AÑO I N.º 10

MICROHOBBY

AMSTRAD Semanal

REVISTA INDEPENDIENTE PARA USUARIOS DE ORDENADORES AMSTRAD

AÑO I N.º 10

150 Ptas.

Canarias 160 ptas.

PCW8256: Probamos a fondo el nuevo modelo de la gama AMSTRAD

AMPLIA EL BASIC DE TU CPC464 CON LOS COMANDOS: FILL, GET, SETCLOCK Y OTROS MEDIANTE EL USO DEL SISTEMA RSX

AMPLIACION DE MEMORIA PARA EL **664** HASTA **256** K

CUATRO EN RAYA: INTELIGENCIA EN 3D

SOFTWARE

Dragontorc de Avalon, el regreso del mago Maroc desde el corazón del bosque de Wispwood Forest

¡FANTASTICO!

Si quieres un ordenador de «una pieza» piensa en el AMSTRAD CPC 464. Tendrás un ordenador de una vez por todas. Gracias a sus 64K RAM y 32K ROM y a sus casi ilimitadas posibilidades de crecimiento, tienes garantizado que el ordenador CPC 464 no se te quedará pequeño.

COMPLETO

Además, gracias a su monitor (color o fósforo verde) de alta resolución (hasta 640 x 200 pixels direccionados individualmente) y a su unidad de cassette incorporada al teclado, podrás disfrutar de tu AMSTRAD de una manera independiente, prescindiendo del televisor y del radiocassette de tu casa (a veces tan solicitados).

¿Y QUE ME DICES DE LOS PROGRAMAS?

Actualmente ya hay cientos de ellos disponibles en España. Sin olvidar que son varias las revistas dedicadas solo a AMSTRAD y que el número de libros y periféricos del CPC 464 crecen día a día, potenciando así la creatividad de tu ordenador personal.

CARACTERISTICAS TECNICAS

- *Microprocesador Z80 (4MHz).*
- *Memoria de 64K RAM y 32K ROM.*
- *Gráficos de alta resolución de hasta 640 por 200 pixels direccionables individualmente.*
- *Unidad de cassette incorporada en el teclado.*
- *Monitor color o fósforo verde incluido en el Sistema.*
- *Texto en pantalla de 20,40 y 80 columnas por 25 líneas.*
- *LOCOMOTIVE BASIC ampliado.*
- *Paleta de 27 colores y efectos de «flash».*
- *Teclado profesional tipo QWERTY con bloque numérico y teclas para cursor independientes.*
- *Salida Centronics paralelo.*
- *Lector de discos de 3" (180K por cara) opcional (con CP/M y Dr. LOGO incluidos junto a la unidad de disco).*
- *Manuales en castellano.*

Al comprar tu ordenador CPC 464, AMSTRAD ESPAÑA te obsequia con 8 cassettes de programas y el libro «Guía de Referencia BASIC para el programador».

Exije la ***GARANTIA AMSTRAD ESPAÑA UNICA VALIDA PARA ACCEDER AL SERVICIO TECNICO OFICIAL.***

PRECIO:

— ***66.900 ptas.***
(monitor fósforo verde)

— ***95.900 ptas.***
(monitor color)

¡¡Increible!!

ESPAÑA

AMSTRAD

GARANTIA INDESCOMP

Avd. del Mediterráneo, 9 - 28007 Madrid Tels. 433 45 48 - 433 48 76 - Telex 47660 FAX - 4332450

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

Edita
HOBBY PRESS S.A.

Presidente
María Andrino
Consejero Delegado
José I. Gómez-Centurión

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

Secretaria de Dirección
Marisa Cogorro

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

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

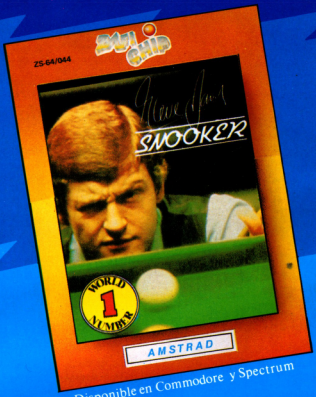
AMSTRAD

sumario

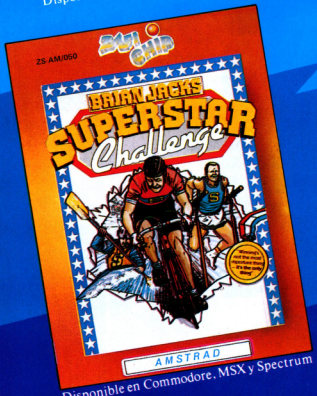
Año I • Número 10 • 5 al 11 de Noviembre de 1985
150 ptas. (sobretasa Canarias, 10 ptas.)

¡Ahora en AMSTRAD los mejores juegos!

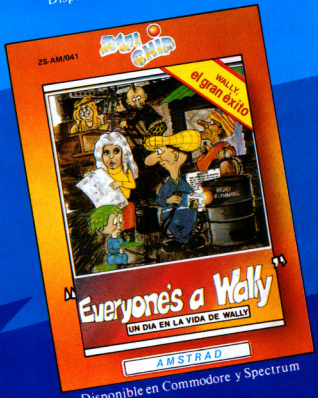
Disponible en Commodore y Spectrum



Disponible en Commodore y Spectrum



Disponible en Commodore, MSX y Spectrum



Disponible en Commodore y Spectrum

Si no los encuentras en tu tienda habitual...Llamanos!

ZAFIRO SOFTWARE DIVISION
Paseo de la Castellana, 141. 28046 Madrid.
Tel. 459 30 04. Tel. Barna. 209 33 65. Telex: 22690 ZAFIR E

ERICSSON VUELVE A CASA

La multinacional sueca no venderá más ordenadores en Estados Unidos, debido lógicamente, a los malos resultados que sus productos están teniendo en el extremadamente difícil y competitivo mercado americano.

Los suecos habían hecho tratos con una empresa petrolífera norteamericana, la Atlantic Richfield, para vender sus máquinas. A través de ellos, han comunicado que los 175 contratos con sus distribuidores no se renovarán a final de año.

Ericsson ha decidido dedicarse exclusivamente al mercado europeo, algo más complaciente y menos competitivo.

MICROSOFT EN IRLANDA

La compañía norteamericana Microsoft, creadora del sistema operativo del IBM PC, el famoso PC DOS (MS DOS para los compatibles), ha decidido asentarse firmemente en Europa estableciendo una fábrica en Irlanda, la cual se espera que alcance muy pronto la producción de 500.000 programas anuales, en su mayor parte dirigidos al Apple y al IBM PC.

¿COMO VAN LAS COSAS?

En el sector informático, pese a informes alarmistas, las cosas parecen ir viento en popa. Según el Ministerio de Industria y Energía, la facturación total de las empresas informáticas creció un 30,5 por 100 en 1984.

En España se facturó la astronómica cifra de 49.324 millones de pesetas, frente a los 37.806 de 1983.

También aumenta rápidamente el ritmo de los ordenadores instalados en nuestro país, a un ritmo de aproximadamente un 8 por 100 anual.

Así que ya sabe: ante los rumores de crisis en el ordenador, abrume a los augures con cifras **«irrebatibles».** Las estadísticas no mienten.

