tel: 227 91 99

Nuevo Amstrad CPC6128

COMPUTIQUE

Embajadores, 90 28012 Madrid Tfno. 2270980

Director Editorial
José I. Gómez-Centurión
Director Ejecutivo
Victor 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
J. Igual
Ilustradores
Javier 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

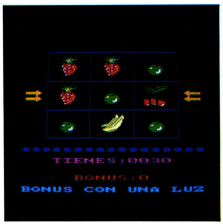
Dto. Circulación
Carlos Peropadre

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 9 • 29 de Octubre al 4 de Noviembre de 1985 150 ptas. (sobretasa Canarias, 10 ptas.)



MINI OFFICE

PROGRAMA DEL AÑO EN INGLATERRA

FINALIST MICRO COMPUTING BRITISH AWARDS 1985

1 PROCESADOR DE TEXTOS

¡Ideal para escribir cartas e informes! Características: Visualización continua del tiempo • Contador de palabras (indicando las palabras por minuto) • Texto normal o doble, en pantalla o impresora.

2 HOJA DE CALCULO

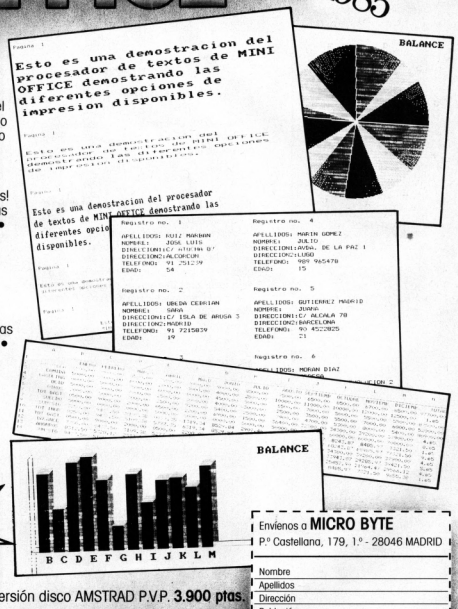
¡Utiliza tu micro para controlar tus cuentas! Características: Cifras visualizadas en filas y columnas • Actualización permanente • Actualización reflejada instantáneamente en toda la hoja • Grabación de los resultados para futuras modificaciones.

3 GRAFICOS

¡Convierte esos números en maravillosos gráficos! Características: Gráficos de barras en tres dimensiones • Gráficos de pastel • Histogramas.

4 BASE DE DATOS

¡Igual que los archivos de la oficina! Características: Cargar ficheros con solo pulsar una tecla • Clasificación • Modificación • Listados • Búsqueda.



*En Castellano
*Servimos en 48 Horas

4 PROGRAMAS P.V.P. 3.200 Ptas AL PRECIO DE 1

*Versión disco AMSTRAD P.V.P. **3.900 ptas.**

Envíenos a **MICRO BYTE**
P.º Castellana, 179, 1.º - 28046 MADRID

Nombre
Apellidos
Dirección
Población
D.P. Teléfono

Deseo que me envíen ☐ ejemplar/es del programa **MINI OFFICE**

PARA EL MICROORDENADOR SEÑALADO

☐ AMSTRAD ☐ COMMODORE ☐ SPECTRUM
☐ AMSTRAD VERSION DISCO
Sin gastos de envío

INCLUYO TALON NOMINATIVO ☐
CONTRA-REEMBOLSO ☐

Pedidos por teléfono
91 - 442 54 33/44

Primera plana

INTELIGENCIA ARTIFICIAL EN ESPAÑA

En Blanes, el Consejo Superior de Investigaciones Científicas, inauguró el 14 de octubre el primer Centro de Investigación y Desarrollo de Inteligencia Artificial en España. La nueva instalación, presumiblemente, se dedicará a la creación de sistemas expertos y de programas que puedan simular y reproducir, en la medida de lo posible, la facultad del razonamiento humano.

Desde hace ya algún tiempo, el éxito conseguido en el desarrollo de sistemas expertos en áreas muy específicas del conocimiento, ha impulsado enormemente esta faceta de la investigación informática que camina a pasos agigantados hacia convertirse en la industria más importante del siglo XX, tanto por el desafío tecnológico que plantea, como por las implicaciones filosóficas y sociales que supone la aparición y operatividad de sistemas **«pensantes»** no humanos.

Esperemos que nuestro país recorra pronto la distancia que nos separa de otras naciones desarrolladas en el campo de la Inteligencia Artificial.

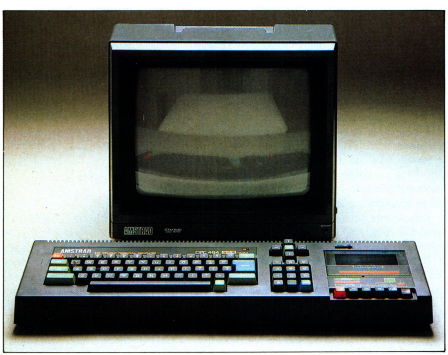
TURGEON LLEGA A ESPAÑA

Una importante empresa canadiense especializada en software y libros de informática, proyecta instalarse en España y otros países europeos.

Para comenzar, se ha formado en Francia Turgeon Europe, junto con la editorial Masson, para empezar ya mismo a comercializar sus productos en Europa.

CAD/CAM, A CANARIAS

Canarias solicita uno de los ocho centros Cad/Cam previstos en el Plan Informático Nacional. Al parecer, el tema se trató entre el vicepresidente del Gobierno canario y el ministro de Industria. Los sistemas Cad/Cam son equipos que utilizan un software sofisticado y un hardware en consonancia para realizar la función de diseño asistido por ordenador, en inglés Computer Assistance Design.



TRANSPUTER

Este es el nombre del nuevo procesador que viene a revolucionar el mundo de la informática.

Desarrollado por la casa Inmos, su producción masiva comenzará en la segunda mitad del año 86.

La principal cualidad de este chip, es que desarrolla todas las labores propias de un ordenador, con la ventaja de que los cálculos son realizados más rápidamente.

El Transputer ha sido diseñado de manera que incluye en un solo chip: proceso de datos, memoria y comunicaciones.

La capacidad de acceso del procesador por sí mismo es de 2 a 4 Kbytes de memoria RAM dinámica, la memoria puede ser ampliada mediante un chip especial, que mediante un interface une directamente la RAM estática con la dinámica, permitiendo una velocidad de transmisión de datos de hasta 26 Mbytes por segundo.

La gran ventaja de este ingenio, es que permite la conexión en paralelo sin la pérdida de tiempo de proceso que representa el organizar la comunicación entre las partes componentes de un sistema.

COMUNICACIONES

Una de las cosas más útiles que pueden hacerse con un ordenador es usarlo como terminal inteligente enlazado a un **«mainframe»**, o super computador.

Para conseguirlo, ambas máquinas deben ser capaces de hablarse entre sí siguiendo un cierto protocolo de comunicaciones.

Uno de estos protocolos, el OSI, para Europa, ve engrosadas las filas de sus partidarios con la incorporación de la empresa Honeywell. Un paso importante hacia la conversión en standard de un sistema al que ya pertenecen IBM y Digital.

BUCLES FOR NEXT

La gran ventaja que ofrecen los ordenadores, es que una vez que se ha programado una determinada tarea, estos la pueden repetir un gran número de veces sin que intervenga la mano del hombre.

Conseguir que se produzcan este número de repeticiones, solamente implica dos líneas adicionales de programa.

Lo que da una clara idea de lo útiles y prácticos que resultan los bucles for... next.

Hemos leído todos los artículos anteriores intentando averiguar el camino que ya hemos recorrido. Así que, desde ahora damos por supuesto que conoce todo sobre CLS, NEW, LIST y PRINT y que si le hablamos de GOTO no se ofenderá.

IF comprende todo esto THEN no tendrá ningún problema para seguir el resto de la serie.

Lo pondremos a prueba en el micro. Y no se quede parado con los programas que le proporcionemos. Pruebe a cambiarlos o a hacer los suyos propios. Y no dé por sentado lo que decimos, pruébelo en la práctica. Si quiere saber **«¿Qué ocurre si...?»** pruébelo en el **Amstrad.**

Y ahora vamos a empezar. Teclee el Programa 1 y veamos qué hace.

No es precisamente la parte de la programación más fascinante del mundo pero nos hace tratar algunos puntos importantes. Con lo que sabe ya no debería tener ninguna dificultad en comprender cómo trabaja.

Contadores

La línea 20 define la variable **«contador»** dándole el valor 1 mientras que la línea 30 lo escribe. La línea 40 aumenta en 1 **«contador»** almacenando el resultado en la variable **«contador».** De esta manera **«contador»** es ahora más grande que cuando salió en pantalla.

Ahora viene la parte ingeniosa. El IF de la línea siguiente examina si **«contador»** es menor o igual a 25. Si es así, el GOTO después de THEN envía al programa de vuelta a la línea 30 y todo el proceso vuelve a empezar.

Cuando **«contador»** es mayor que 25 el bucle va al final del programa. Técnicamente se dice que el programa sale del bucle. Entonces el proceso va a la línea siguiente. En este caso no existe ninguna, por lo tanto el proceso termina.

El programa es lo que conocemos como bucle condicional, la condición es que **«contador»** debe ser menor o igual que 25 para repetir el ciclo. El resultado es que los números 1 a 25 salen en pantalla y entonces termina el programa. ¿Podemos adivinar el valor de **«contador»** cuando acaba el programa? Probemos añadiendo:

```
60 PRINT «Al final el valor de contador es:»
   contador
```

y encontraremos que el resultado es 26.

Podríamos pensar que fuera 25 pero una oportuna mirada a la condición nos desengañará.

Como la condición es menor o igual a 25, cuando **«contador»** es 25 el programa cumplirá todavía el GOTO. Hará un salto atrás a la línea 30 e imprimirá **«contador».** La siguiente línea añadirá uno y el nuevo **«contador»** tendrá un valor de 26 cuando llegue a la pregunta de la línea 50. Como 26 es mayor que 25 el test falla, la condición es falsa y el GOTO se ignora. El programa sale del bucle y, no encontrando más líneas para procesar, termina.

Probemos cambiar la línea 50 por:

```
50 IF contador
25 THEN GOTO 30
```

o

```
50 IF contador=25 THEN GOTO 30
```

e intentaremos comprender qué es lo que está ocurriendo. Como ve, introduciendo pequeños cambios en las condiciones de los bucles se pueden producir grandes efectos en el modo de trabajar de los programas.

Encontrará un montón de problemas cuando utilice IFs y GOTOs para crear bucles engañosos con el uso de condiciones incorrectas. Tiene que conseguir interrumpirlos en el momento oportuno, lo que no siempre es fácil en programas más complicados.

Actualmente habiendo leído bastante no tendrá dificultad para entender por qué el Programa II dice HOLA 25 veces.

Utilice sus conocimientos para realizar cinco HOLAS, luego cinco mil. O haga trescientos cinco. ¿La condición deberá ser menor que 306? o ¿menor o igual que 305? o ¿harán ambas el trabajo? o ¿no lo hará ninguna? Pruébelo y veremos. Cuando acabe de juguetear con ello, corra el Programa III.

Ciclo FOR NEXT

La salida producida por este programa es exactamente la misma que la del Programa I, y sin embargo es una línea más corta. ¿Cómo es posible?

La respuesta se encuentra en el FOR de la línea 20 y el NEXT de la línea 40. A esta forma se la conoce como un bucle FOR... NEXT.

Todas las líneas del programa que vienen entre el FOR y el NEXT que le sigue forman parte del bucle. En realidad se conocen como cuerpo del bucle FOR... NEXT. Y, como suponemos por los bucles de los que ya nos hemos ocupado, se repite el ciclo una y otra vez.

En cada vuelta se ejecutan las líneas que forman parte del cuerpo del bucle. En este caso sólo había una línea intercalada entre el FOR y el NEXT. Es la línea 30 la que imprime el valor de **«contador».**

Obviamente el valor de **«contador»** varía según progresa el bucle, ¿pero cómo? Anteriormente ya tuvimos algo parecido a

```
contador=contador+1
```

pero no hay nada parecido a esto en el Programa III. Y ¿cómo sabe ahora el bucle que ha terminado? Y mientras estamos en ello, **¿cuál es el valor inicial de «contador»?**

Las respuestas se encuentran en la línea 20 donde FOR está seguido de:

```
contador=1 TO 25
```

Esta es la línea que le dice al **Amstrad** el valor inicial de **«contador»** que, en este caso, es uno. También automáticamente suma uno al valor cada vez que da una vuelta al bucle y además determina cuántas veces hará el ciclo. Vamos a verlo más atentamente.

Variable de control

Como sabemos, FOR y NEXT delimitan el cuerpo del bucle, las líneas que se han de repetir. Observemos que FOR va seguido de una variable, en este caso **«contador».** Es lo que pomposamente conocemos como variable de control del bucle y es la que decide cuántas veces el bucle hará el ciclo.

La variable de control comienza en el valor que le ha sido asignado después de FOR y cada vez que da una vuelta al bucle el valor es

PROGRAMAS

```
10 REM Programa I
20 contador=1
30 PRINT contador
40 LET contador=contador+1
50 IF contador<=25 THEN GOTO 30
```

```
10 REM Programa II
20 contador=1
30 PRINT "HOLA"
40 LET contador=contador+1
50 IF contador<=25 THEN GOTO 30
```

```
10 REM Programa III
20 FOR contador=1 TO 25
30 PRINT contador
40 NEXT contador
```

```
10 REM Programa IV
20 FOR contador=1 TO 25
30 PRINT "HOLA"
40 NEXT contador
```

```
10 REM Programa V
20 FOR contador=1 TO 25
30 PRINT contador,contador*2,contad
or*contador
40 NEXT contador
```

```
10 REM Programa VI
20 FOR contador=0 TO 20 STEP 2
30 PRINT contador
40 NEXT contador
```

```
10 REM Programa VII
20 FOR contador=0 TO 20 STEP 3
30 PRINT contador
40 NEXT contador
```

```
10 REM Programa VIII
20 FOR contador=0 TO 15
30 PRINT 15-contador
40 NEXT contador
```

```
10 REM Programa IX
20 FOR contador=15 TO 0 STEP -1
30 PRINT contador
40 NEXT contador
```

```
10 REM Programa X
20 FOR contador=2 TO 1
30 PRINT "ESTO ES ABSURDO"
40 NEXT contador
```

```
10 REM Programa XI
20 FOR contador=1 TO 10 STEP 2
30 PRINT contador
40 NEXT contador
```

```
10 REM Programa XII
20 FOR contador=10 TO 1 STEP -2
30 PRINT contador
40 NEXT contador
```

```
10 REM Programa XIII
20 FOR externo=1 TO 3
30 PRINT "Este es el bucle externo"
 externo
40 FOR interno=1 TO 3
50 PRINT "Este es el bucle interno"
 interno
60 NEXT interno
70 NEXT externo
```

Primeros pasos

incrementado en uno. El bucle para cuando la variable de control excede el valor especificado que figura después de TO.

En otras palabras, la variable de control actúa como un contador. Si queremos hacer algo 10 veces podemos usar nuestros dedos como variable de control. Veamos cómo el bucle FOR... NEXT, haría algo parecido:

FOR dedos=1 TO 10
haz algo
NEXT dedos

Cuando hemos utilizado los 10 dedos hemos hecho cualquier cosa que ha sido realizada en diez veces. Los dedos han actuado como una variable de control oscilando desde 1 TO 10.

En el Programa III está la línea:

FOR contador=1 TO 25

Le dice al **Amstrad** que debe repetir todas las líneas que están encima del siguiente NEXT. La variable de control del bucle es «contador», su valor inicial será 1 y cada vez que haga el bucle se incrementará en uno. El bucle repite el ciclo hasta que el valor final de «contador» sea mayor que 25.

La primera vez que se hace el bucle el **«contador»** es 1. El NEXT envía al programa de vuelta al FOR y **«contador»** se convierte en 2, luego en 3 cuando progresa el bucle y así hasta 25.

Entonces cuando NEXT envía al programa de vuelta a FOR, el 1 que se añade a **«contador»** le da el valor de 26. Esto excede el límite colocado por TO por lo que en este momento se sale del bucle, yendo el **Amstrad** a la línea siguiente a NEXT. En este caso no hay ninguna, por tanto el programa ha terminado.

El resultado es que todos los números entre 1 y 25 han sido escritos, ya que en los ciclos del bucle FOR los valores de «contador» van desde 1 a 25. Probemos añadiendo:

50 PRINT «El valor final de contador es» contador

y veremos que realmente tiene un valor final de 26.

FOR...NEXT = claridad + sencillez

Comparemos el Programa I con el Programa III. Creemos que estará de acuerdo en que el Programa III es más claro. Con la estructura FOR... NEXT es fácil ver que el bucle completará el ciclo 25 veces. Con el Programa I nunca estaremos seguros si lo hará 24 ó 25

veces así que la lección es que el bucle FOR... NEXT hace las cosas más claras. También son más fáciles de modificar que los bucles llenos de GOTOS.

Echemos un vistazo al Programa IV que hace el mismo trabajo que el Programa II pero utiliza un bucle FOR... NEXT.

Todavía conseguimos nuestros 25 HOLAS en la pantalla justamente como en el Programa II. Notemos, sin embargo, que así es más fácil hacerlos 500 ó 5.000 veces. Y todo lo que tenemos que hacer para obtener 305 es cambiar una línea.

20 FOR contador = 1 TO 305

nos da lo que queremos. Como podemos ver es más fácil que perdiendo el tiempo con el Programa II.

El bucle FOR... NEXT no es sólo sencillo, sino también poderoso, veamos en el Programa V lo que queremos decir.

Otra vez el bucle hace el ciclo 25 veces, pero ahora la línea que forma parte del cuerpo del bucle es diferente. No sólo imprime el valor de **«contador»** sino que también lo hace multiplicado por dos y elevado al cuadrado. Probemos alterar la línea para que nos de **«contador»** elevado al cubo o sumarle cinco en cada vuelta. Es fácil **¿no?**

Antes que dejemos los Programas III y IV veamos solamente si podemos observar alguna mayor diferencia entre sus maneras de trabajar. No se preocupe si no puede, es bastante difícil.

La respuesta es que el Programa III no sólo usa **«contador»** para controlar el número de repeticiones del bucle, sino que también utiliza **«contador»** en el cuerpo del mismo. El Programa V hace lo mismo.

El Programa IV, sin embargo, no utiliza **«contador»** en el cuerpo principal del bucle, lo usa normalmente para mantenerse al tanto de los ciclos. Como dijimos es un punto oscuro pero vale la pena tenerlo en cuenta.

Si utilizamos la variable de control del bucle en el interior del cuerpo principal del mismo, en caso de cambiarla debemos hacerlo con cuidado o alterará la cuenta del bucle.

Una línea como:

25 contador = contador + 3

en el Programa III demostrará lo que queremos decir. Aunque es bastante obvio en este caso, en un programa largo puede ocasionarnos problemas. ¡Cuidado!

Marcapasos = STEP

Hasta aquí todos nuestros bucles FOR... NEXT han tenido la variable de control aumentando en pasos de 1 cada vez que se hacia el bucle. Esto no tiene que ser siempre así. Podemos cambiar el número de pasos que queramos usar el comando correctamente llamado STEP. El Programa VI lo utiliza para escribir todos los números pares entre 0 y 20.

Es un comando fácil de usar. Como se desprende del Programa, la cifra después de STEP decide cuánto debe ser incrementada la variable de control al final de cada ciclo. En este caso **«contador»** se incrementa en 2 en cada vuelta. **¿Qué ocurriría si el STEP fuera 4?** Probemos e imaginémoslo fuera antes de intentarlo con el micro.

El programa VII demuestra que ocurre cuando el STEP es 3.

No hay problema con la mayoría de los resultados que salen. Se parte de 0, y se añaden tres hasta alcanzar 18. Pero **¿por qué se para en 18?** Si añadimos:

50 PRINT **«El valor final de contador es»** contador

al final del programa veremos por qué.

Cuando **«contador»** es 15 NEXT le añade 3, haciéndole 18 y envía al bucle de vuelta a la línea 20. La línea 30 imprime el 18 y el programa va otra vez al NEXT.

Aquí **«contador»** se incrementa de nuevo en tres, haciéndole 21 y el programa regresa al FOR.

Ahora cuando el micro lo revisa, **«contador»** está fuera de rango *(0 a 20)* así que se salta el bucle. Por lo tanto 18 es el último número que se ha escrito.

Probemos cambiar STEP a 5 u 8 y veamos si podemos explicar los resultados.

Se habrá dado cuenta que en el último programa junto al NEXT de la línea 40 no se puso un **«contador»** pero sin embargo el programa funciona. Desde aquí podemos ver que no siempre NEXT tiene la etiqueta *(label)* de la variable de control del bucle.

Habiendo dicho esto le aconsejamos que la ponga también, ya que así es más fácil depurar los programas.

Todo esto está bien, si queremos números producidos por un bucle en orden creciente, pero, por alguna razón que conozca usted mismo, **¿quiere que el valor de los números decrezca?**

El Programa VIII muestra la manera de hacerlo.

Aquí **«contador»** todavía parte desde cero, aumentando uno cada vez, abarcando un rango de 0 a 15. No obstante, en lugar de imprimir el valor de **«contador»,** ahora se lo restamos a 15 cada vez que hace el bucle. En la primera vuelta **«contador»** es 0 de modo que aparece en pantalla el 15. La segunda vez **«contador»** se ha decrementado en 1 así que aparece 14 **(15-2),** se sigue en el ciclo siguiente con 13 **(15-2).**

Como el rango de **«contador»** va de 0 a 15, el resultado de la sustracción va de 15 a 0. ¿Puede hacerlo bajar de dos en dos?

Cuenta atrás

El Programa IX muestra una manera sencilla de contar hacia atrás.

El secreto se encuentra en el parámetro STEP —1. Cada vez que se hace el bucle se añade —1 al **«contador».** El resultado de sumar —1 cada vez que se hace el bucle es que «contador» decrece.

Observe que las cifras a ambos lados de TO han cambiado para hacer frente a esto. Ahora van de un valor inicial de 15 a un valor final de 0.

Probemos a utilizar otros intervalos negativos y veremos qué pasa. Son bastante fáciles de utilizar, pero hay que asegurarse que los valores de comienzo y finalización están de acuerdo con los intervalos.

Si no lo comprende veamos en el Programa X lo que queremos decir.

Probemos:

STEP —1

o

FOR contador = 1 TO 2

para corregirlo.

Después de entrenarse con los bucles FOR... NEXT en casa, no debe tenerle miedo al Programa XI. Debe ser capaz de comprender que el último número que saldrá en pantalla será 9.

Advierta que cuando STEP es positivo, el bucle termina cuando la variable de control excede el límite señalado.

Si tiene alguna duda:

50 PRINT **«El último valor es»** contador

debe convencerle, de hecho 11 es mayor que el 10 que hay detrás de TO.

De cualquier forma, cuando STEP es negativo como en el Programa XII las cosas son distintas. El bucle termina cuando la variable de control es menor que la cifra que hay después de TO.

De nuevo:

50 PRINT **«El último valor es»** contador

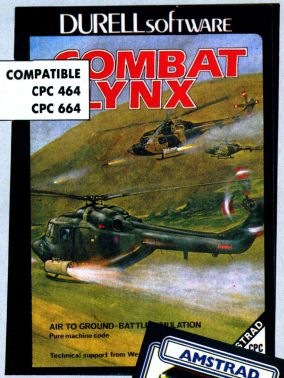
añadido al final del programa debe convencerle. No hay duda que 0, el valor final de «contador», es menor que 1, el límite señalado en la línea 20.

Y esto es todo más o menos, por ahora. Hasta la próxima vez, ¿por qué no escribe sus propios programas usando pasos positivos y negativos? ¿Y por qué no usa valores fraccionarios en lugar de números enteros?, ¿lo damos por hecho? Y si, después de todo, tiene algún tiempo libre, ¿puede explicar el Programa XIII?

¿Qué ocurre aquí? Lo explicaremos...

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.

BASE DE DATOS: LA INFORMACION A MANO

Una de las tareas para lo que los ordenadores parecen haber sido creados con toda la alevosía del mundo, son los programas de bases de datos, el manejo organizado de la información.

Imagínese una escena, sombría escena, como la siguiente: usted tiene 1.000 clientes en su negocio, cada uno de los cuales realiza una serie de pedidos, que deben ser servidos en una fecha determinada, de un producto específico.

Usted tiene que averiguar, por ejemplo, qué cantidad de lo que sea, y a quién, debe servirse antes del día tal, y tiene que hacerlo a mano, esto es, mirarse las 1.000 fichas y anotar, una por una, aquellas que cumplan con las especificaciones requeridas.

Al menos para mí, un panorama como el que antecede, lo llamaría, pecando de coloquial, de auténtico suicidio.

Al parecer, gran cantidad de gente comparte esta opinión, porque cuando surgieron los primeros programas que realizaban estas ingratas tareas, se vendieron en cantidades industriales.

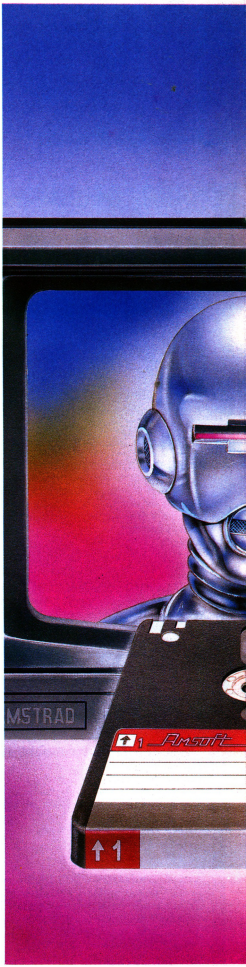
Tales aplicaciones se conocen como bases de datos, y constituyen la mejor manera encontrada hasta ahora de relacionar información de temas muy dispares de forma coherente, en base a unas ciertas reglas o criterios de selección definidos por el usuario.

La ventaja estriba, lo recalco, en el hecho de que una base de datos puede ser interrogada en un lenguaje más o menos natural *(depende del diseño del programa y de la «astucia» del programador)*, y obtener un listado por pantalla o impresora, de, por ejemplo, **«todos los clientes que han pedido 1.000 kg de madera de abeto, a entregar antes del 15 de marzo y que residen en Madrid por la tarde de 4 a 5»**.

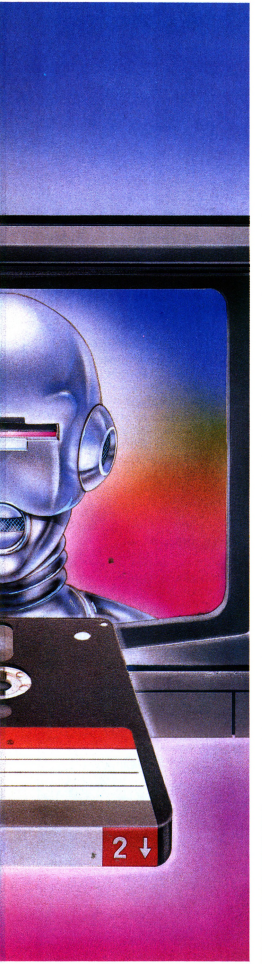
La complejidad y longitud del criterio de selección, como el anterior, puede alargarse todo lo que queramos. Lógicamente, cuanto más complejo sea, y más cantidad de información deba manipular el programa en cuestión, el tiempo de respuesta se alargará más.

El número de programas de bases de datos puede contarse por millares; hay de todos los tipos y concepciones.

En el caso particular de **Amstrad,** también hay varias. Hemos elegido para Banco de Pruebas una creada por RPA Sistems, que así, a grosso modo, posee dos particularidades interesantes: está escrita en Basic, y no necesita del CP/M para funcionar, es decir, corre bajo Amsdos, sistema operativo nativo de **Amstrad.** Esto nos permitirá observar una aplicación de gestión desde un punto de vista distinto de la aproximación CP/M.



Banco de pruebas



Más abajo, dentro de este mismo artículo, analizamos en detalle el manejo del programa prácticamente comando a comando, por lo que vamos a centrarnos aquí en algo que, a mi modo de ver, es especialmente importante: sencillez en el uso del programa, de cara al usuario desconocedor de la informática.

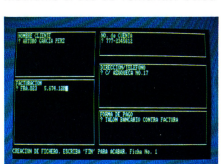
En primer lugar, RPA Sistems no entrega manual de instrucciones. Ni falta que hace, fue mi primera reacción al trabajar con él *(luego pude comprobar que existe una opción para ver las instrucciones por pantalla o impresora)*, y creo que es acertada. Este programa es la sencillez por antonomasia, es muy difícil equivocarse.

En todo momento, el usuario se ve asistido por mensajes del sistema, absolutamente unívocos,, que nos indican, paso a paso, lo que se espera que hagamos o las opciones de que disponemos. El programa está basado en una estructura de menú: existe un menú principal, en el que decidimos lo que deseamos hacer, y luego el programa bifurca hacia la opción elegida.

Pude observar que en este menú, las posibilidades van numeradas, en lugar de acceder a ellas por su inicial.

Bien, supongo que es cuestión de gustos o costumbre, pero una y otra vez mi dedo se lanzaba a las letras en lugar de a los números, con lo que, o no obtenía nada, o me encontraba metido de lleno en la opción de dar colores a la pantalla cuando pretendía crear un fichero *(sí, Color también empieza por C)*.

Está muy claro que criticar esto es un tanto discutible, pero basándome en el hecho de que al resto de las personas que manejaron el programa les ocurrió lo mismo, tal vez hubiera sido más acertado escoger otro sistema. Veredicto: en el peor de los casos, mal menor; en la mayoría de ocasiones, cuestión de gustos. La creación de un fichero se hace de forma completamente interactiva, mediante el uso de ventanas. Usted simplemente mueve el cursor por la pantalla, fija dos puntos y ya está, tenemos creada una ventana que corresponderá a un campo de la base de datos, en la cual, una vez etiquetada con algo como **«nombre», «cliente número»**, o lo que sea, se almacenará la información relativa a ese concepto.



Podemos tener simultáneamente en pantalla hasta 8 ventanas, lo cual no implica que sólo puedan existir 8 campos; como se verá en la descripción del programa, esto ha sido previsto.

No podemos por menos que felicitar al programador por conseguir la siempre difícil tarea de crear el formato de una base de datos, de una forma tan sencilla y elegante.

En el mismo estilo se ubica el tema de dar color a la pantalla diseñada con las ventanas: aparece otra pantalla con una serie de informes acerca de los colores disponibles. Para cambiarlos, se pulsan una serie de teclas y el nuevo set de colores aparece en la pantalla. En definitiva, escogemos visualmente, como debe ser, la combinación que más nos acomode. Nada de cosas como **«Introduzca un número del 0 al 27 para cambiar la tinta...»**, o algo igualmente esotérico.

El resto de las opciones son las que cabe esperar de este tipo de programas: ordenación por uno o varios campos, búsqueda selectiva de información por un determinado campo, sacar fichas por impresora, etc., todas igualmente simples de manejar desde el primer momento.

Pudimos observar que el programa se encuentra dividido en multitud de partes en el disco, de forma que cada una está en memoria sólo cuando se la necesita. A pesar de ello, no se observa un retraso **«crispante»** en las operaciones: unos instantes de espera, y nos encontramos donde queríamos estar. Las tareas de ordenació y búsqueda de fichas se realizan también a una velocidad aceptable.

Hablando de ordenación, hay que andarse con cuidado con la opción de borrar una ficha: esto se realiza pulsando una tecla, y la ficha, sea cual sea, que se encuentre en pantalla, desaparece sin previo aviso.

Esto es lógico hasta cierto punto, ya que supone que el usuario se ha colocado antes en la ficha que quiere borrar, pero puede causar serios disgustos al mínimo despiste. No hubiera estado de más hacer la operación de borrado un poco más dificultosa, o pedir confirmación, o algo que obligará a concentrar la atención sobre lo que estemos haciendo. Una vez borrada, no se puede recuperar.

Por último, respecto al número de fichas que caben en un fichero, es variable. El programa pregunta en un momento dado el número de caracteres por campo, ofreciendo por defecto el número de 15.

Con un fichero de 3 campos, y con 15 caracteres por campo, el número de fichas posibles calculadas por el programa fue de 628. No obstante, creo que ese número es aproximativo, ya que en cada campo cabe todo el texto que queramos meter, o al menos, más de 15 caracteres, por lo que cabe esperar que el número real de fichas se vea afectado si se mantienen en memoria central.

No conseguimos arrancar el programa en un CPC464 con unidad de disco, sin embargo, funcionó perfectamente en un CPC664 y en el CPC6128.

Pasamos ahora a describir con detalle las opciones y manejo del programa, dejando que cada cual, a tenor de lo expuesto, saque sus propias conclusiones.

INSTRUCCIONES

El menú principal consta de nueve opciones, más las instrucciones:

CREAR FORMATO
CREAR FICHERO
ORDENAR
NUEVA ORDENACION
AÑADIR FICHAS
REVISAR
GRABAR
CARGAR
IMPRESORA

1 CREAR FORMATO

Esta opción nos permite definir el formato de un fichero de la siguiente manera:

Podemos escoger el modo 40/80 columnas. Con 40 columnas tendremos 4 colores, con 80, 2.