AMPLIACION DE MEMORIA PARA EL 664

Desde la aparición del CPC6128, **Amstrad** ha recibido críticas muy duras debido a lo que muchas personas han interpretado como una deliberada indiferencia hacia sus clientes, una vez que han adquirido sus máquinas.

Me refiero a que los felices compradores de uno CPC664, poco, muy poco tiempo después, se han encontrado con la aparición en el mercado de un nuevo modelo de superiores prestaciones que el suyo y encima al mismo precio.

Alan Sugar, chairman de la compañía británica, salió del paso como pudo explicando que el advenimiento del CPC6128 se debía a **«inevitables exigencias tecnológicas»,** y no cabe duda de que le asiste cierta o toda la razón. Los ordenadores de 64 K se están quedando pequeños a pasos agigantados.

Las declaraciones de Sugar puede que mitigaran el escozor de la herida de los dueños de un 664, pero aparentemente no resolvían mucho.

No obstante, a instancias de **Amstrad** y de Indescomp, sus distribuidores en España, la compañía inglesa D'ktronics, conocida, entre otras cosas, por su teclado profesional para el ZX-Spectrum, tiene en un estado de desarrollo muy avanzado ampliaciones de memoria para el CPC664.

Las nuevas ampliaciones permitirán igualar en prestaciones al 664 con el 6128, e incluso superarle en lo que a memoria RAM se refiere.

El 664 podrá tener un total de 128 K o de 256 K de memoria. Buenas noticias para los usuarios del 664, a la espera, naturalmente, de los precios a los que se venderán estas ampliaciones.

BUCLES ANIDADOS

Raro es el programa que no hace uso de la iteración, o repetición de una serie de sentencias un número determinado de veces, como una sencilla y potente técnica de programación. Para eso están los bucles, pero, ¿pueden anidarse?, ¿pueden usarse para representaciones gráficas?

F. L. Frontán

La última vez analizamos el uso de los bucles FOR... NEXT para repetir un cuerpo de instrucciones por un número fijo de veces. Vimos que podemos usar las mismas líneas de programa una y otra vez intercambiándolas entre un FOR y un NEXT.

Programa uno

Esto no sólo aumenta la potencia de un programa sino que ambién ahorra mucho. El programa 1 no debe presentarnos ninguna dificultad.

Es una utilización muy elemental del bucle FOR... NEXT formado por las líneas 20 y 40. El cuerpo es la línea 30. Imprime un asterisco en la pantalla cada vez que el bucle realiza un ciclo.

El punto y coma que está al final de la línea asegura que los asteriscos se «peguen» juntos, uno tras otro.

El número de veces que el bucle hace el ciclo está determinado por la variable de control **«línea»**. Empieza con un valor de 1 y se incrementa en 1 cada vez que el programa llega al NEXT.

Cuando el valor de **«línea»** finalmente llega a 11 el bucle finaliza, habiendo hecho el ciclo 10 veces. El resultado es una línea de 10 asteriscos extendida a través de la pantalla.

¿Pero dejemos el programa sin hacerle correr. Probemos, cambiando la variable de control **«línea»**, a producir líneas de 20 ó 30 asteriscos. Preste atención a esto:

```
FOR línea=10 TO 19
```

o

```
FOR línea=99 TO 90 STEP —1
```

producen también 10 asteriscos. No siempre tiene que ser 1 el valor inicial de la variable de control del bucle.

La razón principal es que cada instrucción entre el FOR y el NEXT se repite. No importa si es un comando PRINT o un LET o cualquier otro. Mientras el bucle aún está funcionando, todo lo interior a los límites de FOR... NEXT se realiza repetidas veces.

Programa dos

Esto también ocurrirá cuando haya otro bucle FOR... NEXT que envuelva el cuerpo del primer bucle. El programa II demuestra lo que queremos decir.

Las líneas 30 a 50 no deben tener misterios, son las mismas que las del FOR... NEXT del programa anterior. Como vimos antes este bucle concretamente hará el ciclo 10 veces y producirá una línea de 10 asteriscos. Sin embargo, con el programa II conseguiremos 5 líneas de 10 asteriscos cada una. **¿Cómo ocurre?**

La respuesta está en el FOR... NEXT formado por las líneas 20 y 70. Mirando la línea 20 podemos ver que este bucle hará el ciclo 5 veces mientras que la variable de control **«nuevalínea»** toma valores de 1 a 5.

No se preocupe nunca de las líneas intermedias. Por el momento concéntrese sólo en que el bucle exterior hace ciclos 5 veces.

Ahora, como vimos antes, todo lo interior al FOR... NEXT se repite cuantas veces el bucle hace el ciclo. Así que su cuerpo —que es todo lo que hay entre el FOR de la línea 20 y el NEXT de la línea 70— se repetirá 5 veces.

Sin embargo, en vez de ser solamente un comando PRINT como en el programa anterior, el cuerpo del bucle es ahora otro FOR... NEXT. Y, como vemos, necesita hacer el ciclo 10 veces. Esto es, en realidad, lo que ocurre.

Cada vez que el bucle exterior, controlado por **«nuevalínea»**, se repite una vez, el bucle interior se ejecuta en él completamente. Es decir, por cada ciclo del bucle exterior, el bucle interior, controlado por «línea», se repite 10 veces produciendo la conocida línea de asteriscos.

Como el exterior hace ciclos 5 veces en total, y el interior repite el ciclo 10 veces cada vez que se recorre el otro, hemos conseguido 5 filas de 10 asteriscos.

Esto es un ejemplo de lo que conocemos como **bucle anidado**. En este caso es un FOR... NEXT (líneas 30-50) anidado en el interior de otro (líneas 20-70). Cada vez que el bucle exterior hace un nuevo ciclo, el bucle interior repite su ciclo hasta que completa su número total de repeticiones.

De nuevo, no nos contentamos con el programa ejemplo, vamos a probar a variarlo. ¿Qué nos gustaría que estas líneas:

```
20 FOR nuevalínea=10 TO 15
30 FOR línea=10 TO 5 STEP —1
```

o éstas

```
20 FOR nuevalínea=10 TO 50 STEP 5
30 FOR línea=15 TO 21 STEP 2
```

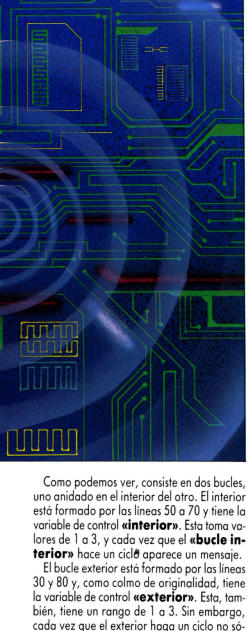
produjeran? El número de asteriscos en cada línea y el número de líneas, ¿se corresponden con sus previsiones?

Atención al cuidado que ha de tener con las variables de control para evitar realizar más ciclos de cada bucle que los que realmente desea.

A propósito, **¿puede ver la utilidad de la línea 60?** Es aquí donde se deshace el efecto del punto y coma del final de la línea 40 cuando el programa sale del bucle interior. Lo que produce es lo que conocemos como retorno de carro. Intentemos omitirlo y ver qué sucede.

Programa tres

Armados con nuestro hallazgo del bucle anidado podemos volver al programa III del artículo anterior *(programa III de este número)* y ver cómo trabaja.



Primeros pasos

Como podemos ver, consiste en dos bucles, uno anidado en el interior del otro. El interior está formado por las líneas 50 a 70 y tiene la variable de control **«interior»**. Esta toma valores de 1 a 3, y cada vez que el **«bucle interior»** hace un ciclo aparece un mensaje.

El bucle exterior está formado por las líneas 30 y 80 y, como colmo de originalidad, tiene la variable de control **«exterior»**. Esta, también, tiene un rango de 1 a 3. Sin embargo, cada vez que el exterior haga un ciclo no sólo escribe un mensaje, sino que también se realizan los tres ciclos del interior.

En el momento en que el bucle exterior ha completado el número total de repeticiones, el bucle interior habrá hecho su cantidad total 3 veces. Sigamos la trayectoria del programa. Las líneas 10 y 20 son precisamente las REMs, que el **Amstrad** ignora. La línea 30 cuenta al micro que está comenzando un bucle y que la variable **«exterior»** se utilizará como control de bucle. Inicialmente tiene un valor de 1.

El programa va entonces a la línea 40, y saca el mensaje:

Este es el bucle exterior 1

La línea 50 señala el comienzo de otro bucle FOR... NEXT. Este bucle tiene la variable de control «interior» tomando valores de 1 a 3. En esta etapa el **Amstrad** da a **«interior»** el valor 1 y va a la línea 60. Esto se transforma en el mensaje:

Este es el bucle interior 1

y el programa va al NEXT de la línea 70. Aquí «interior» se incrementa en 1 y el programa vuelve al FOR de la línea 50. Puesto que «interior» está dentro de los límites señalados, el control va a la línea 60 y el mensaje:

Este es el bucle interior 2

aparece en pantalla.

Otra vez se encuentra con el NEXT de la línea 70, **«interior»** se aumenta a 3 y el programa regresa a la línea 50. La línea 60 se escribe ahora:

Este es el bucle interior 3

y el programa da con el NEXT de la línea 70 otra vez. Ahora, sin embargo, cuando «interior» aumenta a 4 se exceden las condiciones del bucle. El programa sale y alcanza la línea 80. Aquí encuentra el NEXT del bucle exterior.

Entonces **«exterior»** se aumenta a 2 y el programa vuelve a la línea 30. La línea 40 se escribe:

Este es el bucle exterior 2

y la línea 50 devuelve el programa al bucle interior, produciendo los tres mensajes **«interior»** como antes.

Una vez que esto está hecho el programa sale del bucle interior y se encuentra otra vez el NEXT de la línea 80, **«exterior»** llega a ser 3 ahora y el programa regresa a la línea 30 por última vez. Cuando **«exterior»** es 4 el programa termina.

No se preocupe si no lo comprende en toda su extensión. Los bucles anidados necesitan ser trabajados un poco, pero una vez que se haya ejercitado con ellos, pasarán a ser un asunto trivial.

Todo lo que tiene que comprender es que por cada ciclo sencillo del bucle exterior el bucle interior hace todos sus ciclos. Añadiendo una línea parecida a:

```
55 PRINT «exterior»=exterior
      «interior»-interior
```

se aclararían las cosas.

Programa cuatro

Mientras hemos estado con el programa III debimos señalar que no necesita colocar variables de control después de NEXT. El programa IV lo demuestra.

No obstante, mientras el **Amstrad** puede ser bastante hábil al mantenerse al tanto de cosas, nosotros podríamos no serlo. Siga nuestros consejos, y coloque siempre la variable, al menos hasta que haya conseguido que su programa funcione correctamente.

Observemos que el programa IV es una variante del programa III. Sólo ha tenido dos alteraciones sin importancia en una parte de las líneas, pero vea la diferencia de resultados. Como podemos apreciar, anidando bucles FOR... NEXT uno dentro del otro, ambos se hacen más flexibles y poderosos.

Hasta aquí sólo hemos anidado un bucle FOR... NEXT dentro de otro. Es perfectamente posible tener 3 o más bucles anidados. Más bien se parecen a esas muñecas rusas o «matrioskas» que aparecen en la portada de las historias de espías. Funcionan de un modo muy semejante pero se les comprende más fácilmente en la práctica que en la teoría. Añadamos:

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

al programa III y veamos lo que queremos decir. Y observe qué enorme incremento en los resultados a causa sólo de 3 líneas extras. Como dijimos, estos bucles anidados son muy potentes.

Programa cinco

Una cosa, debemos ser cuidadosos cuando estemos ocupados con los bucles anidados, para colocar los NEXT en el orden correcto. El programa V lo demuestra muy bien.

Cuando lo hagamos correr obtendremos el mensaje de error:

Unexpected NEXT in 70 *(NEXT inesperado...)*

No nos llevará mucho tiempo descubrir que hemos confundido al computador al encontrar que las variables del control en las líneas 60 y 70 están intercambiadas erróneamente. Tendría que ser:

```
60 NEXT interior
70 NEXT exterior
```

Si realmente desea que probemos sus conocimientos sobre los bucles anidados nos pueden aclarar, ¿por qué cambiando las últimas líneas a:

```
60 NEXT exterior
70 NEXT exterior
```

producirían el mensaje:

```
Este es el bucle exterior 1R
 Este es el bucle exterior 1
```

antes que el mensaje de error?

Programa seis

Pero basta ya de decir que podemos equivocarnos, esto es demasiado teórico. Después de todo, nosotros no podemos cometer errores, **¿verdad?** Vamos a estar demasiado ocupados haciendo cosas importantes, tales como producir triángulos de asteriscos. Veamos el programa VI.

Aunque el resultado no nos entusiasme, el programa contiene algunos puntos interesantes. Por ahora somos capaces de reconocer los juegos de bucles anidados y el CHR$(13) no nos asusta.

El bucle exterior, con la variable de control **«longitud»** variando de 1 a 10, es bastante sencillo. El bucle interior es algo diferente. Aquí la variable de control **«línea»** va desde 1 hasta **«longitud»**. Ahora **«longitud»** cambia en cada vuelta del bucle exterior.

En la primera vuelta **«longitud»** es 1. Por tanto, la variable de control del bucle interior va de 1 a 1. El primer resultado es un asterisco solitario.

En la siguiente vuelta del exterior, **«longitud»** es 2, así que ahora cuando el programa realice el bucle interior, **«línea»** va desde 1 a 2. Aparecen un par de asteriscos.

La tercera vez que se hace el exterior, **«longitud»** es 3, de modo que el interior saca un trío de asteriscos puesto que **«línea»** va de 1 a 3. En el momento en que **«longitud»** es 10, aparecen 10 asteriscos.

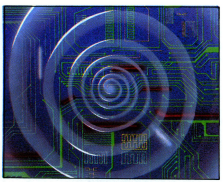
La finalidad es comprender que el número de repeticiones del bucle interior depende de la variable de control del bucle exterior. **Los bucles no sólo están anidados, el exterior controla al interior.**

Programa siete

¿Podemos alterar el programa para que realice el mismo triángulo pero al revés? El programa VII nos lo demuestra.

Aquí la **«longitud»** de decrementa cada vez que se hace el bucle exterior, por lo tanto, se reduce el número de asteriscos en cada línea.

Una vez seguros de haber comprendido los dos últimos programas tenemos que conseguir sacar **«la imagen sobre el espejo»** de los asteriscos del programa VI.