Con las teclas de movimiento del cursor, éste se mueve hasta la posición deseada y se pulsa 'Q' para marcar los límites.

Aparecen unas líneas delimitando la esquina superior izquieda. Ahora el cursor sólo puede recorrer el cuadrante inferior derecho, se lleva a la posición donde queremos marcar el fin de la ventana, y pulsamos 'Q' de nuevo.

La ventana se ilumina y al pie de pantalla aparece la pregunta: más (s/n)? Si responde 's' podrá definir más ventanas visualizando las anteriores.

Se puede pulsar 'B' para borrar la ventana anterior si no ha quedado bien. Cuando defina más de siete ventanas, verá el mensaje: NO HAY ESPACIO PARA MAS VENTANAS. ¿SEGUIMOS O LO DEJAMOS?

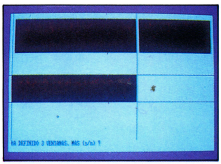
Si responde 's' ya sólo tiene que responder a la pregunta: ¿CUANTOS CAMPOS MAS?

A medida que vaya definiendo la base de su archivo, el programa le mostrará como está quedando y le preguntará: ¿CORRECTO?, por si quisiera rectificar.

Los próximos datos a introducir son los nombres de cada campo, que podrían ser NOMBRE, DIRECCION, etc..., o similares.

Una vez terminado el modelo del fichero, el programa le pedirá el promedio de caracteres por campo (si responde ENTER, se tomará el valor de 15 caracteres por campo), para calcular así el número máximo de fichas.

Para ordenar, puede interesar hacerlo por pocos campos, acelerando la ordenación, o por todos. Deberá especificar por qué campos quiere ordenar, el sistema es muy sencillo y basta con seguir los mensajes.



Ahora ya tiene su MODELO de fichero, que le permitirá crear, revisar, actualizar, etc... Es aconsejable grabar este MODELO en la 2.ª cara de un disco (opción SALVAR).

2 CREAR FICHERO

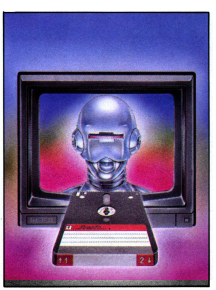
Para seleccionar esta opción, es obligatorio haber definido previamente un MODELO de fichero o haberlo cargado de un disco, de lo contrario el ordenador no sabrá COMO presentar los datos.

Crear un fichero consiste en rellenar las fichas de información, siempre que no se supere el máximo establecido. Para acabar la introducción de datos, escriba FIN en cualquier campo seguido de ENTER para volver al menú principal.

3 ORDENAR

Consiste en la clasificación alfabética del fichero según los campos especificados al CREAR FORMATO. Esta clasificación se puede modificar en cualquier momento con la opción 4 (NUEVA ORDENACION).

Al estar en revisión, la opción 'C' nos permitirá cambiar la numeración



de las fichas con respecto a un campo u otro.

4 NUEVA ORDENACION

Esta opción permite seleccionar los campos por los que se va a efectuar la ordenación, utilizando las teclas del cursor para seleccionar un nuevo campo y pulsando 'P' para fijar ese campo.

Pulsando 'T' la ordenación se hará por todos los campos, pulsando 'Q' finaliza ésta y deja los campos marcados anteriormente como campos matrices de la ordenación.

5 AÑADIR FICHAS

Obviamente, sirve para añadir nuevos elementos (fichas) a la base de datos. Para acabar, basta escribir FIN en cualquier campo seguido de ENTER.

6 REVISAR

En este modo se puede, además de visualizar las fichas, modificar, borrar, buscar, etc... Para facilitar la tarea esta opción posee un submenú de comandos, a los que se accede de pulsando «A».

Los comandos de REVISAR son: *'A' ayuda, le dice los comandos disponibles.)*

Pra visualizar los campos que no han cabido en las ventanas, es decir cuando hay más de siete campos.

'P' ficha posterior, 'O' ficha anterior.

'L' borra la ficha que hay en pantalla. Esta operación desordena el fichero.

'N' nueva ficha: conociendo su número con respecto a la ordenación escogida, aparecerá en pantalla.

'C' modifica el campo que determina la numeración. El sistema es análogo al empleado en NUEVA ORDENACION, pero sólo se pueden seleccionar los campos que se eligieron para ordenar. Si se quieren cambiar, debemos usar NUEVA ORDENACION.

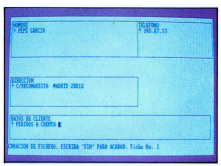
'M' modificar: se selecciona con las teclas de cursor el campo a modificar y se pulsa 'Q', pasando a introducir la nueva información. Se puede insertar CADENA, escribiendo '$ CADENA', o añadir CADENA, escribiendo 'CADENA'.

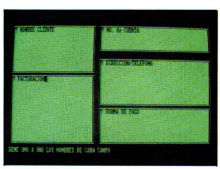
'B' buscar. Seleccionar con las teclas de cursor los campos en los que se busca, fijándolos con 'P'. Quedarán señalados con una flecha, hasta que pulse 'Q'.

El ordenador pide, entonces, la cadena a buscar, y comenzará en los campos seleccionados. A medida que vaya encontrando fichas con esa información, el ordenador parará la búsqueda. Ahora hay dos posibilidades:

Pulsando 'Q' se vuelve al modo REVISION con esa ficha en pantalla. Si se pulsa otra tecla la búsqueda continuará.

'Q' permite volver al menú, para grabar datos, cargar, imprimir, etc.





7 SALVAR

Permite escribir en disco dos tipos de información:

1) el MODELO del fichero (talla de ventanas, nombres, colores).

2) un fichero de datos (clientes, direcciones, etc.).

Es importante aclarar que, si se elige grabar los dos, sólo se pedirá UN nombre. Si respondemos por ejemplo: NOMBRE, el modelo del fichero se llamará NOMBRE.MOD, y el fichero se llamará NOMBRE.DAT, aunque no se debe especificar NUNCA .MOD o .DAT., al dar el nombre de fichero (el apellido lo pone el propio programa).

Sólo debe dar el nombre sin extensión, el ordenador sabe cuándo buscar .MOD o .DAT. Cualquier fichero creado con N campos puede funcionar con otro modelo que tenga también N campos, sin importar su modo, colores o nombres.

Banco de pruebas

8 CARGAR

Hace lo contrario que SALVAR. Se encontrará con la misma pregunta, y al igual que antes, podemos dar sólo UN nombre de fichero, siendo NOMBRE.MOD el nombre del modelo y NOMBRE.DAT el del fichero.

9 IMPRESORA

Respondiendo DESDE que ficha y HASTA que ficha, (pudiendo cancelar dando 'Q') se listará por impresora el contenido de las fichas especificadas.

Existen otras tres opciones: INSTRUCCIONES pulsando 'I', cambiar colores de base, pulsando 'C' y ABANDONAR, pulsando 'O'. Esta última opción le pedirá confirmación, ya que provoca un borrado de los datos y del programa.

FICHA DEL PROGRAMA

Creado por: **RPA systems**
Comercializado por: **RPA systems**
Soporte: **Disco**
Sistema operativo: **Amsdos**
Compatibilidad: **CPC664/CPC6128**
P.V.P.: **6.500 ptas.**

FRUTAS

COMPATIBLE
CPC 464
CPC 664

Frutas es un programa creado para emular a las famosas «máquinas de los bares». Todo el mundo sabe en que consisten, así que cualquier explicación resulta ociosa. Sólo decir quelos gráficos son excelentes, como puede verse en las fotos, y que el programa lleva cerca de 2 Kbytes y medio en lenguaje máquina, responsables de la perfección de los gráficos y de su movimiento. El autor del programa nos explica ahora el resto de la historia:

Este programa ha sido realizado por el lector:
Andoni Martín
Bilbao

Este programa escrito en Basic y Código Máquina es un juego parecido a las máquinas tragaperras de los bares. Partimos con una cantidad limitada de dinero, obsequiada generosamente por nuestro CPC-464. Cada tirada cuesta 10 unidades monetarias si tenemos menos de 100, 20 si tenemos más de 100 y menos de 200... y así sucesivamente. Para jugar, debemos de pulsar la tecla ENTER. Mientras esté pulsada, se encenderán unas luces encima del marcador. Cuando la soltemos, las frutas darán tantas vueltas como luces encendidas haya. Obtendremos bonus automáticos si en la pantalla de frutas hay más de 4 iguales, contando los bonus por el valor de la fruta en cuestión. Tanto si hemos conseguido bonus automáticos como si no, tenemos la posibiladad de lograr bonus manuales al parecer en pantalla la frase: BONUS CON UNA LUZ. Para conseguirlos, debemos lograr que se encienda tan sólo una luz, mediante la pulsación de la tecla ENTER. La manera más fácil de hacerlo es tener ENTER pulsada hasta que las luces den una o más vueltas y soltarla justo en el momento en que sólo encendida una luz. Si lo logramos iremos sumando bonus. Cuando fallamos, y si tenemos bonus, aparece en pantalla la indicación: MOVER FRUTAS O COGER.

El propósito es mover las frutas de modo que la combinación premiada quede en la línea que marcan las flechas. Para mover las frutas nos ayudamos del keypad numérico: cada uno de los 9 cuadros de la pantalla de frutas se corresponde con los dígitos 1-9 del keypad,siendo 1 el cuadro inferior izquierda. Para mover 2 frutas hay que pulsar los dígitos de los cuadros donde están situadas.

Sólo se pueden mover frutas adyacentes horizontal o verticalmente. Por ejemplo, movimientos como 1-9 ó 5-7 son ilegales u no serán aceptados. Sin embargo, solamente se permiten los movimientos, horizontales del tipo 2-3, 7-8, ... si tenemos más de 3 bonus. En caso contrario, sólo podemos mover verticalmente. Cada movimiento legal gasta un bonus. Cuando hayamos dispuesto las frutas como queríamos, en función de los bonus en nuestro haber, o cuando ni siquiera queríamos mover las frutas, debemos pulsar una tecla cualquiera *(por ejemplo, ENTER de nuevo)* para coger el premio, caso que lo hubiera. Y eso es todo. Jugando un par de veces se observa que es muy fácil de manejar y muy adictivo. Si nos quedamos sin dinero, observaremos un bonito efecto de templado de imagen, realizado con sucesivos scrolls. El orden de importancia de las frutas, de menor a mayor, es: Cereza, Plátano, Fresa, Limón, Manzana y BAR. El tipo de premios es: 3 frutas iguales, 2 iguales con BAR en medio y 2 BARES en las esquinas.

El propio programa se encarga de encargar el código máquina grabado a continuación de él y que contiene los dibujos de las frutas y la rutina que genera su representación en pantalla.

```
1000 REM INICIALIZACION
1010 RESTORE 1010:FOR i=0 TO 15:REA
D a:INK i,a:NEXT:DATA 0,24,1,3,10,4
,7,9,0,15,12,6,11,12,26,16
1020 INK 8,18,0:MODE 0:BORDER 0:PAP
ER 0:PEN 2:CLS
1030 FEN 8:LOCATE 2,11:PRINT"NO SEA
S IMPACIENTE":GOSUB 2050
1040 FOR n=1 TO 3:a$(n)="000":NEXT:
ex=40000:col=ex+1:fil=ex+3:car=ex+5
1050 REM CARACTERES GRAFICOS
1060 SYMBOL 251,24,28,254,255,255,2
54,28,24
1070 SYMBOL 252,0,0,31,31,31,31,0,0
1080 SYMBOL 253,24,56,127,255,255,1
27,56,24
1090 SYMBOL 254,0,0,248,248,248,248
,0,0
1100 SYMBOL 255,0,60,126,126,126,12
6,60,0
1110 REM DIBUJOS
1120 sc=50:CLS:PEN 9:LOCATE 3,8:PRI
NT CHR$(252)+CHR$(251):LOCATE 3,9:P
RINT CHR$(252)+CHR$(251)
1130 LOCATE 18,8:PRINT CHR$(253)+CH
R$(254):LOCATE 18,9:PRINT CHR$(253)
+CHR$(254)
1140 FOR m=144 TO 400 STEP 128:FOR
n=232 TO 392 STEP 80:MOVE m,n:PLOT
m,n:DRAWR 128,0:DRAWR 0,-80:DRAWR
-128,0:DRAWR 0,80:NEXT:NEXT
1150 GOSUB 1600
1160 REM PROGRAMA PRINCIPAL
1170 GOSUB 1510:PEN 10:LOCATE 1,25:
PRINT" ENTER PARA JUGAR ";
1180 GOSUB 1420
1190 GOSUB 1530:sc=sc-10*(sc\100+1)
1200 GOSUB 1600
1210 GOSUB 1660
1220 GOSUB 1710
1230 GOSUB 2010
1240 IF bn>0 THEN GOSUB 1780:GOSUB
1510:GOTO 1240
1250 LOCATE 1,21:PRINT SPACE$(20);
1260 GOSUB 1930:z=z*10:IF pr>0 TH
EN GOSUB 1620 ELSE SOUND 1,2000,100
:FOR i=1 TO 750:NEXT
1270 sc=sc+pr:GOSUB 1600
1280 IF sc*2>0 THEN 1170
1290 IF INKEY$<>"" THEN 1290
1300 FOR N=1 TO 3:PRINT CHR$(30)+CH
R$(11)+NEXT
1310 WHILE INKEY$="":WEND
1320 LOCATE 1,26:PRINT:LOCATE 1,26:
PRINT:LOCATE 1,26:PRINT
1330 PRINT CHR$(30)+CHR$(11):PRINT
CHR$(30)+CHR$(11):PRINT CHR$(30)+CH
R$(11)
1340 WEND
1350 FOR n=1 TO 25:LOCATE 1,26:PRIN
T:NEXT
1360 FOR n=1 TO 12:LOCATE 1,26:PRIN
T:NEXT:PEN 6:LOCATE 2,25:PRINT"QUIE
RES JUGAR MAS?":FOR n=1 TO 14:LOCAT
E 1,26:PRINT:NEXT
1370 a$=UPPER$(INKEY$):IF a$<>"S" A
ND a$<>"N" THEN 1370 ELSE IF a$="S"
THEN 1110
1380 FOR n=1 TO 12:LOCATE 1,26:PRIN
T:NEXT:PEN 9:LOCATE 4,25:PRINT"SEGU
RO QUE NO?":FOR n=1 TO 14:LOCATE 1,
26:PRINT:NEXT
1390 a$=UPPER$(INKEY$):IF a$<>"S" A
ND A$<>"N" THEN 1390 ELSE IF a$="N"
THEN 1360
1400 FOR n=1 TO 12:LOCATE 1,26:PRIN
T:NEXT:CALL 0
1410 REM APAGAR LUCES
1420 PEN 2:LOCATE 3,18:PRINT STRING
$(17,CHR$(255));:RETURN
1430 REM PROBABILIDADES
1440 IF a<4 THEN a=6:RETURN:REM BAR
3%
1450 IF a<12 THEN a=5:RETURN:REM MA
NZANA 8%
1460 IF a<25 THEN a=4:RETURN:REM LI
MON 13%
1470 IF a<43 THEN a=3:RETURN:REM FR
ESA 18%
1480 IF a<68 THEN a=2:RETURN:REM PL
ATANO 25%
1490 a=1:RETURN:REM CEREZA 33%
1500 REM PRINT BONUS
1510 FEN 11:LOCATE 8,23:PRINT"BONUS
";RIGHT$(STR$(bn),1):RETURN
1520 REM ENCENDER LUCES
1530 k=2:N=200
1540 IF INKEY(18)<>0 THEN 1540
1550 k=k+1:N=N-10:IF k<20 THEN PEN
2:GOSUB 1420:GOTO 1530
1560 PEN 4:LOCATE K,18:PRINT CHR$(2
55):SOUND 7,N,1,2
1570 IF INKEY(18)=0 THEN 1550
1580 RETURN
1590 REM SCORE
1600 PEN 6:LOCATE 6,20:PRINT"TIENES
:"LEFT$("0000",5-LEN(STR$(sc)))RI
GHT$(STR$(sc),LEN(STR$(sc))-1):RETU
RN
1610 REM PREMIO
1620 PEN 6:LOCATE 6,21:PRINT"PREMIO
:"LEFT$("0000",5-LEN(STR$(pr)))RI
GHT$(STR$(pr),LEN(STR$(pr))-1)
1630 FOR n=1 TO pr/10:FOR m=100 TO
20 STEP -10:SOUND 1,m,7,3:NEXT:a=IN
T(RND*26+1):BORDER a:INK 0,a*NEXT:B
ORDER 0:INK 0,0
1640 LOCATE 1,21:PRINT SPACE$(20):R
ETURN
1650 REM TIRADA
1660 LOCATE 1,25:PRINT SPACE$(20);:
FOR J=K TO 3 STEP -1:FOR n=5 TO 13
STEP 4:SOUND 1,n*10,1 STEP 5:POKE fi
l,n:POKE col,n+a:INT(RND$100+1):GOS
UB 1440:POKE car,(a-1)$12+1:CALL 40
100
1670 IF j=3 THEN MID$(a$(m\5+1),n\4
,1)=RIGHT$(STR$(a),1)
1680 NEXT:NEXT:FEN 2:LOCATE j,18:PR
INT CHR$(255):SOUND 7,1000,10:NEXT
1690 RETURN
1700 REM BONUS AUTOMATICOS
1710 bn=0:DIM b(6):FOR h=1 TO 6:b$=
RIGHT$(STR$(h),1)
1720 FOR p=1 TO 3:FOR q=1 TO 3:IF M
ID$(a$(p),q,1)=b$ THEN b(h)=b(h)+1
1730 NEXT:NEXT:NEXT
1740 FOR h=6 TO 1 STEP -1:IF b(h)>4
 THEN bn=h ELSE NEXT
1750 GOSUB 1510
1760 ERASE b:RETURN
1770 REM COLLECT O USAR BONUS
1780 FEN 7:LOCATE 1,25:PRINT"MOVER
FRUTAS O COGER";
1790 IF INKEY$<>"" THEN 1790
1800 i1$=UPPER$(INKEY$):IF i1$="" T
HEN 1800
1810 IF i1$<"1" OR i1$>"9" THEN GOS
UB 1600:bn=0:LOCATE 1,25:PRINT SPAC
E$(20);:RETURN
1820 LOCATE 1,25:PRINT SPACE$(20);
1830 b1=3:b2=3:IF bn>3 THEN b2=1 .
1840 i2$=UPPER$(INKEY$):IF i2$<"1"
OR i2$>"9" THEN 1840
1850 i(1)=VAL(i1$):i(2)=VAL(i2$):d=
ABS(i(1)-i(2)):IF d=0 OR (d<>b1 AND
 d<>b2) THEN 1780
1860 FOR p=1 TO 2:r=i(p):m(p)=1-5$(
r<7)-5$(r<4)-n(p)=-5$(r=1 OR r=4 OR
 r=7)-9$(r=2 OR r=5 OR r=8)-13$(r=3
OR r=6 OR r=9):a1(p)=(m(p)+3)\4:a2
(p)=n(p)\4:NEXT
1870 a(1)=VAL(MID$(a$(a1(2)),a2(2),
1)):a(2)=VAL(MID$(a$(a1(1)),a2(1),1
))
1880 POKE fil,m(1):POKE col,n(1):PO
KE car,(a(1)-1)$12+1:CALL 40100
1890 POKE fil,m(2):POKE col,n(2):PO
KE car,(a(2)-1)$12+1:CALL 40100
1900 b$=MID$(a$(a1(1)),a2(1),1):MID
$(a$(a1(1)),a2(1),1)=MID$(a$(a1(2)
),a2(2),1):MID$(a$(a1(2)),a2(2),1)=b
$:bn=bn-1
1910 RETURN
1920 REM PREMIOS
1930 IF a$(2)="666" THEN pr=50:RETU
RN
1940 IF LEFT$(a$(2),1)="6" AND RIGH
T$(a$(2),1)="6" THEN pr=20:RETURN
1950 IF a$(2)="565" OR a$(2)="565"
THEN pr=10:RETURN
1960 IF a$(2)="444" OR a$(2)="464"
THEN pr=8:RETURN
1970 IF a$(2)="333" OR a$(2)="363"
THEN pr=6:RETURN
1980 IF a$(2)="222" OR a$(2)="262"
THEN pr=4:RETURN
1990 IF a$(2)="111" OR a$(2)="161"
THEN pr=2:RETURN
2000 pr=0:RETURN
2010 PEN 12:LOCATE 3,25:PRINT"BONUS
 CON ESPACIO";
2020 GOSUB 1530:IF k=3 AND bn<9 THE
N bn=bn+1:GOSUB 1510:GOTO 2020
2030 GOSUB 1420:LOCATE 1,25:PRINT S
PACE$(20);:RETURN
2040 REM CARGA DEL CODIGO MAQUINA
2050 MEMORY 37695:RESTORE 2910:FOR
i=37696 TO 40040:READ a$:POKE i,VAL
("&"+a$):NEXT
2060 RESTORE 2070:FOR i=0 TO 54:REA
D a$:POKE (40100+i),VAL("&"+a$):NEX
T
2070 DATA DD,E5,DD,21,A0,9C,DD,66,0
1,DD,6E,03,DD,7E,05,06,04,E5,C5,D6,
03,DB,74,01,DD,75,03,DD,77,05,E5,C5
,F5,DD,E5,CD,A0,9C,DD,E1,F1,C1,E1,2
4,3C,10,E6,C1,E1,2C,10,DD,DD,E1,C9
2080 DATA 0,0,0,0,0,0,0,0,0,0,aB
9000 CLEAR:SAVE "frutas.jue":SAVE "
frutas.bin",b,37696,2363
2910 DATA 0,0,0,0,0,0,0,0,0,AB
2920 DATA 0,0,0,FC,0,0,0,AC,FC,0
2930 DATA 0,FC,5C,0,0,54,AC,0,0,54
2940 DATA AC,0,0,0,0,0,0,0,0,0
2950 DATA 0,0,FC,0,0,FC,FC,0,0,FC
2960 DATA AC,AB,54,AC,5C,FC,54,5C,
FC,FC
2970 DATA FC,5C,FC,0,0,0,0,FC,AB
2980 DATA 0,FC,AC,AB,0,AC,5C,AB,0,
5C
2990 DATA FC,0,0,FC,FC,0,0,FC,AB,0
3000 DATA 0,FC,0,0,0,0,0,0,FC,0
3010 DATA 0,0,FC,0,0,0,54,0,0,0
```

SUBRUTINAS

1420:	Apagar luces.
1440:	Probabilidades de las frutas.
1510:	Imprimir bonus.
1530:	Encender luces.
1600:	Marcador.
1620:	Obtención de premio.
1660:	Tirada, Llama al Código Máquina.
1710:	Bonus automáticos.
1780:	Mover frutas o coger.
1930:	Premios.
2010:	Lograr bonus al encender sólo una luz.
2050:	Cargador del Código Máquina.

```
3020 DATA 0,0,0,0,0,0,0,0,0,0
3030 DATA 0,0,0,0,0,0,0,5C,AC,FC
3040 DATA FC,5C,5C,FC,0,AC,FC,AB,
0,FC
3050 DATA AB,AB,0,AB,AB,AB,0,AB,
AB,54
3060 DATA 0,AB,AB,54,0,AB,54,0,AB,
0
3070 DATA 0,0,0,0,0,0,0,0,0,0
3080 DATA 0,0,0,0,0,0,0,0,0,0
3090 DATA 0,0,0,0,0,0,0,0,0,0
3100 DATA 0,0,0,0,54,0,0,0,54,0
3110 DATA 0,0,54,0,0,0,AB,0,0,CF
3120 DATA AC,0,45,DE,BE,0,45,CF,
BE,0
3130 DATA 45,CF,BE,0,54,0,AB,0,54,
0
3140 DATA 54,0,54,0,54,45,DE,BA,0,
CF
3150 DATA ED,ED,0,CF,CF,AC,CF,CF,
CF,4D
3160 DATA DE,CF,CF,4D,CF,0,0,0,0,0
3170 DATA 0,0,0,0,0,0,AB,0,0
3180 DATA 0,AB,0,0,0,ED,0,0,0,DE
3190 DATA BA,0,0,DE,BA,0,0,0,45,CF
3200 DATA BE,0,45,CF,BE,0,45,CF,
BE,0
3210 DATA 45,CF,BE,0,45,CF,6F,0,0,
CF
3220 DATA CF,0,0,0,0,0,0,0,0,CF
3230 DATA CF,4D,CF,CF,CF,4D,CF,CF,
CF,4D
3240 DATA CF,CF,9F,4D,CF,4D,CF,4D,
CF,0
3250 DATA 0,45,CF,0,0,0,CF,0,0,0
3260 DATA 0,CF,BA,0,0,CF,BA,0,0,CF
3270 DATA BA,0,0,9F,BA,0,0,9F,BA,0
3280 DATA 0,6F,BA,0,0,CF,0,0,0,0
3290 DATA 0,0,0,0,0,0,0,0,0,0
3300 DATA 0,0,0,0,0,0,0,0,0,0
3310 DATA 0,0,0,0,0,0,0,0,0,0
3320 DATA 0,0,0,0,0,0,0,0,0,0
3330 DATA 0,0,54,0,0,0,54,0,0,0
3340 DATA 40,0,0,0,40,0,0,0,40,0
3350 DATA 0,0,0,0,0,0,D4,0,0,0
3360 DATA 0,FC,0,0,0,AB,54,FC,0,AB
3370 DATA 54,AB,0,B0,40,AB,0,B0,
40,B0
3380 DATA 0,B0,C0,B0,0,B0,D4,B0,0,
0
3390 DATA 0,0,0,0,0,0,0,0,0,0
3400 DATA 0,0,0,0,0,0,0,0,0,0
3410 DATA 0,0,0,0,0,0,0,0,54
3420 DATA FC,0,0,40,E8,0,0,40,E8,0
3430 DATA 0,D4,B0,0,0,D4,B0,0,40,
E8
3440 DATA 40,0,40,E8,40,0,D4,B0,
C0,B0
3450 DATA D4,B0,D4,40,D4,B0,0,40,
E8,0
3460 DATA 0,C0,E8,0,0,D4,C0,0,0,D4
3470 DATA C0,0,0,E8,C0,0,0,E8,B0,0
3480 DATA 0,C0,B0,0,0,0,0,54,FC,0
3490 DATA 0,54,E8,0,0,40,C0,0,0,40
3500 DATA 0,0,0,0,FC,0,0,54,FC,0
3510 DATA 0,54,FC,0,0,0,C0,E8,E8,
40
3520 DATA D4,E8,E8,40,E8,D4,B0,C0,
E8,40
3530 DATA 40,D4,C0,B0,C0,E8,C0,E8,
FC,C0
3540 DATA C0,D4,C0,C0,C0,C0,C0,C0,
48,C0
3550 DATA B0,0,0,C0,B0,0,0,C0,0,0
3560 DATA 0,C0,0,0,0,B0,0,0,0,B0
3570 DATA 0,0,0,0,0,0,0,0,0,0
3580 DATA 0,0,0,0,C0,0,0,0,40,0
3590 DATA 0,0,0,0,0,0,0,0,0,0
3600 DATA 0,0,0,0,0,0,0,0,0,0
3610 DATA 0,0,0,C0,C0,C,B0,C0,C,C0
3620 DATA 0,40,C0,0,0,0,0,0,0,0
3630 DATA 0,0,0,0,0,0,0,0,0,0
3640 DATA 0,0,0,0,0,0,0,0,0,0
3650 DATA 0,0,0,0,0,0,0,0,0,0
3660 DATA 0,0,0,0,0,0,0,0,0,0
3670 DATA 0,0,0,0,0,0,0,0,0,0
3680 DATA 0,0,0,0,0,0,0,0,0,0
3690 DATA 0,0,0,0,0,0,AB,0,0,0
3700 DATA FC,0,0,0,54,0,0,0,54,0
3710 DATA 0,0,0,0,0,0,0,FC,0,54
3720 DATA AB,FC,AB,FC,AB,54,FC,FC,
0,FC
3730 DATA EB,FC,FC,FC,FC,D4,D4,FC,
D4,D4
3740 DATA AB,0,0,0,0,0,0,0,0,0
3750 DATA 0,0,0,54,AB,0,0,FC,0,0
3760 DATA 0,FC,0,0,0,AB,0,0,0,0
3770 DATA 0,0,0,0,0,0,0,0,0,0
3780 DATA 0,0,0,0,0,0,0,0,45,0
3790 DATA 0,0,45,0,0,0,45,0,0,0
```


```
3800 DATA 9F,0,0,0,CF,54,EB,FC,AB,
0
3810 DATA EB,EB,0,DE,FC,FC,AB,DE,
FC,DE
3820 DATA ED,DE,CF,CF,ED,CF,CF,6F,
CF,CD
3830 DATA CE,CF,CD,CF,CF,CF,9F,0,
0,0
3840 DATA 0,0,0,0,0,0,0,0,0,0
3850 DATA 0,0,0,0,0,0,0,BA,0,0
3860 DATA 0,BA,0,0,0,0,CF,0,0,0
3870 DATA 0,0,CF,0,0,0,CF,0,0,0
3880 DATA CE,0,0,0,45,0,0,0,45,0
3890 DATA 0,0,45,0,0,0,45,0,0,0
3900 DATA 0,CE,CF,CF,CF,9F,CF,6E,
CF,CF
3910 DATA CF,CF,CF,6F,CE,CD,CF,CF,
CF,CF
3920 DATA CD,CE,CF,6F,CF,9F,CE,CF,
DF,CF
3930 DATA CF,CF,EF,CF,0,0,0,CF,0,0
3940 DATA 0,EF,0,0,0,EF,0,0,0,AA
3950 DATA 0,0,0,AA,0,0,0,BA,0,0
3960 DATA 0,0,0,0,0,0,0,0,0,0
3970 DATA 0,0,0,0,0,0,0,0,0,0
3980 DATA 0,0,0,0,0,0,0,0,0,0
3990 DATA 0,0,0,0,0,0,CF,CF,DF
4000 DATA CF,45,CD,EF,BA,45,6F,CF,
0,45
4010 DATA CF,BA,0,45,BA,0,0,0,0,0
4020 DATA 0,0,0,0,0,0,0,0,0,0
4030 DATA 0,0,0,0,0,0,0,0,0,0
4040 DATA 0,0,0,0,0,0,0,0,0,0
4050 DATA 0,0,0,0,0,0,0,0,0,0
4060 DATA 0,0,0,0,0,0,0,0,0,0
4070 DATA 0,0,0,0,0,0,0,0,0,0
4080 DATA 0,0,0,0,0,0,0,0,0,0
4090 DATA 0,0,0,0,0,0,0,0,0,0
4100 DATA 0,0,0,0,0,0,0,0,0,0
4110 DATA 0,0,0,0,0,0,0,0,C0,C0
4120 DATA 0,C0,C0,C0,C0,C0,0,0,0,0
4130 DATA 0,0,0,0,0,0,0,0,0,0
4140 DATA 0,0,0,0,0,0,0,0,0,0
4150 DATA 0,0,0,0,0,0,0,0,0,0
4160 DATA 40,0,0,0,C0,0,0,0,C0,0
4170 DATA 0,40,C0,0,0,40,C0,0,0,C0
4180 DATA C0,0,0,C0,C0,0,0,C0,C0,
C0
4190 DATA C0,C0,C0,C0,C0,C0,C0,C0,
C0,C0
4200 DATA C0,C0,C0,C0,C0,C0,C0,C0,
C0,C0
4210 DATA C0,C0,C0,C0,C0,C0,C0,C0,
C0,C0
4220 DATA C0,B0,0,0,0,C0,0,0,0,C0
4230 DATA 0,0,0,C0,B0,0,0,C0,B0,0
4240 DATA 0,C0,C0,0,0,48,48,0,0,C0
4250 DATA 48,0,0,0,0,C0,C0,0,0,C0
4260 DATA C0,0,0,C0,C0,0,0,40,C0,0
4270 DATA 0,40,C0,0,0,0,C0,0,0,0
4280 DATA C0,0,0,0,40,C0,48,84,C0,
C0
4290 DATA C0,C0,C0,C0,C0,84,84,C0,
48
4300 DATA C0,C0,C0,C0,48,C0,84,84,
C0,C0
```

VARIABLES PRINCIPALES

h, i, j, m, n, p: Contadores de bucle.
a: Lectur de DATA y diversos usos.
a$(): Codificación de la tirada obtenida.
ex: Dirección de entrada al código máquina.
col: Dirección donde se almacena la columna del dibujo.
fil: Dirección donde se almacena la fila del dibujo.
car: Dirección donde se almacena el dibujo.
sc: Dinero disponible.
bn: Número de bonus.
pr: Premio obtenido.
a$, il$, i2$: Lectura del teclado.
k, n: Fila y columna para encender luces.
b(): Contador de bonus de distintas frutas.
b$: String con números 1-6.

Las variables utilizadas en la fase de «tirada» son: b1, b2, d, r, i(), m(), a(), al(), a2(), n().