PROGRAMAS

```
10 REM PROGRAMA I
20 FOR linea=1 TO 10
30 PRINT "*";
40 NEXT linea
```

```
10 REM PROGRAMA II
20 FOR nuevalinea=1 TO 5
30 FOR linea=1 TO 10
40 PRINT "*";
50 NEXT linea
60 PRINT CHR$(13)
70 NEXT nuevalinea
```

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

```
10 REM PROGRAMA IV
20 FOR exterior=1 TO 5
30 PRINT"Este es el bucle exterior"
 exterior
40 FOR interior=1 TO 4
50 PRINT TAB(10) "Este es el bucle
interior" interior
60 NEXT
70 NEXT
```

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

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

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

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

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

Programa ocho

No es tan sencillo como el programa anterior, pero si recordamos cómo trabaja LOCATE no tendremos ningún problema en saber cómo funcionaría. El programa VIII es una muestra.

Y ahora, **¿podemos volverle al revés?** Los cambios son:

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

Y esto es todo en esta ocasión. Le dejamos que se entretenga con los bucles anidados, continúe probando hacer formas con asteriscos.

Aunque esto no sea el uso más importante de su **Amstrad**, le asombrará cómo aumentará su conocimiento de los bucles anidados y de los posibles usos de esta técnica en sus propias aplicaciones.

Y cuando se canse de ellos, **¿puede comprender lo que ocurre en el programa IX?**

Produce el mismo resultado que el programa VIII, pero tiene un solo bucle en lugar de dos. **¿Qué ocurre?** La respuesta está en las variables literales.

Te presentamos un equipo sensacional: el ***AMSTRAD CPC 6128****.*

Con un sólo cable para enchufar a la red, el Sistema 6128 está listo para funcionar.

JUEGA Y APRENDE CON EL 6128

Para jugar, el 6128 es un ordenador muy serio, gracias a sus cientos de programas disponibles, tienes aseguradas horas de entretenimiento. Y en el mundo de la enseñanza no es menos.

Gracias a sus sensacionales capacidades gráficas (paleta de 27 colores y hasta 640 x 200 PIXELS) y sonoras (3 voces y 8 octavas, altavoz interior y salida stéreo) el 6128 es una herramienta inigualable. Además, dentro del paquete de programas que se entrega con el sistema, está incluido el lenguaje educativo por excelencia: el ***Dr. LOGO*** *de Digital Research. Y para profundizar en el lenguaje de la informática recuerda que el 6128 es el ordenador idóneo, ya que posee uno de los más rápidos y potentes BASIC —el* ***LOCOMOTIVE BASIC****—, así como otros muchos lenguajes de programación:* ***FORTH, PASCAL,*** *etc.*

TRABAJA CON EL 6128

Haz un sitio en tu negocio al 6128. Planifica presupuestos, lleva contabilidades, gestiona archivos, todo fácilmente gracias a su Sistema Operativo CP/M (en versiones 2.2 y Plus), que (como ya sabes) te permitirá acceder a la más extensa biblioteca de programas profesionales: bases de datos, procesadores de textos, hojas de cálculo, etc.

CARACTERISTICAS TECNICAS

- *128K RAM y 48K ROM (incluye Locomotive BASIC y Sistema Operativo).*
- *Monitor: Color de 14" y fósforo verde de 12".*
- *Unidad de Disco 3" incorporada (180K por cara).*
- *Teclado profesional.*
- *Sistema Operativo: AMS-DOS, CP/M 2.2 y CP/M Plus.*
- *Salida para segunda unidad de disco y cassette externa.*

El CPC 6128 incluye en su suministro:

- *Disco con Sistema Operativo CP/M 2.2 y lenguaje Dr. LOGO.*
- *Disco con Sistema Operativo CP/M Plus y utilidades.*
- *Disco con seis programas de obsequio.*
- *Manuales en castellano.*
- ***GARANTIA AMSTRAD ESPAÑA UNICA VALIDA PARA ACCEDER AL SERVICIO TECNICO OFICIAL.***

TODO POR:

— 109.500 ptas.
(monitor fósforo verde)

— 134.500 ptas.
(monitor color)

¡¡Increible!!

ESPAÑA

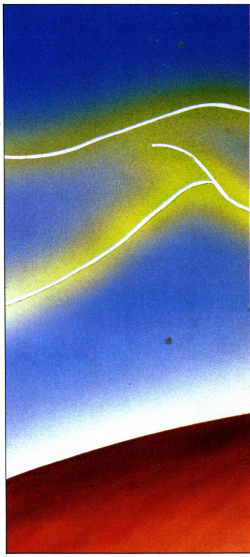
AMSTRAD

GARANTIA INDESCOMP

Avd. del Mediterráneo, 9 - 28007 Madrid Tels. 433 45 48 - 433 48 76 - Telex 47660 FAX - 4332450

AMPLIACION DEL BASIC: RSX

Los usuarios del CPC464 se han encontrado con que Amstrad, en modelos posteriores, ha añadido nuevos comandos al Basic, mejorándolo bastante. Surge, por tanto, en estas personas el lógico deseo de disfrutar, ellos también, de los nuevos comandos. Esto resulta muy sencillo utilizando como herramienta las Extensiones Residentes del Sistema, en inglés RSX *(Resident System Extension)*.



J. Igual

El Basic **Amstrad** del CPC464 es bastante peculiar. En algunas áreas de la programación tiene unas prestaciones francamente excepcionales *(como su habilidad de manejar las interrupciones, permitiendo simular multitarea)*, mientras que otras falla notablemente, como en el caso de los gráficos.

A nivel del firmware, del sistema operativo del ordenador, el panorama es mucho más promisorio: existen multitud de rutinas aprovechables para crear nuevos comandos Basic que se encuentran plenamente documentadas en el manual de Firmware.

La respuesta está en el firmware

Estos comandos adicionales se escriben como subrutinas en código máquina, las cuales pueden estar situadas en cualquier zona de la memoria RAM y tener la longitud que queramos.

Los nombres y las direcciones de ejecución de las rutinas se almacenan en una tabla, acerca de la cual el sistema operativo debe ser informado, de modo que pueda utilizarla para reconocer los nuevos comandos cuando se ejecute un programa Basic que haga uso de ellos.

Como comentábamos anteriormente, estos comandos se denominan RSX y deben ir precedidos del signo «**|**» para que el sistema sepa a qué atenerse.

El programa 1 añade 7 nuevos comandos al Basic del CPC464. Su sintaxis y función serán descritas más tarde; primero analizaremos cómo están construidos y cómo funcionan.

¿Cómo se crea un nuevo comando Basic?

Todos los dialectos de Basic tienen un método para llamar a una subrutina escrita en código máquina. En el **Amstrad** toma la forma del comando CALL, el cual puede ir seguido, opcionalmente, de una serie de parámetros, además de la dirección de comienzo de la subrutina en máquina. La existencia de estos parámetros dotan de gran potencia al comando CALL, ya que pueden ser usados por la rutina en máquina y, además, ésta puede también devolver resultados de su acción y atesorarlos en variables Basic, estipuladas por nosotros.

Los parámetros se colocan detrás de la dirección de llamada, separados por comas:

CALL dirección, parámetro1, parámetro 2, ...

y, cuando Basic ejecuta CALL, el bloqueo entero de parámetros se ubica en una zona, la que sea, en memoria RAM.