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 casisette mensual, solicítanalo.

```
4310 DATA C0,C0,84,C0,4B,84,4B,4B,
4B,0
4320 DATA 0,C0,4B,0,0,84,C0,0,0,84
4330 DATA B0,0,0,4B,B0,0,0,4B,0,0
4340 DATA 0,C0,0,0,0,B0,0,0,0,0
4350 DATA 0,0,0,0,0,0,0,0,0,0
4360 DATA 0,0,0,0,0,0,0,0,0,0
4370 DATA 0,0,0,0,0,0,0,0,0,0
4380 DATA 0,C0,84,4B,C0,0,C0,C0,0,
0
4390 DATA 0,0,0,0,0,0,0,0,0,0
4400 DATA 0,0,0,0,0,0,0,0,0,0
4410 DATA 0,0,0,0,0,0,0,0,0,0
4420 DATA 0,0,0,0,0,0,0,0,0,0
4430 DATA 0,0,0,0,0,0,0,0,0,0
4440 DATA 0,0,0,0,0,0,0,0,0,0
4450 DATA 0,0,0,0,0,0,0,0,0,0
4460 DATA 0,0,0,0,0,0,0,0,0,0
4470 DATA 0,0,0,0,0,0,0,0,0,0
4480 DATA 0,0,0,0,0,0,0,0,0,0
4490 DATA 0,0,0,0,0,0,0,0,0,0
4500 DATA 0,0,0,0,0,0,0,0,0,0
4510 DATA 0,0,0,0,0,0,0,0,0,0
4520 DATA 0,0,0,0,0,0,0,0,0,0
4530 DATA 0,0,0,0,0,0,0,0,0,0
4540 DATA 0,0,0,0,0,0,0,0,0,0
4550 DATA 0,0,0,0,0,0,0,0,0,0
4560 DATA 0,0,0,0,0,0,0,0,0,0
4570 DATA 0,0,54,0,0,51,0,0,0,51
4580 DATA 0,0,0,A2,0,0,0,A2,0,0
4590 DATA 0,A2,0,54,FC,F6,AB,FC,
FC,A6
4600 DATA FC,FC,AC,5C,FC,0,0,0,0,0
4610 DATA 0,0,0,0,0,0,0,0,0,0
4620 DATA 0,0,0,0,0,0,0,0,0,0
4630 DATA 0,0,0,AB,0,0,0,0,0,0
4640 DATA 54,0,0,0,FC,0,0,0,FC,0
4650 DATA 0,0,FC,0,0,0,FC,0,0,0
4660 DATA FC,0,0,0,FC,0,0,0,54,FC
4670 DATA FC,FC,FC,EB,FC,FC,FC,C0,
FC,FC
4680 DATA FC,FC,FC,FC,FC,FC,FC,FC,
BD,FC
4690 DATA FC,FC,BD,FC,FC,FC,BD,FC,
FC,BD
4700 DATA 3F,AB,0,0,0,FC,0,0,0,FC
4710 DATA 0,0,0,FC,0,0,0,FC,0,0
4720 DATA 0,FC,0,0,0,FC,0,0,0,AB
4730 DATA 0,0,0,0,0,0,54,0,0,0
4740 DATA 0,0,0,0,0,0,0,0,0,0
4750 DATA 0,0,0,0,0,0,0,0,0,0
4760 DATA 0,0,0,0,0,FC,FC,BD,7E,FC
4770 DATA FC,FC,FC,54,FC,FC,AB,0,
0,0
4780 DATA 0,0,0,0,0,0,0,0,0,0
4790 DATA 0,0,0,0,0,0,0,AB,0,0
4800 DATA 0,0,0,0,0,0,0,0,0,0
4810 DATA 0,0,0,0,0,0,0,0,0,0
4820 DATA 0,0,0,0,0,0,0,0,0,0
4830 DATA 0,0,0,0,0,0,0,0,0,0
4840 DATA 0,0,0,0,0,C0,C0,C0,C0,C0
4850 DATA C0,C0,C0,FC,FC,FC,FC,FC,
FC,FC
4860 DATA FC,0,0,0,0,0,0,0,0,0
4870 DATA 0,0,0,0,0,0,0,C0,C0,C0
4880 DATA C0,C0,C0,C0,C0,FC,FC,FC,
FC,FC
4890 DATA FC,FC,FC,0,0,0,0,0,0,0
4900 DATA 0,0,0,0,0,0,0,0,0,C0
4910 DATA C0,C0,C0,C0,C0,C0,C0,FC,
FC,FC
4920 DATA FC,FC,FC,FC,FC,CF,CF,CF,
CF,CF
4930 DATA CF,CF,CF,CF,3F,3F,6F,CF,
3F,3F
4940 DATA 3F,CF,6F,CF,9F,CF,6F,CF,
9F,CF
4950 DATA 6F,CF,9F,CF,3F,3F,6F,CF,
CF,CF
4960 DATA CF,CF,CF,CF,CF,CF,3F,3F,
CF,9F
4970 DATA 3F,3F,6F,9F,CF,CF,6F,9F,
CF,CF
4980 DATA 6F,9F,CF,CF,6F,9F,3F,3F,
6F,CF
4990 DATA CF,CF,CF,CF,CF,CF,3F,
3F,6F
5000 DATA CF,3F,3F,3F,CF,6F,CF,9F,
CF,6F
5010 DATA CF,9F,CF,6F,CF,9F,CF,3F,
3F,6F
5020 DATA CF,CF,3F,3F,6F,CF,6F,CF,
9F,CF
5030 DATA 6F,CF,9F,CF,6F,CF,9F,CF,
3F,3F
5040 DATA 3F,CF,3F,3F,6F,CF,CF,CF,
CF,CF
5050 DATA CF,CF,CF,9F,3F,3F,6F,9F,
CF,CF
5060 DATA 6F,9F,CF,CF,6F,9F,CF,CF,
6F,9F
5070 DATA CF,CF,6F,9F,CF,CF,6F,CF,
CF,CF
5080 DATA CF,CF,CF,CF,CF,3F,3F,6F,
CF,6F
5090 DATA CF,9F,CF,6F,CF,9F,CF,6F,
CF,9F
5100 DATA CF,6F,CF,9F,CF,6F,CF,9F,
CF,CF
5110 DATA CF,CF,CF,CF,CF,CF,CF,FC,
FC,FC
5120 DATA FC,FC,FC,FC,FC,C0,C0,C0,
C0,C0
5130 DATA C0,C0,C0,0,0,0,0,0,0,0
5140 DATA 0,0,0,0,0,0,0,0,0,FC
5150 DATA FC,FC,FC,FC,FC,FC,FC,C0,
C0,C0
5160 DATA C0,C0,C0,C0,C0,0,0,0,0,0
5170 DATA 0,0,0,0,0,0,0,0,0,0
5180 DATA 0,FC,FC,FC,FC,FC,FC,FC,
FC,C0
5190 DATA C0,C0,C0,C0,C0,C0,C0,0,
0,0
5200 DATA 0,0,0,0,0,0,0,0,0,0
5210 DATA 0,0,0,26,F,2E,E,3E,18,6
5220 DATA 4,F5,CD,1A,BC,F1,11,40,
93,4B
5230 DATA CB,21,CB,21,CB,21,C5,3D,
28,7
5240 DATA 41,13,10,FD,C1,18,F5,C1,
E,8
5250 DATA C5,F1,F5,47,E5,1A,77,13,
23,10
5260 DATA FA,E1,6,8,24,10,FD,D,20,
ED
5270 DATA F1,C9,0,0,0,0,0,0,0,0
```

HERBERT

Un niño travieso, perdido en unos grandes almacenes, vaga sin cesar por las distintas plantas y secciones comerciales; deportes, juguetes, sonido, librería, muebles, ropa de caballero, restaurante. ¿Conseguirá encontrar a sus padres?

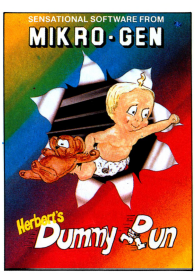
Probablemente el nombre de Herbert no tenga un significado especial para nadie, pero si empezamos a hablar de Wally y su familia, tal vez la cosa comience a tener algo de sentido.

Wally es la estrella de Micro Gen, compañía inglesa autora de todas sus aventuras.

El debut de Wally en la pantalla de ordenador tuvo lugar en Automanía, su primera aventura, en la cual nuestro amigo es un mecánico de automóviles, desarrollándose la acción en su garaje. Wally debe recoger las distintas piezas de los coches y llevarlas al foso donde montará el automóvil.

Sus enemigos son ruedas locas, cafeteras, bujías, etc.

La segunda aventura es Pyjamarama, la pesadilla de Wally. El espíritu de nuestro héroe sufre terribles alucinaciones y deambula por su hogar, que se transforma en un nido de trampas, cada habitación tiene sus moradores y terribles sorpresas.



Compatible CPC 464/CPC 664/CPC 6128

Esta segunda aparición de Wally, es mucho más que un arcade como era Automanía, en ella debemos utilizar los objetos que encontramos para facilitar nuestro camino por las distintas estancias.

La trama es bastante complicada y completar la aventura requiere largas horas de juego. En su pesadilla el protagonista incluso debe viajar a la Luna, caminar por cornisas y matar marcianos. Su objetivo, poder hacer que su espíritu se reúna con su cuerpo que mientras tanto duerme plácidamente.

Por fin aparece la familia en su tercera aventura, Evryone's a Wally, nuestro mecánico ahora es el jefe de una banda de ladrones de cajas fuertes.

Los componentes de la banda:

Wally, el cerebro de la banda.

Wilma, mujer de Wally.

Tom, el punky de la banda experto en mecánica.

Dick, un fontanero capaz de desatascar lo que sea.

Harry, el hippie diplomado en electrónica.

Herbert, hijo de Wally y Wilma.

El escenario en esta ocasión es la ciudad natal de Wally, con sus sitios típicos; zoológico, parques, carnicería, banco, muelle e incluso un pub el Blue Lyon.

El objetivo: abrir la caja fuerte del banco.

Un juego con personajes interactivos, que siguiendo la estela del Pyjamarama tiene una trama muy difícil de desvelar.

Después de un largo viaje por la historia de Wally y sus aventuras, por fin llegamos a Herbert.

El bebé que en Evryone's a Wally andaba a gatas por las calles de Wallytown, con el paso del tiempo ha aprendido a andar y ahora es un niño de lo más travieso.

Una tarde nuestro amiguito salió de compras con sus papás, como todas las familias cuando salen de compras; van a unos grandes almacenes.

De todos es conocida la fama de los almacenes ingleses, con un número sin fin de secciones y plantas.

En una de estas expediciones familiares en busca de ropa y artículos para el hogar, nuestro amiguito ensimismado por las maravillas del departamento de juguetes se entretiene demasiado y pierde el contacto con sus padres.

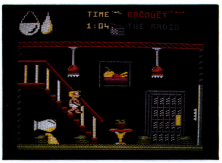
Wilma y Wally no pudiendo encontrar al muchacho deciden esperarle en el departamento de niños y objetos extraviados.

Todo iría bien si nos encontrásemos en una situación normal, pero la suerte no acompaña a la familia Week.

Una amenaza de bomba, ha sido recibida en la centralita de los almacenes y éstos han sido evacuados urgentemente.

Herbert, distraído entre los juguetes pasó desapercibido a los empleados que huían despavoridos, cumpliendo a rajatabla aquello de **«sálvese quien pueda»**, traicionando lo de **«el capitán es el último que abandona el barco»**.

Mr. Joystick



Los padres de Herbert deciden esperar a su hijo hasta el fin y se quedan en el departamento de niños perdidos en espera de su querido retoño.

Es exactamente la una del mediodía, 1 PM en el Reino Unido y la bomba tiene programada la explosión a las cinco y media, nuestro héroe debe llegar hasta sus padres antes de la hora fatídica, de lo contrario todo estará perdido.

Para resolver la aventura, debemos superar una gran variedad de pantallas arcade, en las que los objetos recogidos en otros departamentos, son imprescindibles para poder afrontarlas; no podremos jugar al tenis sin raqueta, ni demoler edificios sin bombas.

Todas las fases arcades representan emocionantes y adictivos juegos por sí mismos, sin tener en cuenta el hecho de que forman parte de una gran aventura.

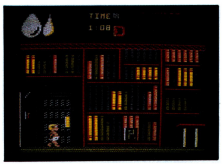
Movernos de planta en planta es posible por medio de los ascensores, cabiendo también la posibilidad de escalar fenomenales maromas.

Las lágrimas, fuente de vida de Herbert, deben ser incrementadas con golosinas, que buscaremos en los distintos departamentos, el dinero también juega su papel y permite obtener otros objetos.

Bombas, interruptores, chocolate, miel, cuerdas, monedas y demás objetos tienen utilidad en la aventura y su uso debe descubrirse a base de ingenio y bastantes horas de aventura.

Es de destacar que a pesar de ser un juego inglés, todos los mensajes que aparecen en pantalla vienen traducidos al castellano, facilitando su reconocimiento y utilización.

Un programa que proporcionará largas horas de diversión, a todos los amantes de los juegos que salen de la esfera del arcade para transformarse en grandes aventuras en las que el ingenio es la clave del éxito.



MENU PARA DISCO

COMPATIBLE
CPC 464
CPC 664

La diferencia fundamental entre un sistema basado en cassette y uno basado en disco es, por supuesto, la velocidad en general para cualquier tarea, y en particular para cargar y salvar programas.

En esto último, mientras el disco tarda segundos, el cassette consume minutos, durante los cuales nos mordemos las uñas de mal disimulada impaciencia *(al menos, yo lo hago)*.

No estaría de más tener a mano un programa que, al ejecutarlo, presentara en pantalla el contenido del disco, permitiéndonos escoger el programa deseado de una manera sencilla y ejecutarlo. Si esto fuera posible, la diferencia entre cassette y disco quedaría aún más patente.

El programa que presentamos aquí lo hace.

No obstante, para que una aplicación así resulte eficiente, debe ser implementada preveiendo frecuentes cambios en el contenido del disco; en efecto, si, por ejemplo, colocáramos en sentencias DATA dicho contenido, nos veríamos obligados a modificar el programa menú cada vez que alteráramos el disco.

En definitiva, necesitamos un programa «inteligente» que mire en el disco, averigüe los programas que hay en él y nos los ponga en pantalla.

Nuestra estrategia será hacer que el **Amstrad** ejecute la misma secuencia que haríamos nosotros, es decir, un catálogo del disco, y luego, simplemente, elegir el programa y cargarlo o ejecutarlo.

El Basic del CPC464 no posee un comando para leer directamente de pantalla *(el 664 y el 128 sí)*, por lo cual, hemos implementado una rutina en máquina para que el programa sirva para los tres ordenadores. Además, esta rutina es muy especial, y la comentaremos en detalle más adelante.

RUTINA READCHAR

Propósito: leer un caracter de la pantalla en la posición especificada por el registro HL y almacenar el caracter leído en la variable Basic **«char»**.
Llamadas desde Basic: líneas 490,530, 570.

Dirección de carga: &8001.
Dirección de arranque: &8001.
Longitud en bytes: 20.
Registros utilizados: IX, HL, AF, BC, DE.
Registros modificados: IX, HL, AF.

LD L, (IX+O)	Obten el valor de la variable «y» *(fila)*.
LD, H, (IX+2)	Obten el valor de la variable «x» *(columna)*.
CALL &BB75	Posiciona el cursor de texto.
CALL &BB60	Lee el caracter de la posición del cursor y almacénalo en el acumulador.
LD L, (IX+4)	Coge la dirección.
LD H, (IX+5)	De la variable char.
LD (HL), A	Almacena el caracter en dicha variable.
RET	Vuelta al Basic.

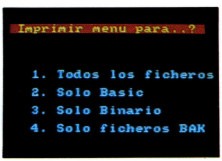
Por tanto, quedamos en que el programa hace un CAT, lee los programas de la pantalla mediante la rutina en máquina y los almacena en una matriz. Todo este proceso es invisible, porque el color del papel y la tinta es el mismo mientras dura.

Una vez hecho esto, el programa presenta en pantalla un submenú que nos permite elegir entre distintos tipos de ficheros.

El método de elección es muy simple: nos desplazamos por la pantalla con las teclas del cursor hasta llegar al programa elegido, y pulsamos ENTER. **¡Y ya está!**

Ahora la rutina en máquina: se pueden ver en el listado ensamblador que hay dos llamadas a rutinas del firmware.

La primera, dirección &BB75, posiciona el cursor de texto en las posiciones especificadas por el registro doble HL.



La segunda, dirección &BB60, es la que lee el caracter en la posición del cursor de texto.

Hasta aquí todo está claro, pero si observamos la línea 490 del programa menú, veremos que se hace una llamada a esta rutina *(call readchar)* y que se le suministran tres parámetros:

@char
x
y

El primero es la dirección de una variable donde queremos que la rutina en máquina readchar deje el caracter, y los otros dos son las posiciones donde readchar colocará el cursor de texto y leerá el caracter.

La clave está en los mnemónicos que manipulan el registro doble IX, y en que el comando CALL de Basic, cuando se le suministran parámetros además de la propia dirección de arranque, deja el registro IX apuntando al ULTIMO DE LOS PARAMETROS.

Por eso, *(IX+O)* corresponde al valor de la variable «y», y como ocupa dos bytes, la variable «x» estará posicionada en (IX+2).

El mismo razonamiento nos dice que la dirección de la variable char estará en *(IX+4)*, en cuyo contenido metemos el valor del acumulador, el caracter leído, mediante la simple maniobra de cargar HL con la dirección y cargar el contenido de HL con el acumulador.

Por tanto, ya sabemos:

1. Mandar parámetros a una rutina en máquina desde Basic.

ESTRUCTURA DEL PROGRAMA

LINEAS	FUNCION
180-290	Inicializa variables.
300-440	Obtiene el tipo de fichero. Se pueden añadir otros si se desea.
460-590	Introduce los nombre-fichero en una matriz.
640-750	Imprime el menú.
760-880	Selecciona programa.

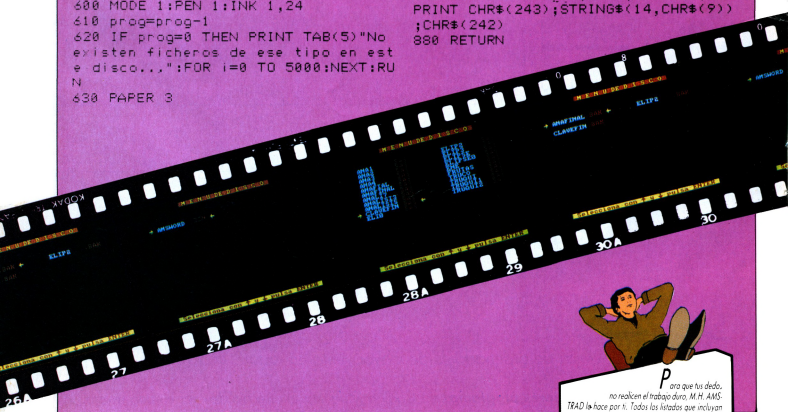
VARIABLES

name$ (64)	Nombre de los ficheros.
coord (64,1)	Posiciones de impresión.
prog	Número de programas.
type$	Tipo de fichero.
char	Caracter leído en la pantalla por la rutina **«readchar»**.

2. Devolver los resultados de la rutina en una variable Basic y luego manipularla.

```
10 REM Menu para disco
20 REM por R.A.Waddilove
30 REM (c) Amstrad semanal
40 REM ------------------------------
-
50 GOSUB 180:REM Inicializacion
60 GOSUB 460:REM obtener nombres
70 IF INSTR(file$,".BIN")=0 THEN RU
N file$
80 CLS:INPUT "HIMEM ";a
90 IF a=&8000 THEN MEMORY &7FFF ELS
E IF a THEN MEMORY a-1
100 PRINT "Load o Run (L/R) ?"
110 k$=""
120 WHILE k$<>"L" AND k$<>"R"
130 k$=UPPER$(INKEY$)
140 WEND
150 IF k$="L" THEN LOAD file$ ELSE
RUN file$
160 END
170 REM ------------------------------
180 REM Inicializacion
190 MODE 1:INK 0,0:BORDER 0
200 DEFINT b-z
220 DIM name$(64),coord(64,1)
230 FOR i=1 TO 20
240 READ a$
250 POKE &8000+i,VAL("&"+a$)
260 NEXT
270 DATA DD,6E,00,DD,66,02,CD,75,BB
,CD,60,BB,DD,6E,04,DD,66,05,77,C9
280 readchar=&8001:char=0
290 prog=1:type$=""
300 PAPER 3:LOCATE 9,5
310 PRINT " Imprimir menu para..? "
320 PAPER 0:PEN 2
330 LOCATE 12,10:PRINT "1. Todos lo
s ficheros"
```

```
340 LOCATE 12,12:PRINT "2. Solo Bas
ic"
350 LOCATE 12,14:PRINT "3. Solo Bin
ario"
360 LOCATE 12,16:PRINT "4. Solo fic
heros BAK"
370 WHILE type$<"1" OR type$>"4"
380 type$=INKEY$
390 WEND
400 ON VAL(type$) GOTO 410,420,430,
440
410 type$="":RETURN
420 type$=".BAS":RETURN
430 type$=".BIN":RETURN
440 type$=".BAK":RETURN
450 REM ------------------------
-
460 REM Obtener los nombres de fich
ero
470 MODE 2:INK 1,0:PEN 1:CAT
480 FOR x=1 TO 61 STEP 20
490 y=4:CALL readchar,$char,x,y
500 WHILE char<>32
510 name$(prog)=""
520 FOR i=0 TO 11
530 CALL readchar,$char,x+i,y
540 name$(prog)=name$(prog)+CHR$(ch
ar)
550 NEXT
560 IF INSTR(name$(prog),type$) THE
N prog=prog+1
570 y=y+1:CALL readchar,$char,x,y
580 WEND
590 NEXT
600 MODE 1:PEN 1:INK 1,24
610 prog=prog-1
620 IF prog=0 THEN PRINT TAB(5)"No
existen ficheros de ese tipo en est
e disco...":FOR i=0 TO 5000:NEXT:RU
N
630 PAPER 3
640 LOCATE 10,1:PRINT " M E N U DE
D I S C O "
650 PAPER 0
660 y=3:num=(prog%2)+(1 AND prog)
670 FOR i=1 TO prog
680 y=y+1-1*(prog(21):IF i=num+1 TH
EN y=4-1*(prog(21)
690 coord(i,0)=4+(20 AND i>num):coo
rd(i,1)=y
700 LOCATE coord(i,0),coord(i,1)
710 PEN 2:PRINT LEFT$(name$(i),INST
R(name$(i),".")-1);
720 PEN 3:PRINT MID$(name$(i),INSTR
(name$(i),"."))
730 NEXT
740 LOCATE 4,25:PEN 3:PAPER 1
750 PRINT " Selecciona con ";CHR$(2
40);" y ";CHR$(241);" pulsa ENTER "
;
760 i=1:PEN 1:PAPER 0
770 GOSUB 870
780 WHILE INKEY$<>CHR$(13)
790 IF INKEY(0)>-1 THEN GOSUB 850:i
=i+(i>1):GOSUB 870
800 IF INKEY(2)>-1 THEN GOSUB 850:i
=i-(i<prog):GOSUB 870
810 FOR j=1 TO 200:NEXT
820 WEND
830 file$=name$(i)
840 RETURN
850 LOCATE coord(i,0)-2,coord(i,1):
PRINT " ";STRING$(14,CHR$(9));" "
860 RETURN
870 LOCATE coord(i,0)-2,coord(i,1):
PRINT CHR$(243);STRING$(14,CHR$(9))
;CHR$(242)
880 RETURN
```



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

QUINIELAS

Análisis

Existen gran cantidad de métodos, exclusivamente basados en la suerte para rellenar una quiniela: dados, bolígrafos mágicos, matrículas de coches, hombre 1, mujer X, niño 2, etc.

Otro método interesante, sobre todo para los poseedores de un ordenador, sería obtener los resultados con él.

Análisis en esta ocasión se adentra en el mundo de los números aleatorios con un programa que solamente necesita la ayuda de la suerte para acertar 12, 13, o incluso 14 resultados.

10-20 Mensajes REM que contienen el título del programa y los que lo separan del resto de las líneas.

30-50 Ciclo WHILE... WEND que controla el número de apuestas a introducir.

Si éste es menor que 1 o mayor que 8, el programa volverá a preguntarnos el número de apuestas, si en cambio aquél está comprendido entre 1 y 8 el ciclo pasa la ejecución a la próxima línea.

60 Limpia de la pantalla todos los mensajes anteriores para proceder a la representación de los distintos pronósticos.

70-100 Constituye el ciclo que se encarga de poner el número de los distintos partidos.

80 El comando LOCATE se ocupa de situar el cursor en la columna 10, fila 5, incrementando esta posición en uno cada vez que rota el ciclo, para que los números queden en la misma columna.

90 Una vez situado el cursor PRINT representa el número de partido correspondiente.

100 Hace que el proceso se repita 14 veces, una por partido.

110-200 Ciclo principal que se repite tantas veces como el número de apuestas que hemos seleccionado en la línea 40.

120-190 Ciclo anidado en el anterior, que realiza todo el proceso de cálculo de los distintos signos de cada apuesta y su representación en pantalla.

130 Sitúa el cursor en la columna 13, fila 5. Cada vez que rota este segundo ciclo, la fila se incrementa en una unidad consiguiendo de esta forma que los signos queden encolumnados. Cuando este ciclo se ha completado con los 14 signos, el ciclo principal entra en acción, incrementándose en uno el valor de la posición, lo que provoca que en la próxima columna de signos la posición se encuentren desplazadas 2 columnas. Formando de esta manera en cada pasada del ciclo principal las distintas columnas de apuestas.

140 Calcula un valor aleatorio comprendido entre 1 y 13.

150 Comprueba si la variable VALOR, calculada en la línea anterior, es mayor que 0 y menor que 8, asignándole en este caso el signo 1 a la variable SIGNOS$.

160 Chequea si el valor está comprendido entre 8 y 11, asignándole el signo XX, en caso afirmativo.

170 Realiza la misma operación con los límites 11 y 14, asignándole el valor 2, en caso de estar comprendido entre ellos.

180 Representa en pantalla el valor contenido en SIGNOS$, éste aparece en la posición marcada por el LOCATE de la línea **130.**

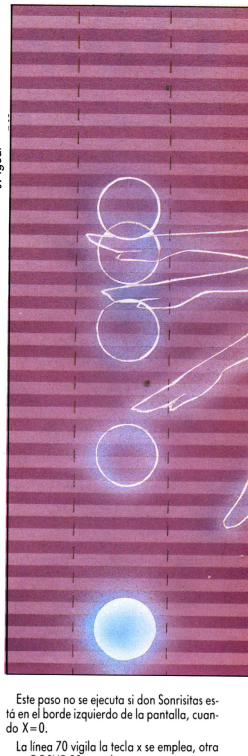
190 NEXT que cierra el ciclo secundario, que calcula y representa los signos de los partidos.

200 Es el NEXT que concluye el ciclo principal, responsable de situación de las distintas columnas de signos.

ANIMACION PLUS

(Tercera parte de la serie Animación en Basic para **Amstrad**)

J. Igual



Una vez conocidas las técnicas básicas de la animación, vamos a entrar en los pequeños detalles, que permiten reproducir los movimientos reales.
La inclusión de choques, rebotes, movimiento sin fin, etc., ayudará a mejorar notablemente nuestros programas, infundiéndoles la vida que necesitaban.

En este artículo, el último de la serie de animación, avanzaremos un poco más por estos interesantes métodos que ayudan a mejorar la animación.

Como ya sabemos, el **Amstrad** tiene dos tipos de cursores, uno para los textos y otro para los gráficos. Normalmente, el comando PRINT realiza los textos con el cursor de textos *(la gran mancha cuadrada)*. Pero utilizando el comando TAG podemos dirigir el texto de la instrucción PRINT en la posición que tiene el cursor de gráficos. El comando TAGOFF invierte el proceso.

El cursor de gráficos, que es invisible, puede colocarse en cualquier punto *(pixel)* de la pantalla, a diferencia del cursor de textos que sólo puede moverse en celdas definidas de caracter.

De manera que una vez se ejecute TAG el texto también puede imprimirse desde cualquier punto *(pixel)*. Además, podemos seleccionar la manera de situar el texto en la pantalla, puede hacerse de tres modos distintos: XOR, AND y OR.

Estos operadores lógicos, como ya sabemos, se utilizan para determinar el color del pixel que va a aparecer. Para nuestro propósito sólo nos interesa XOR.

Vamos a considerar el caso en el que el texto esté impreso con el cursor de gráficos, XOR se utiliza para calcular los colores resultantes y estamos en Modo 1.

Imprimir y borrar caracteres

Si tenemos un fondo azul *(color 0)* y escribimos un caracter amarillo *(color 1)* el resultado será amarillo porque 0 XOR 1 = 1 *(amarillo)*.

Pero si ahora queremos escribir este mismo caracter —con el mismo color— encima del caracter anterior, éste desaparecerá, porque 1 *(amarillo)* XOR 1 *(amarillo)* = 0 *(azul)*. Así que a cambio de sacar en pantalla otro caracter borramos el anterior.

Utilizar la opción XOR nos permite sacar caracteres en pantalla y borrarlos, repitiendo la misma operación. Ahora que esto ya nos suena a conocido, vamos a utilizarlo.

El Programa 1 hace exactamente esto. Usaremos las teclas Z y X para mover a don Sonrisitas a izquierda y derecha.

La línea 30 selecciona el modo XOR para escribir.

La línea 40 inicializa la coordenada X en 0.

La línea 50 saca en pantalla a Don Sonrisitas usando la subrutina formada por las líneas 90-130.

La línea 60 comprueba si se está pulsando la tecla Z. Si es así, se suprime el caracter (con el primer GOSUB 90) y se desplaza una celda de caracter a la izquierda (X= X—16). Ahora don Sonrisitas se ha desplazado a su nueva posición *(con el segundo GOSUB 90)*.

Este paso no se ejecuta si don Sonrisitas está en el borde izquierdo de la pantalla, cuando X=0.

La línea 70 vigila la tecla x se emplea, otra vez, GOSUB 90 para borrar y sacar en pantalla a don Sonrisitas. El caracter se mueve una celda de caracter a la derecha sumando 16 a X. Tampoco se mueve el caracter si está al borde derecho de la pantalla en este caso.

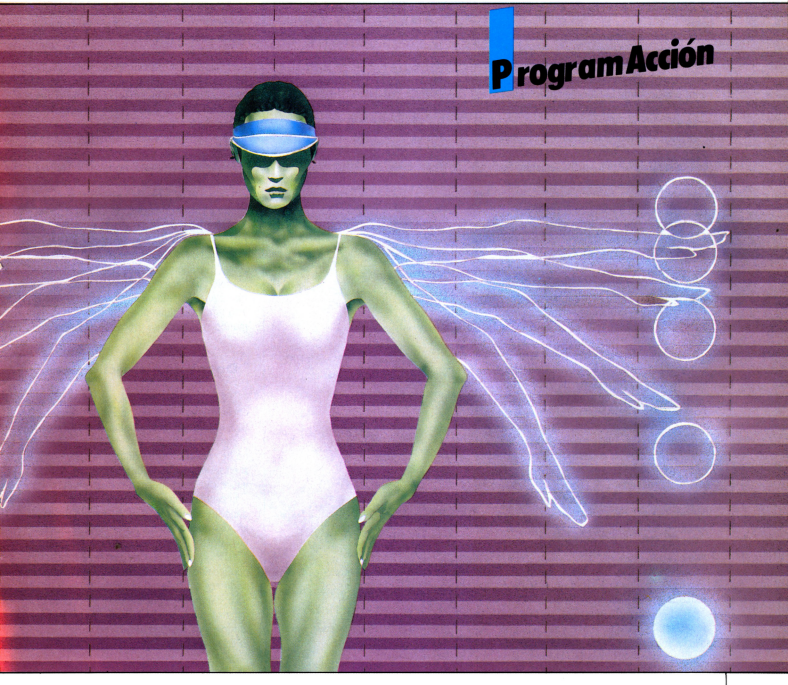
La línea 80 **«salta»** hacia atrás, a la 60.

La línea 90 mueve *(MOVE)* el cursor de gráficos a la posición actual de don Sonrisitas.

La línea 100 hace que todos los textos vayan a la posición del cursor gráfico.

La línea 110 imprime el caracter don Sonrisitas en el cursor gráfico.

La línea 120 devuelve la salida del texto al cursor de textos.



Las líneas 90-130 se utilizan para sacar en pantalla y borrar caracteres. Lo que ahorra repetir líneas similares en los programas.

Inconvenientes del XOR

Una demostración de las desventajas de XOR se añade en la siguiente línea del programa:

25 LOCATE 20,16: PEN 3: PRINT **«XOR es ingenioso».**

Ahora veamos qué ocurre cuando movemos a don Sonrisitas por el texto. El cyan *(color 2)* aparece donde don Sonrisitas y el texto se solapan.

Pero **¿de dónde viene el cyan?** ¡No hemos escrito nada en cyan! En efecto, el cyan aparece porque 1 don Sonrisitas amarillo) XOR 3 *(el texto es rojo)=2 Cyan).*

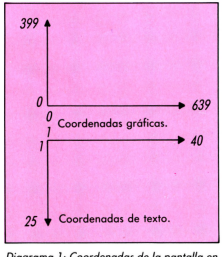


Diagrama 1: Coordenadas de la pantalla en Mode 1.