La dirección concreta de esta zona no nos interesa, ya que, previsoramente, el sistema operativo coloca en el registro A *(acumulador)*, el número de parámetros pasados a la rutina desde el comando CALL, y, lo más importante, el registro IX apuntando al comienzo de la zona de memoria donde están almacenados los valores. Si meditamos en esto un poco, concluiremos en que manipular todo esto es un juego de niños.

Los parámetros pueden obtenerse utilizando direccionamiento indexado con el registro IX. Por ejemplo:

LD B, (IX+0)

cargará el registro B con el primer ítem del bloque de parámetros *(sólo el byte bajo)*.

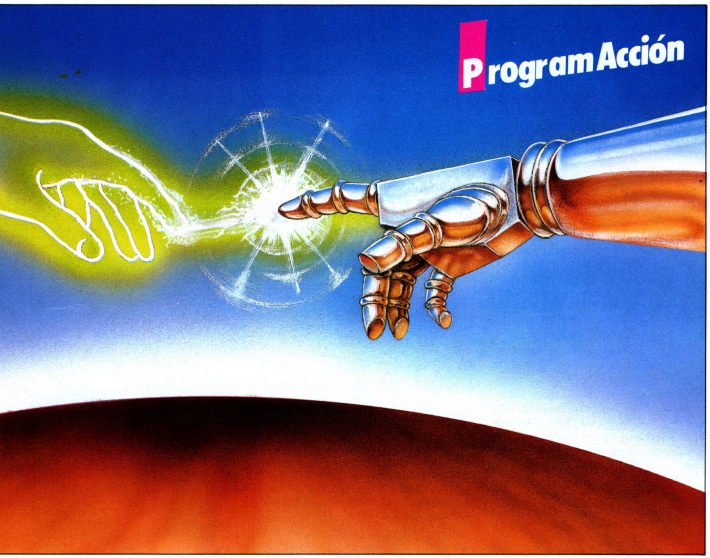
Estos pueden ser expresiones enteras, reales o la dirección de una variable. Un número de dos bytes se coloca en el bloque por cada parámetro asignado al CALL.

Un número entero, variable o expresión, se pasa al bloque de parámetros directamente; un número real, variable o expresión, primero se convierte a un valor entero, y luego se pasa.

Colocando el signo «**@**» antes de una variable, la dirección donde se encuentra el valor de dicha variable *(no el valor, insisto)*, se coloca en el bloque de parámetros.

Unos cuantos ejemplos

De esta forma podemos poner en esa variable el valor que sea resultado de la acción de la rutina en máquina.



Por ejemplo:

```
a%=10
CALL &8000,@a%
```

pondría en el bloque de parámetros la dirección de la variable a%, y la siguiente secuencia de instrucciones en lenguaje máquina, carga en el acumulador el valor numérico de dicha variable:

```
LD L, (IX+0)
LD H, (IX+1)
LD A, (HL)
```

Al final de estas instrucciones, el acumulador contendrá 10. Supongamos que a continuación se ejecutan otras instruccione, las que sean, que transforman el valor del acumulador a 5. Para colocar este número en la variable a%, habría que hacer lo siguiente:

```
LD L, (IX+O)
LD H, (IX+1)
LD (HL), A
```

admitiendo que el registro IX no ha sido alterado por la acción de la rutina y que la instrucción de vuelta al Basic RET ha sido incluida en un lugar apropiado, por ejemplo al final de la serie de órdenes antedichas.

Todas las cadenas alfanuméricas deben ser pasadas precisamente de esta forma, a través del operador «@». De todas formas, en el caso de cadenas de caracteres el tema es algo más complejo, porque lo que aparece en el bloque de parámetros es un «descriptor de cadena» que consta de tres bytes: el primero indica la longitud de la cadena, y los otros dos la dirección verdadera donde se encuentran ubicados los caracteres. Para hacer lo que antes se mostró con la variable a%, habría que trabajárselo un poquito más, como por ejemplo:

```
LD B, (IX+O): Longitud de la cadena al registro b.
LD L, (IX+2)
LD H, (IX+3): HL apunta al primer caracter de la cadena.
LD A, (HL): El acumulador contiene el primer caracter de la cadena.
```

La operación inversa se la dejamos a nuestros lectores.

A la inversa también funciona

Por alguna razón desconocida, Basic coloca los valores en el bloque de parámetros en orden inverso, es decir:

```
CALL &8000,a,b,c,d
```

al ejecutarse, dejará el registro IX apuntando al parámetro d:

IX apunta ad	2 bytes	*(baja/alto)*
IX+2 apunta ac	2 bytes	*(baja/alto)*
IX+4 apunta ab	2 bytes	*(baja/alto)*
IX+6 apunta aa	2 bytes	*(baja/alto)*
Total: 8 bytes ocupados por el bloque de parámetros *(de 0 a 7)*		

Obsérvese que las secuencias de órdenes dadas anteriormente para la variable a% y la cadena de caracteres sólo son válidas si hay un único parámetro después del comando CALL, o bien se trata del último parámetro. No obstante, conociendo el método que acabamos de explicar, adaptarlo adecuadamente no constituye mayor problema.

La ventaja de los comandos significativos

Una orden como CALL &A000, no tiene mucho sentido que digamos. Obviamente, sería mucho mejor que, si la rutina &A000 borra la pantalla, por ejemplo, pudiéramos escribir simplemente en el ordenador **«borra-pantalla»**, y se ejecutará.

Esta es exactamente la facilidad que un RSX proporciona, llamando a una rutina en máquina por nombre, como un comando Basic más.

Para poder hacer esto, debemos crear una **«tabla externa de comandos»**, haciéndole saber al sistema operativo como se llaman las rutinas y dónde están situadas. Esta dividida en otras dos, que no necesariamente deben residir cerca una de otra en memoria.

La primera mitad es una tabla de salto a las direcciones de ejecución a las rutinas, la segunda mitad contiene los nombres de las mismas.

TABLA N.º UNO

```
.tabla-rsx
DEFW tabla-
nombres          ; puntero a la lista de
                 ; nombres
JP reset         ; tabla de salto
JP set-clock
JP gpen
;
;
. tabla-nombres
DEFS 'RESE'      ; a la última letra (T),
                 ; se le ha sumado
DEFB &D4         ; &80, resultando el
                 ; nuevo número en
                 ; &D4
DEFS 'SETCLO'
DEFB &CB
;
;
DEFB 0
.esp-trabajo
DEFS 4
```

Atención a la tabla número 1. Los dos primeros bytes contienen la dirección de la lista de nombres, y al código ASCII de la última letra de cada nombre se le ha sumado &80 para que el sistema sepa dónde acaba.

El final de la lista de nombres se señala con un 0. A continuación se dejan 4 bytes como espacio de trabajo requeridos por el sistema operativo.

Para indicarle a éste la presencia de RSX, el registro BC debe cargarse con la dirección de la tabla, HL con la dirección donde comienzan los 4 bytes del espacio de trabajo y..., sólo queda llamar a una providencial rutina del firmware, llamada:

KL-LOG-EXT

que comienza en la dirección &BCD1 y se encuentra completamente explicada en el manual del firmware.