Si esto le parece chino no se preocupe, el CPC 464 hace el cambio de color. Puede saber más sobre color con BITS and BYTES de mayo, pero ¡no lo use sin entenderlo!

Cambiemos de tema, algunos movimientos que tienen los objetos están restringidos. En el **Amstrad,** estas restricciones son los movimientos en los bordes de la pantalla.

El mejor modo de parar un objeto en movimiento, yendo a un **«prohibido el paso»,** es guardar un registro de esa posición. Sabiendo donde está un objeto se nos permite controlar que cualquier futuro movimiento no se realice fuera de los límites definidos.

Estos límites no tienen por qué ser los bordes de la pantalla necesariamente. Pueden ser cualquier parte de la pantalla que elijamos.

Movimiento y rebote

El Programa II hace botar una bola en sentido diagonal alrededor de la pantalla en Modo 1. Las variables X e Y contienen las coordenadas x e y de la bola. Estas corresponden a las posiciones de las coordenadas de texto en la pantalla. Vamos a ver como trabaja.

La línea 30 inicializa X e Y.

La línea 40 selecciona un número aleatorio entre 1 y 4 y salta a la línea 50 si el número es 1, a la 60 si es 2, y así sucesivamente. Esta selección da a la bola una dirección aleatoria.

Las líneas 50 a 80 definen el cambio relativo en la dirección de la bola entre las cuatro direcciones diferentes. RX es el cambio relativo de la coordenada X, RY es el cambio relativo de la coordenada Y.

Por ejemplo, la línea 70 indica que el cambio en X será 1 *(RX=1 es la manera de mover la bola a la derecha en una celda de carácter)* y el cambio en Y será —1 *(RY=—1 es la manera de subir la bola una fila de carácter)*.

Mover un objeto a la derecha y arriba es lo mismo que mover la bola al noreste donde el norte es la parte superior de la pantalla.

La línea 90 vigila que la coordenada X permanezca dentro de la pantalla al mover la bola en la nueva dirección. Si la bola saliera de la pantalla se selecciona una nueva dirección—el programa salta a la línea 40.

La línea 100 hace lo mismo que la 90 pero con la coordenada Y.

La línea 110 coloca el cursor de texto en la posición anterior de la bola y la borra haciendo PRINT CHR$(32) un espacio.

La línea 120 calcula la nueva posición de la bola.

La línea 130 dibuja la bola.

La línea 150 continúa el movimiento en la misma dirección saltando a la línea 90.

Por el momento la bola sólo cambiará de dirección si choca contra el borde de la pantalla. Si añadimos la línea siguiente la bola tendrá también la oportunidad de cambiar de dirección siempre que quiera. Veamos si podemos hacerlo funcionar.

```
140 IF INT (RND(1)*20)=10
    THEN 40
```

Choques con cambio de banda

El Programa III controla los choques con los límites. En este caso se traslada al lado opuesto de la pantalla cuando encuentra un límite en lugar de cambiar de dirección.

Las líneas 100 a 130 son las responsables de controlar si la bola está fuera de los límites. Si es así, las coordenadas X e Y se ajustan convenientemente.

Por ejemplo, si una bola está en la coordenada X 40— a mano derecha en el borde de la pantalla y movemos al noreste, la nueva coordenada X será 41.

Pero esto está fuera de la pantalla— el Modo 1 sólo tiene 40 caracteres por línea, así que la línea 110 hará comenzar a X en 1— el borde izquierdo de la pantalla. Lo mismo ocurre si la coordenada Y sale fuera de pantalla.

Si se siente capaz, puede intentar hacer que la bola se mueva vertical y horizontalmente aunque no al mismo tiempo.

Finalmente vamos al Programa IV. Teclee y pulse RUN. Veremos a don Sonrisitas pasearse cruzando la pantalla hasta que se choca contra un obstáculo colocado aleatoriamente.

¿Qué hay de ingenioso en esto? La respuesta es detectar la colisión.

Lo que tenemos que hacer es probar si el carácter puede moverse a la siguiente celda de caracter. Se hace mirando a la esquina superior izquierda de la celda de caracter a ver si está señalado algún pixel. Se hace con el comando TEST como su propio nombre indica.

Detección de choques

TEST x, y devuelve el color del pixel que se encuentra en las coordenadas de gráficos x, y. Si el color del pixel es el mismo del fondo, el caracter puede moverse sin peligro, porque no ha chocado contra nada. De otro modo, si encuentra un color diferente al del fondo, indica que ha tropezado con un objeto.

El problema que presenta es que las coordenadas x e y para gráficos no son las mismas coordenadas que para textos. Echemos un vistazo al Diagrama 1.

Así es que necesitamos convertir las coordenadas de textos a las de gráficos para que podamos examinar el color del pixel.

La instrucción siguiente devolverá el color del pixel superior izquierdo de un caracter situado en las coordenadas de texto X, Y— sólo en el Modo 1:

```
colpix=TEST
((X—1)*16,(((26—Y)*16)—1))
```

Observe que solamente se examina el pixel superior izquierdo del caracter. Si miramos la línea 80 veremos la ecuación.

Si le extraña que *(X—1)*16* sea X*16 en el programa recuerde que estamos chequeando un carácter antes en la dirección del eje X. Así que *(X—1+1)*16* se convirtió en X*16.

La línea 40 sitúa un obstáculo aleatorio en la misma línea como **«guardia de la porra»**.

Actualmente debe ser capaz de entender el resto del programa.

Y este es el final de la serie. Esperamos haberle dado más ideas para que realice su propia animación.

Como ve, la animación es bastante fácil y puede realizarse usando unos comandos Basic muy sencillos. Ahora le toca a usted escribir sus propios programas de animación. Esperamos con ilusión ver alguno.

PROGRAMAS

```
10 REM PROGRAMA I
20 MODE 1
30 PRINT CHR$(23);CHR$(1)
40 X=0
50 GOSUB 90
60 IF INKEY(71)=0 AND X>0 THEN GOSU
B 90:X=X-1:GOSUB 90
70 IF INKEY(63)=0 AND X<623 THEN GO
SUB 90:X=X+1:GOSUB 90
80 GOTO 60
90 MOVE X,159
100 TAG
110 PRINT CHR$(224);
120 TAGOFF
130 RETURN
```

```
10 REM PROGRAMA II
20 MODE 1
30 X=5:Y=5
40 ON INT(RND(1)*4)+1 GOTO 50,60,70
,80
50 RX=-1:RY=-1:GOTO 90
60 RX=-1:RY=1:GOTO 90
70 RX=1:RY=-1:GOTO 90
80 RX=1:RY=1
90 IF (X+RX)<1 OR (X+RX)>39 THEN 40
100 IF (Y+RY)<1 OR (Y+RY)>24 THEN 4
0
110 LOCATE X,Y:PRINT CHR$(32);
120 X=X+RX:Y=Y+RY
130 LOCATE X,Y:PRINT CHR$(231);
150 GOTO 90
```

```
10 REM PROGRAMA III
20 MODE 1
30 X=5:Y=5
40 LOCATE X,Y:PRINT CHR$(231)
50 ON INT(RND(1)*4)+1 GOTO 60,70,80
,90
60 RX=-1:RY=-1:GOTO 100
70 RX=-1:RY=1:GOTO 100
80 RX=1:RY=-1:GOTO 100
90 RX=1:RY=1
100 IF (X+RX)<1 THEN GOSUB 190:X=40
:GOTO 160
110 IF (X+RX)>39 THEN GOSUB 190:X=1
:GOTO 160
120 IF (Y+RY)<1 THEN GOSUB 190:Y=24
:GOTO 160
130 IF (Y+RY)>24 THEN GOSUB 190:Y=1
:GOTO 160
140 GOSUB 190
150 X=X+RX:Y=Y+RY
160 LOCATE X,Y:PRINT CHR$(231);
170 IF INT(RND(1)*15)+1=10 THEN 50
180 GOTO 100
190 FOR RETARDO=1 TO 50:NEXT RETARD
O
200 LOCATE X,Y: PRINT CHR$(32);
210 RETURN
```

```
10 REM PROGRAMA IV
20 MODE 1
30 X=1:Y=14
40 LOCATE INT(RND(1)*20)+15,Y
50 PRINT CHR$(143)
60 LOCATE X,Y
70 PRINT CHR$(224)
80 IF TEST(X*16,(((26-Y)*16)-1))<0
 THEN END
90 FOR retardo=1 TO 50:NEXT retardo
100 LOCATE X,Y
110 PRINT CHR$(32)
120 X=X+1
130 GOTO 40
```

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

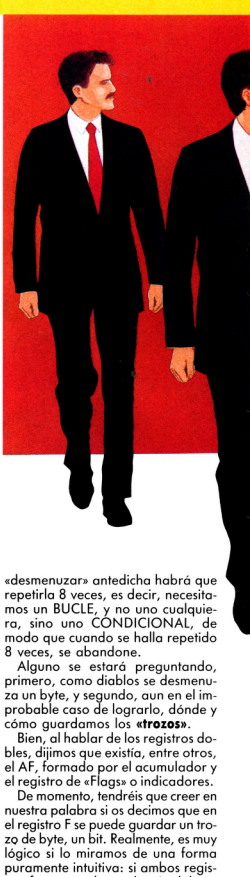
Programas incluidos en la cinta número 2

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.

DE LA TEORIA A LA PRACTICA



De la teoría a la práctica no hay más que un paso, pero a menudo es largo y difícil. Ya que tenemos a nuestra disposición un juego completo de herramientas de programación, vamos a emplearlas aprendiendo de paso que el código máquina es mucho más que bytes y teoría de números.

Aprovechando el ensamblador publicado en el número anterior, además de las pequeñas rutinas que se dieron para utilizarlo, vamos a estudiar en detalle en el presente artículo 3 de ellas, con el doble objeto de usar libremente el ensamblador, enfrentándonos a los posibles problemas que nos causaría un programa normal, y comprobar que el lenguaje máquina tiene utilidad práctica, que sirve para algo, vaya.

SENCILLEZ Y BREVEDAD ANTE TODO

Hay que aclarar que, a la hora de diseñar estas rutinas, nos hemos encontrado con dos problemas:

1. La longitud del programa. El código máquina es un lenguaje de muy bajo nivel, y, normalmente, con cada instrucción puede especificarse sólo una acción *(una línea Basic especifica miles de ellas)*, por lo que hemos tenido que optar por cosas sencillas y, sobre todo, breves, recurriendo, a las llamadas al firmware en la medida de lo posible.

2. La necesidad de usar nuevas instrucciones. En efecto, con las instrucciones de LD, CALL y las que hemos visto hasta ahora, algo puede hacerse, pero no todo. Por ello, nos hemos tomado la libertad de usar instrucciones no explicadas todavía en aras de conseguir programas operativos, tratando siempre de emplear las menos posibles. De todas maneras, serán explicadas, aunque brevemente.

Nuestras tres rutinas se dividen en dos grupos: la número 1 imprime en la pantalla un número decimal en formato binario, las otras dos manejan gráficos de formas parecidas pero diferentes, y están pensadas para usarse en conjunción con un programa Basic, aunque de por sí funcionan bien.

RUTINA NUMERO UNO

Vamos con la mano número 1.

Lo primero de todo es ensamblar en una dirección, la &8 por ejemplo, mediante el directivo ORG.

Una vez hecho esto, cargamos el acumulador (línea 20) con el contenido de la variable BYTE, el número que queremos ver impreso en binario y... lo bueno empieza ahora.

Intentemos analizar la estrategia a seguir para conseguir nuestro propósito; primero los hechos:

1. Un número binario está compuesto, a la postre, de bits, que pueden valer 0 ó 1.

2. Como nuestro número está en el acumulador, sólo puede tener 8 bits.

Ahora el plan:

Lo ideal sería **«desmenuzar»** el acumulador bit a bit, de tal modo que, si nos encontramos un bit que valga 1, imprimir un 1 en la pantalla, y si vale cero, imprimir un cero.

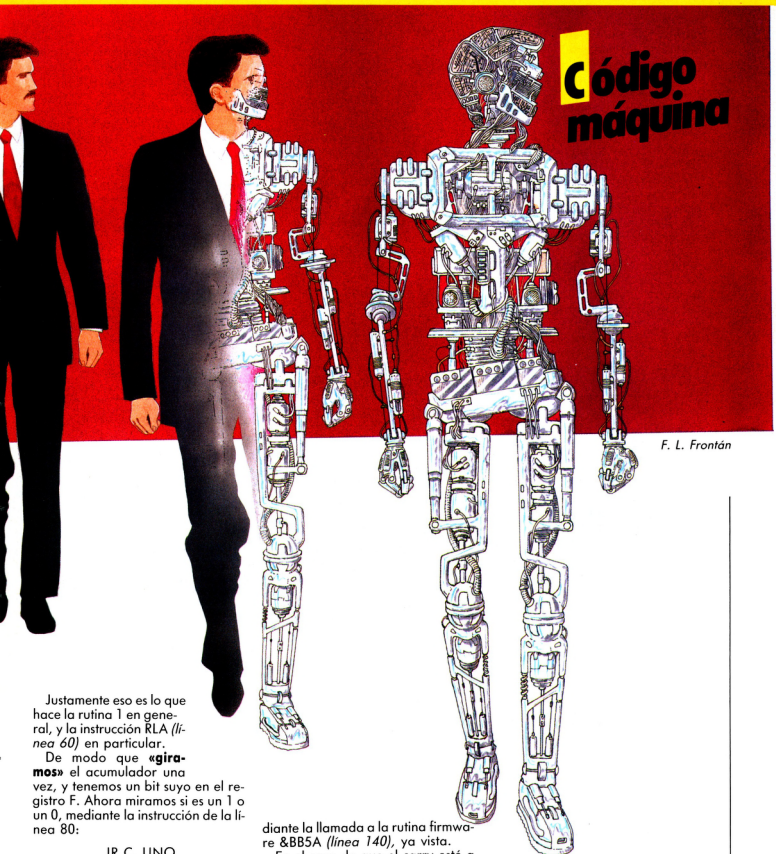
Esta aproximación permite profundizar un poco más en nuestra estrategia: en efecto, según el **«hecho»** número 2, tenemos que bregar con 8 bits, por lo que la operación de «desmenuzar» antedicha habrá que repetirla 8 veces, es decir, necesitamos un BUCLE, y no uno cualquiera, sino uno CONDICIONAL, de modo que cuando se halla repetido 8 veces, se abandone.

Alguno se estará preguntando, primero, como diablos se desmenuza un byte, y segundo, aun en el improbable caso de lograrlo, dónde y cómo guardamos los **«trozos»**.

Bien, al hablar de los registros dobles, dijimos que existía, entre otros, el AF, formado por el acumulador y el registro de «Flags» o indicadores.

De momento, tendréis que creer en nuestra palabra si os decimos que en el registro F se puede guardar un trozo de byte, un bit. Realmente, es muy lógico si lo miramos de una forma puramente intuitiva: si ambos registros forman cada uno la mitad de un círculo, y en cada mitad hay 8 trozos *(bits)*, y yo giro una mitad a la derecha o a la izquierda en pasos de un trozo, uno de estos bits aparecerá en la otra mitad del círculo; si era del acumulador, tras el giro se situará en el registro F.

Código máquina



F. L. Frontán

Justamente eso es lo que hace la rutina 1 en general, y la instrucción RLA *(línea 60)* en particular.

De modo que **«giramos»** el acumulador una vez, y tenemos un bit suyo en el registro F. Ahora miramos si es un 1 o un 0, mediante la instrucción de la línea 80:

JR C, UNO

es decir, hacemos un salto relativo a la etiqueta UNO si el carry está a 1, indicando que el bit que hemos rotado es 1. En la subrutina que comienza en UNO (línea 120), cargamos el acumulador con 49 decimal *(código ASC II de «1», línea 130)*, y lo imprimimos por fin en pantalla mediante la llamada a la rutina firmware &BB5A *(línea 140)*, ya vista.

En el caso de que el carry esté a 0, indicando que el bit que acabamos de rotar es un 0, la condición de la línea 80 no se cumple, por lo que el programa continúa en la línea 90: cargamos el A con el ASC II de **«0»** y lo imprimimos en pantalla, haciendo posteriormente (línea 110) otro salto relativo para no repetir la parte de la rutina que se encarga de imprimir el «1».

Obviamente, si repetimos todo el proceso anterior 8 veces, analizamos 8 trozos *(bits)* del A, consiguiendo

nuestro propósito, que su contenido aparezca en binario en la pantalla. Obsérvese las líneas 50, 70, 160 y 170. En ellas se realiza algo muy importante y útil: poner a salvo el valor actual de cualquier registro doble antes del advenimiento de una rutina que lo alterará *(líneas 50 y 70,* instrucción PUSH), para recuperarlo posteriormente, cuando interese *(líneas 160 y 170, instrucción POP).*

El lugar donde se guardan o preservan los registros se llama **«Stack»** *(en español, «pila», «estructura apilada»),* y por el momento lo ignoraremos. Baste saber que existe y que se usa así.