El programa 1

Los 9 primeros bytes del código máquina del programa 1 se muestran en la tabla número 2, y realizan la función que acabamos de comentar. Por tanto, después de ejecutado el programa 1, CALL &8000 activa la tabla RSX y los nuevos comandos *(el resto del programa 1)*.

Cuidado con esto porque las tablas RSX están ligadas y puede haber más de un activa a un tiempo, y si habilitamos la misma dos veces, el sistema operativo se volverá completamente loco.

En efecto, «ambas» están ligadas, pero ocupan el mismo espacio en la memoria. Resultado: system crash.

Explicar completamente paso a paso el código máquina del programa 1, sería terriblemente largo, pero sí podemos analizar como está construido el más simple de todos los nuevos comandos, **|**GPAPER, que puede usarse para definir el color de la pantalla gráfica *(Tabla número 3)*.

TABLA N.º DOS

```
LD BC, tabla-rsx  ; código ensamblado
                  ; en &8000
LD HL, esp-trabajo
JP &BCD1          ; dile al sistema que
                  ; hay en activo
                  ; una RSX y RETorna
                  ; al Basic
```

Paso a paso

Es muy simple: hemos decidido que esta orden sólo puede tener un parámetro, el color. Por tanto, lo primero que hace la rutina es DECrementar el acumulador, aprovechando la circunstancia de que en él se almacena el número de parámetros que siguen a **|**GPAPER. Si, después de DECrementar, el valor del acumulador no es cero, saltamos a una rutina de manejo de errores *(JP NZ, error)* porque hemos dado más de un parámetro.

TABLA N.º TRES

```
.gpaper
DEC A            ; ¿un solo paráme-
                 ; tro?
JP NZ, error     ; si hay más de
                 ; uno, salta a la
                 ; rutina
                 ; de; manejo de
                 ; errores
LD A, (IX+0)     ; obten color
JP &BBE4         ; llamada a la rutina
                 ; firmware GRA-SET-
                 ; PAPER
                 ; pon el color y
                 ; RETorna a Basic
```

Si, por el contrario, el acumulador contiene cero, JP NZ no se ejecutará, y saltamos a la dirección del firmware &BBE4 *(JP &BBE4)* que colocará el color adecuadamente después de haber introducido el valor en el acumulador [LD A, (IX+0)].

Y ya está. Así de fácil.

PROGRAMAS

programa numero 1

```
10 REM PROGRAMA I
20 REM By R.A.Waddilove
30 REM(c)Amstrad Semanal
40 sum=0:address=&8000
50 FOR i=1 TO 36
60 READ code$
70 FOR j=1 TO 25 STEP 2
80 byte=VAL("&"+MID$(code$,j,2))
90 POKE address,byte
100 sum=sum+byte:address=address+1
110 NEXT
120 NEXT
130 IF sum<>50246 THEN PRINT"Error"
140 END
150 DATA 0109B021B5B0C3D1BC2F80C3E6
160 DATA B0C3A180C3C6B0C3DCB0C30381
170 DATA C315B1C326B1000000000000000
180 DATA 0000000000000005245534 5D4
190 DATA 534554434C4F43CB475045CE47
200 DATA 50415045D24745D4464C5553CB
210 DATA 46494CCC0000000000000000000
220 DATA 0000000000000000000000000000
230 DATA 0000000000000000000000000000
240 DATA 0000000000000000000000000000
250 DATA 0000000000000002195B07ECD5A
260 DATA BB23A720FBC9525358206577 2
270 DATA 6F720D0A0021000011000A7CA
280 DATA 10BD3D2009DD6E00DD6601C310
290 DATA BDDD5E00DD5601DD6E02DD6603
300 DATA C310BDFE02C2B98 0DD7E00CDDE
310 DATA BB3E17CD5ABBDD7E02C35ABB3D
320 DATA C2B9B0DD7E00C3E4BBCD06B9F5
330 DATA CD37BDCD00BBCDFFBBCD65BCCD
340 DATA A7BCCD5ABBCD4EBBF1C30CB93D
350 DATA C2B9B0CD06BBDD6E00DD660177
360 DATA 233600C93DC2B98 0DD7E00A7C2
370 DATA A7BCCD09BB3BFBC9FE03C2B9B0
380 DATA DD7E00CDDEBBDD6E02DD6603DD
390 DATA 5E04DD5605D5ED53C7B122C9B1
400 DATA CDF0BB32CB813CCDE4BBCD11BC
410 DATA 3E043B063E022B023E0132CCB1
420 DATA CD95B1ED42CD7BB12BF6D1ED53
430 DATA C7B1CD95B109CD7BB12BF7C922
440 DATA C781EB2AC9B1CDF0BB21CBB1BE
450 DATA C9ED5BC7B1E5CDF0BB21CBB1BE
460 DATA E1C92AC9B12323CD87B128F92B
470 DATA 2BE52AC9B12B2BCD87B128F923
480 DATA 23ED5BC7B1CDC0BBE1ED5BC7B1
490 DATA CDF6BB2AC7B1AFED4BCCB147C9
500 DATA 00000000000000000000000000
```

programa numero 2

```
10 REM PROGRAMA II
20 DEFINT a,i,r,x,y
30 x=320:y=100:r=100
40 MODE 0:INK 0,0:BORDER 0
50 PRINT:PRINT SPC(5)"POLIGONOS"
60 PRINT:PRINT "  Cuantos lados ?"
70 PRINT:PRINT SPC(5)"( 4 - 9 )"
80 !FLUSH,0:a=0
90 WHILE a<52 OR a>57:!GET,@a:WEND
100 DEG
110 MOVE x+r,y
120 !GPEN,0,1
130 FOR i=0 TO 360 STEP 360\(a-48)
140 DRAW x+r*COS(i),y+r*SIN(i)
150 NEXT
160 !FILL,x,y,1+INT(RND*15)
170 !SETCLOCK
180 WHILE TIME<900:WEND
190 RUN
```

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.

Todos los nuevos comandos se explican en la Tabla 4, y el programa 2 muestra una pequeña aplicación de algunos de ellos.

En el programa 1 se pueden observar una serie de bloques de ceros, espacio reservado para que cada uno añada sus propias rutinas, nombres y saltos.

Para hacer esto, situar un JP al comienzo de su propia rutina en la dirección &8020, y poner el nombre (mayúsculas, con el código ASCII de la última letra+&80) en &8052.

El nuevo código máquina puede situarse donde sea, a partir de &81CD hacia abajo.

¿Alguién podría crear **|**CIRCULO, *x,y,r (coordenadas del centro y radio)*?

Hay sitio para 5 comandos más.

TABLA N.º CUATRO

|RESET	Ningún parámetro. Resetea el ordenador pero retiene el programa basic, si lo hay.
|SETCLOCK [,bajo [,alto]]	Inicializa el reloj a cero si no hay parámetros. Lo coloca a «bajo» si sólo hay un parámetro y es menor que 65536. Lo coloca a bajo+65536*alto si hay 2 parámetros.
|GPEN opción, color	Asigna la pluma para gráficos y la opción de «ploteado».
|GPAPER	Asigna el color del papel de gráficos.
|GET, @variable-entera%	Espera a que se pulse una tecla y deja su código ASCII en variable-entera%.
|FLUSH, buffer	Vacía el buffer de sonido si el parámetro es igual a 1. Vacía el de teclado si es igual a 0.
|FILL,x,y,color	Llena de color una figura. El origen debe estar en 0,0.

DRAGONTORC

Morac, tocado por la fuerza mágica de Merlyn, se erige en el único ser que puede enfrentarse al terrible poder de la hechicera Morag, de las tierras del norte de Europa y que pretende adueñarse del poder de Dargontorc.

Habían transcurrido muchos años desde que Maroc derrotó al señor del kaos y le expulsó de su reino terrenal, liberando su alma sepultada con el hechizo de Avalon Wraithbane.

El cetro del poder y el anillo del sirviente, son los únicos recuerdos que conserva Maroc de este terrible encuentro.

Desde entonces decidió alejarse de los asuntos terrenales, adoptando la vida de un nómada en los gigantescos bosques, donde los viejos magos aún habitan.

El reino de Britania es acosado por cientos de oscuras fuerzas, desde que las legiones romanas abandonaron el intento de conquista.

Vortigern, Lord de los cinco reinos de Britania hace un intento desesperado por mantener los cinco reinos unidos contra el ataque de las ordas bárbaras, que navegan por el Mar del Norte en busca de nuevos dominios.

Como último recurso, Vortigern manda una flota sajona al encuentro de los bárbaros intentando defender las costas del este.

Durante varios años los sajones defendieron con fiereza el reino británico, hasta que la terrible Morag, bruja del norte, con sus hechizos consiguió destruir su lealtad al reino británico.

En una gran fiesta celebrada por los sajones en honor de Vortigern y sus señores de la guerra, sus anfitriones se tornaron en traidores y asesinaron al rey y a todos sus seguidores.

La noche de los cuchillos largos, sumió en un completo caos a los cinco reinos y los sajones comenzaron su plan de conquista uno por uno.

A Morag la temible hechicera, no le interesaban ni los británicos ni los sajones, su plan era recobrar las cinco coronas del Dragontorc de Avalon, para hacerse con ayuda de ellas con la llama del poder.

Sus planes comenzaron a tener éxito apoderándose de la primera parte de la corona de Vortigern, la del reino de Dumnovia.

Habían transcurrido pocas noches desde la de los cuchillos largos cuando Maroc, meditando frente a las llamas de su hoguera, fue tocado por una extraña fuerza, de las llamas surgieron extrañas formas y una débil voz resonó en sus oídos.

No advirtiendo presencia humana alguna, se dispuso a dormir con la sensación de ser observado por alguien. En las largas horas de la noche, antes de la salida del sol, por fin se rindió ante la extraña sensación, recogió sus escasas pertenencias y comenzó a caminar.

Durante muchos días viajó descansando pocas horas, día a día, la extraña sensación crecía, Maroc no sabía hacia dónde se dirigía, sólo sabía que la fuerza le guiaba.

Viajó hasta llegar a un claro en un tupido bosque, donde sintió el poder que emanaba del tronco de un árbol milenario, los últimos días del otoño cubrían de hojas el suelo.

Compatible CPC 464/CPC 664/CPC 6128

Buscando entre las hojas caídas, su mano encontró un objeto metálico que emitía vibraciones caloríficas, inequívoco signo de la energía.

Mirando fijamente al extraño artefacto, comprendió que era la magia que le había llevado hasta allí.

Este era el signo del poder de Merlyn, el último de los grandes magos, guardián de la herencia del dragón. Al empuñar el emblema del gran dragón, el poder pasó a través de su brazo, apoderándose de su sistema nervioso, hasta que sintió su cuerpo flotar en un lecho de aire caliente.

De nuevo escuchó la voz, reconociendo en ella la voz de Merlyn, su maestro e iniciador en el mundo de la magia.

¡Por fin Maroc!, **«tú eres el único que puede ayudarme, escucha...».** Relatándole la terrible muerte de Morac concluyendo: **«Durante muchos años he sido prisionero de Morac.»** La voz de Merlyn se desvanecía y sus últimas palabras eran un signo inequívoco de la proximidad de la muerte. **«Encuentra el cetro de la ley.»** Maroc pudo escuchar el crujido de las hojas, percibiendo un aire helado y súbito que llenaba el bosque, sus rodillas temblaron y calló al suelo, mientras permanecía tumbado recobrándose de la sacudida repentinamente, se asombró de lo que yacía enfrente de él.

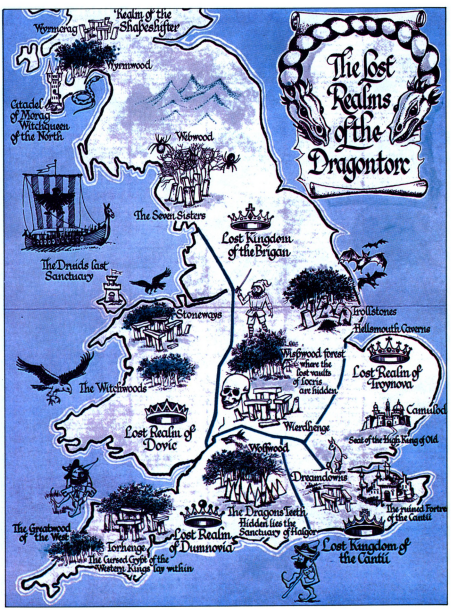
Desde este momento nos hemos apoderado de la proyección astral de Maroc, conduciéndole por la foresta en busca del secreto de Dragontorc. Como mago discípulo de Merlyn, podemos utilizar hechizos y atacar a los distintos seres que nos atacan, nuestro camino a través de bosques y abruptos páramos, está jalonado por inimaginables peligros que sólo la magia podrá eludir.

Mr. Joystick

Debemos tener cuidado con la gente que encontraremos, pues nuestros enemigos pueden hacerse pasar por amigos y alejarnos de nuestra meta.

Merlyn nos ayudará en nuestra larga búsqueda, sus mensajes están encerrados en objetos de energía.

Ser un gran mago requiere largos años de aprendizaje, no te desanimes si en unas horas no consigues salvar a Merlyn.



MATCH DAY

UN AUTENTICO PARTIDO DE FUTBOL EN TU ORDENADOR. REGATEAR, DRIBLA, CABECEAR, CENTRAR, CHUTA... Y TRATA DE METER GOL SI ES QUE TE DEJA LA DEFENSA. CUIDADO TAMBIEN CON EL CONTRATAQUE INTENTA ROBAR EL BALON O HAZ QUE TU PORTERO LO PARE. TODA LA EMOCION DE UNA FINAL EN TUS MANOS.

ERBE Software

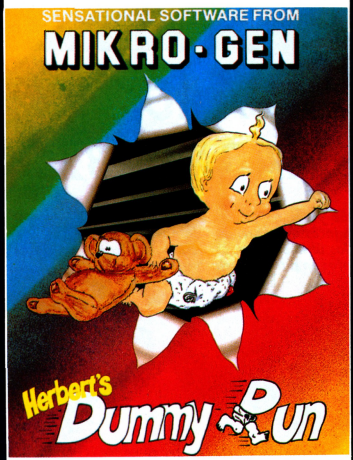
¡¡DALE MARCHA

DRAGONTORC

UNA PELICULA EN TU AMSTRAD. EN LA INGLATERRA MEDIEVAL HAS DE CONSEGUIR REUNIR LAS CORONAS QUE MORAG EL PERVERSO HA ESCONDIDO POR TODO EL PAIS. A TRAVES DE TU VIAJE PODRAS OBTENER PODERES MAGICOS QUE TE AYUDEN EN LA LUCHA. TODO EL JUEGO ESTA REALIZADO CON UN AUTENTICO EFECTO TRIDIMENSIONAL Y UN NUEVO TIPO DE MOVIMIENTO QUE HACE VER A LOS PERSONAJES COMO SI DE UNA PELICULA SE TRATARA.