Guardamos en el stack los registros BC y AF porque son cruciales para nuestra rutina, y en todo momento debemos tener controlado su valor.

Así, B es cargado con 8 *(línea 30),* indicando el número de veces que la rutina se repetirá, y A *(línea 20)* contiene el valor a imprimir en binario.

Por último, controlamos que el bucle que comienza en la línea 40 y acaba en la 180 se repita sólo B veces, mediante la instrucción de la línea 180:

DJNZ BUC

la cual es una orden muy especializada del Z80 y significa: **«Decrementa el valor del registro B; si no es 0, salta a la etiqueta BUC, y si es 0 sal del bucle y ejecuta la instrucción siguiente a ti misma, en este caso RET»** *(DJNZ son las iniciales inglesas de Decrement and Jump if Not Zero, literalmente decrementa y salta si no cero).* Espero que ahora quede claro porque es imprescindible preservar el valor del registro B con PUSH BC y recuperarlo con POP BC inmediatamente antes de la instrucción DJNZ.

RUTINA NUMERO DOS

El programa número 2 está pensado para imprimir en modo transparente, es decir, imprimir sobre algo que ya está en pantalla sin borrarlo, un carácter de un juego de gráficos definido por nosotros mismos, mediante el comando SYMBOL AFTER del Basic.

La rutina no necesita que se le comunique ningún parámetro, porque incorpora una llamada al firmware (línea 20) que consigue que el ordenador espere hasta que pulsemos una tecla, dejando el resultado en el acumulador.

PROGRAMAS

PROGRAMA I

```
10 DATA ORG &8000
20 DATA LD A,(BYTE)
30 DATA LD B,8
40 DATA .BUC
50 DATA PUSH BC
60 DATA RLA
70 DATA PUSH AF
80 DATA JR C,UNO
90 DATA LD A,48
100 DATA CALL &BB5A
110 DATA JR SAL
120 DATA .UNO
130 DATA LD A,49
140 DATA CALL &BB5A
150 DATA .SAL
160 DATA POP AF
170 DATA POP BC
180 DATA DJNZ BUC
190 DATA RET
200 DATA .BYTE
210 DATA DEFB 5
220 DATA END
```

PROGRAMA II

```
10 DATA ORG &8000
20 DATA CALL &BB06
30 DATA LD D,0
40 DATA LD E,A
50 DATA LD HL,STORE
60 DATA CALL &BBAB
70 DATA CALL &BB78
80 DATA CALL &BC1A
90 DATA LD DE, STORE
100 DATA LD B,8
110 DATA .BUC
120 DATA LD A,(DE)
130 DATA XOR 255
140 DATA LD (HL),A
150 DATA INC DE
160 DATA LD A,B
170 DATA LD BC,2048
180 DATA ADD HL,BC
190 DATA LD B,A
200 DATA DJNZ BUC
210 DATA RET
220 DATA .STORE
230 DATA DEFS 2048
```

PROGRAMA III

```
10 DATA ORG &8000
20 DATA LD A,1
30 DATA CALL &BC0E
40 DATA LD IX,STORE
50 DATA LD HL,(PMEM)
60 DATA CALL &BC1A
70 DATA LD DE,2048
80 DATA LD B,8
90 DATA .BUC
100 DATA LD A,(IX+0)
110 DATA LD (HL),A
120 DATA INC IX
130 DATA ADD HL,DE
140 DATA DJNZ BUC
150 DATA RET
160 DATA .PMEM
170 DATA DEFS 2
180 DATA .STORE
190 DATA DEFS 8
```

Necesitamos definir un espacio de memoria para que los caracteres se ubiquen adecuadamente, y lo hacemos mediante el directivo del ensamblador DEFS, que guarda, en este caso, 2.048 bytes, espacio para 256 caracteres *(línea 230).*

Hay cuatro llamadas al firmware, relacionadas entre sí de la siguiente manera:

1. CALL &BB06. Espera a que se pulse una tecla, como comentábamos antes, y deja el resultado en el acumulador.

2. CALL &BBAB (línea 60). Esta rutina requiere dos parámetros: la dirección donde queremos cargar los caracteres en el registro HL *(línea 50),* y el primer caracter que hay que trasladar al nuevo lugar de almacenamiento en DE; por eso cargamos D con 0 y E con el caracter que hay en el acumulador *(líneas 30 y 40),* el cual, recordemos, se obtiene directamente del teclado.

Nuestra siguiente tarea es imprimir el caracter elegido en la pantalla y, para no tener que darle al programa la posición como parámetro, recurrimos a:

3. CALL &BB78 (línea 70). La rutina obtiene la posición actual del cursos de texto en la pantalla, y la deja en el registro HL, en L la columna y en H la fila.

Por último, para colocar nuestro caracter en forma transparente sobre lo que haya en ese momento en pantalla en esa posición, acudimos a:

4. CALL &BC1A *(línea 80).* Decodifica la dirección de pantalla que encuentra en HL *(la cual viene de la rutina anterior)* y deja el resultado de nuevo en HL.

Después de la línea 90 tenemos el siguiente panorama:

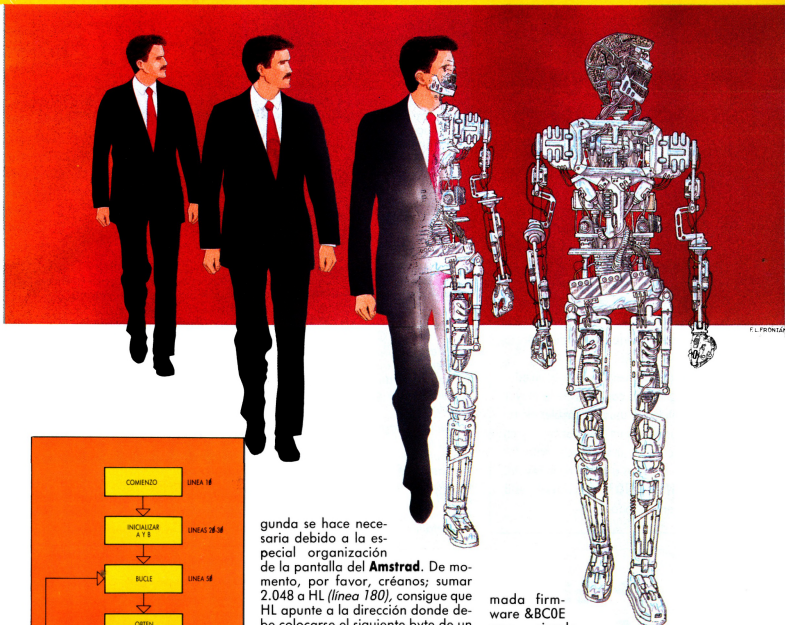
DE apunta a la zona de memoria donde se encuentra nuestro juego de caracteres, al primero de ellos.

HL apunta a una dirección de pantalla que corresponde a la posición del cursor.

Todo lo que tenemos que hacer es volcar los 8 bytes que constituyen un caracter mediante un bucle del mismo tipo que el del programa 1 *(líneas 110-210).*

En este bucle, si se estudia detalladamente, se verá que sólo hay dos órdenes conflictivas *(líneas 130 y 170).*

En la primera, la orden XOR 255 consigue el efecto de transparencia como se verá claramente al estudiar este grupo de instrucciones, y la se-



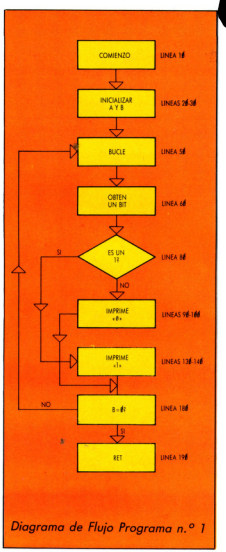


Diagrama de Flujo Programa n.º 1

gunda se hace necesaria debido a la especial organización de la pantalla del **Amstrad**. De momento, por favor, créanos; sumar 2.048 a HL *(línea 180)*, consigue que HL apunte a la dirección donde debe colocarse el siguiente byte de un caracter.

Recomendamos el atento estudio de este programa, y practicar con él, junto con otro Basic que cree los caracteres definidos. Enseguida se comprenderá su lógica de funcionamiento.

RUTINA NUMERO TRES

Por último, el programa tres también imprime en pantalla un gráfico, pero debemos definirlo antes dándole los 8 números binarios que lo constituyen en el área de almacenamiento *(.STORE)*, desde Basic con la sentencia POKE. Es fácil, simplemente al ensamblar tome nota de la dirección correspondiente a la etiqueta .STORE y luego «poke» en ella.

Lo único nuevo de esta rutina es el uso de IX como puntero para indexar una zona de memoria, y la llamada firmware &BC0E que, simplemente, pone la pantalla en modo 1, o más exactamente, en el número almacenado en el acumulador *(linea 20)*.

En este caso, el programa necesita la posición de pantalla, la cual pokeamos desde Basic en .PMEM, de la misma forma explicada antes.

Esperamos que estos tres programas sirvan de utilidad, y que, usando vuestra imaginación, lo combinéis con el Basic para ver la potencia y efectividad del lenguaje máquina.

NOTA: El manual del ordenador explica con claridad cómo salvar en cinta/disco código máquina. Las direcciones de comienzo y final de la rutina ensamblada, y, por tanto, su longitud, se obtienen fácilmente anotando la dirección de ORG (normalmente &8000) y la última que el ensamblador presenta en pantalla al final de la segunda pasada.

Sin duda alguna

A través de esta sección se pretende resolver, en la medida de lo posible, todas las posibles dudas que **«atormenten»** a todas las personas interesadas en el mundo del AMSTRAD, sean o no poseedores de uno y, si lo son, se encuentren en cualquier nivel de destreza en su manejo.

Semanalmente, aparecen en estas páginas las consultas de la mayor cantidad de usuarios posible; ello redundará en un mejor servicio y en un contacto más estrecho entre todos nosotros a través de la revista.

SIN DUDA ALGUNA está abierta a todos.

Queridos amigos:

Soy un estudiante de informática y como tal lleno de dudas informáticas. También soy dueño de un **Amstrad** CPC464. Mi gran curiosidad sobre el aparato son las rutinas internas llamadas con el comando CALL, ya que en muchos programas son mencionados sin ninguna explicación.

Agradecería mucho dicha explicación o el nombre de algún libro que lo explique.

Esperando que esta pregunta sea de su interés y confiando en su respuesta se despide este amigo.

P.D. Enhorabuena por la revista **MICROHOBBY AMSTRAD.**

Jesús Pérez Narezo / ***Madrid***

El comando CALL se usa para llamar desde el Basic a una rutina escrita en lenguaje máquina, que comienza en la dirección especificada por el número que sigue a este comando.

Muchos programas, efectivamente, las usan, bien escritas por el propio autor o bien aprovechando las rutinas del Firmware, que ya están escritas en la memoria de la máquina. El libro del Firmware del ***Amstrad*** *analiza todo esto con todo lujo de detalles.*

COLABORADORES:

Si eres poseedor de un **Amstrad**, tienes conocimientos de programación en Basic u otros lenguajes y te sientes capacitado para escribir acerca de un determinado tema directamente relacionado con el mundo de **Amstrad**, envíanos una carta mecanografiada, indicando los siguientes datos:

Edad
nombre y dirección
teléfono de contacto
relación de tus conocimientos
lenguaje que dominas

Se valorará el conocimiento del inglés.

Dirigirse a:

C/ La Granja, s/n. Polígono Industrial de Alcobendas (Madrid) indicando claramente en el sobre

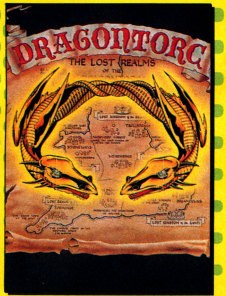
AMSTRAD SEMANAL

¡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

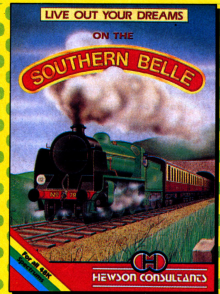
Cerca de 200 pantallas con miles de objetos diferentes y más de cien personajes con animación en tres dimensiones, hacen que de este juego la revista inglesa Crash Micro haya llegado a decir «Dragontorc es lo mejor que hemos visto en juegos de acción y aventura».

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 _____

Mercado común

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

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

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

Vendo Amstrad CPC-664 disk, Monitor color, nuevo y con garantía, comprado en julio, con todos sus programas y con 4 diskettes de obsequio, todo por 112.000 ptas. Tel. (977) 32 03 13 y Apdo. 480 de Reus (*Tarragona*).

Vendo «Sinclair QL» con garantía Investrónica, muy poco uso. Incluyo un libro manual de referencia en castellano y un juego de ajedrez. Todo por 115.000 ptas. Negociables. Luis (91) 270 81 68.

Desearía contactar con usuarios del CPC 464 para intercambio de ideas, información, etc. Dirigirse a Vicente Belaguer Gómez. Avda. José Antonio, 81. Onda *(Castellón)*, o llamar al teléfono (964) 60 29 63. Les saluda dándoles las gracias.

Urge **vender Amstrad** CPC-664 (*diskette y monitor de fósforo verde*); totalmente nuevo. Regalo joystick, programas en diskette, y dos diskettes con el lenguaje Logo, CP/M y Utilidades y seis Programas Comerciales. La dirección es: Angel Calva Bustamante. Juan José Pérez del Molino, 11 - 1.º D. 39006 Santander. Tel. (942) 37 24 40 (llamar de 1 a 3.30 PM).

Intercambio todo tipo de programas en cassette para el **Amstrad** CPC 464. Escribir a: Luis Antonio Cortiñas Rodríguez. Alfredo Brañas, 6 - 1.º A, Escalera A. Santiago *(Coruña)*.

Vendo Amstrad 464 con disco y un completo paquete de software de gestión. Precio muy interesante. Tel. (91) 269 98 33.

GANA 100.000 PESETAS CON MICROHOBBY AMSTRAD SEMANAL

Porque pretendemos que **AMSTRAD** SEMANAL sea también vuestra revista, hemos abierto una sección en la que se publicarán los mejores programas originales recibidos en nuestra redacción. Vosotros seréis los encargados de realizar estas páginas, en las que podréis aportar ideas y programas interesantes para otros lectores.

Las condiciones son sencillas:

— Los programas se enviarán a **AMSTRAD** SEMANAL en una cinta de cassette, sin protección en el software, de forma que sea posible obtener un listado de los mismos.

— Cada programa debe ir acompañado de un texto explicativo en el cual se incluyan:

- *Descripción general del programa.*
- *Tabla de subrutinas y variables utilizadas, explicando claramente la función de cada una de ellas.*
- *Instrucciones de manejo.*

— Todos estos datos deberán ir escritos a máquina o con letra clara para mayor comprensión del programa.

— En una sola cinta puede introducirse más de un programa.

— Una vez publicado, **AMSTRAD** SEMANAL abonará al autor del programa de **15.000** a **100.000** pesetas, en concepto de derechos de autor.

— Los autores de los programas seleccionados para su publicación, recibirán una comunicación escrita de ello en un plazo no superior a dos meses a partir de la fecha en que su programa llegue a nuestra redacción.

— **AMSTRAD** SEMANAL se reserva el derecho de publicación o no del programa.

— Todos los programas recibidos quedarán en poder de **AMSTRAD** SEMANAL.

— Los programas sospechosos de plagio serán eliminados inmediatamente.

¡ENVIANOS TU PROGRAMA!

Indicando claramente en el sobre:

AMSTRAD SEMANAL

a HOBBY PRESS, S. A. La Granja, n.º 8. Pol. Ind. Alcobendas (Madrid)

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 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

DISPONIBLE PARA ZX SPECTRUM
AMSTRAD

Sound on Sound es una marca registrada producida y distribuida por Iberofón, S. a.
Telf. 671.22.00 / 04 / 08 / 12 / 16

Sound-on-Sound

JUEGA CON EL FUTURO

SOFTWARE

¡¡¡NO LO SUEÑES!!! ¡JUEGALO!
SIENTE LA EMOCION DE LO DESCONOCIDO
CORRE TU PROPIO RIESGO
SALVA A TU COMPAÑERO/A-ATRAPADO/A
REUNE LOS FRAGMENTOS DEL CUADRO
SON TU AMULETO

¡¡¡POR FIN EN CASTELLANO!!!
LA PRIMERA COMEDIA MUSICAL EN VIDEO-JUEGO

Graphis, S.A. Publicidad