BASEBALL

CONVIERTETE EN EL CAMPEON DE ESTE FANTASTICO DEPORTE. MAGNIFICOS GRAFICOS QUE INCLUYEN PANTALLA DE VIDEO GIGANTE PARA SEGUIR LAS JUGADAS DE CERCA. NO INPORTA QUE NO HAYAS JUGADO NUNCA, BASEBALL TE CONVERTIRA EN EL N.º1



DUMMY RUN

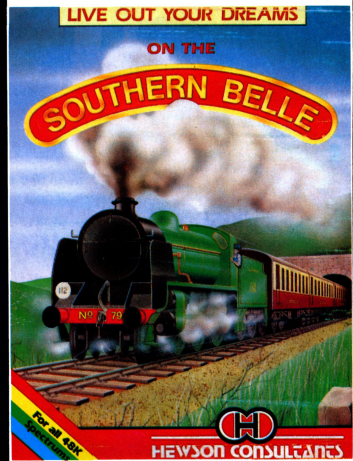
POR FIN EN AMSTRAD LAS AVENTURAS DEL PEQUEÑO DE LA FAMILIA WALLY. UN JUEGO QUE TE SORPRENDERA POR LA GRAN CANTIDAD Y CALIDAD DE SUS GRAFICOS. TODO PUEDE PASAR EN ESTE FANTASTICO PROGRAMA QUE SUPERA LOS LIMITES DE LA IMAGINACION Y QUE SE HA CONVERTIDO EN UN CLASICO DE LOS JUEGOS DE ORDENADOR.

A TU AMSTRAD!!

ERBE Software

RAID

DEFIENDE A U.S.A. y CANADA DEL ATAQUE NUCLEAR QUE HA LANZADO RUSIA CONTRA ELLOS.
AL MANDO DE TU ESCUADRILLA HABRAS DE HACER UN VIAJE LLENO DE PELIGROS HASTA EL MISMISIMO KREMLIN Y DESTRUIR LAS BASES DE LANZAMIENTO SOVIETICAS. ACCION A TOPE.



SOUTHERN BELLE

¡PASAJEROS AL TREN! Y TU A LOS MANDOS. PREPARATE PARA REALIZAR UN VIAJE EN "LA BELLA DEL SUR" LA LOCOMOTORA QUE FUE FAMOSA A PRINCIPIOS DE SIGLO. CONTROLARAS TODOS LOS INSTRUMENTOS DE ABORDO: PRESION, DIRECCION, FRENOS, CALDERA, PARADAS, TRANSBORDOS... RESUMIENDO TODO LO QUE HA DE HACER UN CONDUCTOR DE TREN INCLUYENDO TOCAR EL SILBATO.

PROBAMOS EL AMSTRAD PCW8256

El PCW8256 ya no es noticia: es una tangible realidad. Dentro de la incesante corriente de novedades lanzada por Amstrad en los últimos tiempos, este equipo corresponde ya a un enfoque pura y exclusivamente profesional.

La máquina encaja perfectamente en el concepto **«llave en mano»**, es decir, se proporciona de una sola vez las herramientas necesarias para empezar a trabajar a pleno rendimiento con el equipo, sin necesidad de adquirir programas o periféricos extras.

Un equipo completo

El conjunto incluye los siguientes componentes:

1. Monitor fósforo verde (90 columnas×32 filas).

2. Unidad de disco de tres pulgadas integrada en el monitor *(la misma que en los modelos CPC)*.

3. Teclado profesional, tipo máquina de escribir, especialmente diseñado para el proceso de textos en castellano.

4. 256 Kbytes de memoria RAM, esto es, cuatro veces la memoria del **Amstrad** CPC464.

5. Impresora matricial controlada por programa.

Esto en cuanto al hadware. En lo que respecta al software, **Amstrad** entrega con el PCW8256 los siguientes programas:

a) Sistema operativo CP/M Plus *(versión 3.1)*, incluyendo una serie de utilidades para los programadores y otras para el usuario no interesado en la programación, como por ejemplo el sistema GSX para manejo de gráficos en pantalla, impresora o plotter.

b) Lenguaje Basic, un intérprete llamado Mallard Basic compatible casi en su totalidad con el Basic de Microsoft.

c) Una implementación de lenguaje Logo creada por Digital Research, el DR Logo.

d) Un procesador de textos, creado por Locomotive, llamado Locoscript, y que merece una mención aparte por su potencia y sencillez de uso de cara al usuario no iniciado.

El conjunto de hardware y software, según información suministrada por Indescopm, distribuidora oficial de **Amstrad** en nuestro país, costará al usuario final 169.900 ptas.

¿Quién va a usar el PCW8256?

¿Y quién es el usuario final? ¿A qué tipo de personas está dirigido este sistema?

Bien, una primera pista la suministra el nombre del ordenador; PCW significa, en inglés, Personal Computer and Wordprocessor, es decir, ordenador personal y procesador de textos. Esto huele a gestión de negocios desde lejos, y efectivamente es así. A ello responde el diseño del teclado, que incorpora funciones tales como la búsqueda y cambio de palabras en una sola tecla, y el Locoscript, el programa de proceso de textos.

Banco de pruebas

Desde luego, sería una lástima emplear una máquina con 256 K de RAM exclusivamente en proceso de textos, aunque ésta sea su función **«principal»**. **Amstrad** así lo ha comprendido, por lo que ha dotado a su ordenador de un sistema operativo standard, el CP/M Plus. Esto implica fundamentalmente dos cosas:

— Poder usar cualquier programa que funcione bajo ese sistema operativo *(Si está disponible en formato de 3 pulgadas)*.

— El PCW8256 queda convertido, por arte del CP/M, en un ordenador de **«propósito general»**, extremadamente potente para ser un **«8 bits»**, y sus campo de aplicación abarca un espectro muy amplio: desde proceso de texto a hojas de cálculo y bases de datos, pasando por todo tipo de lenguajes y utilidades de programación, hasta aplicaciones que permitan comunicación con grandes ordenadores a través de modems (cuando la infraestructura necesaria exista en España).

Conclusión: el PCW8256 está, sorprendentemente, dirigido a todo el mundo, pero con la siguiente prioridad:

— negocios a corto plazo;

— programación y desarrollo de sistemas a medio plazo;

— entretenimiento y juegos a muy largo plazo.

¿Revolución?

¿Y cuáles son los límites de este paradisíaco panorama, si los hay? O, ¿podemos empezar a usar palabras tales como revolución y milagro a la hora de hablar de la máquina?

En mi opinión, existen límites. El primero, hasta qué punto el PCW8256 estará dotado en breve plazo de aplicaciones que permitan aprovechar sus cualidades para algo más que el proceso de textos.

El segundo, el precio. No cabe duda de que todo el paquete, al precio de 169.900 ptas., es una cantidad más que razonable, sobre todo para una empresa, por pequeña que sea.

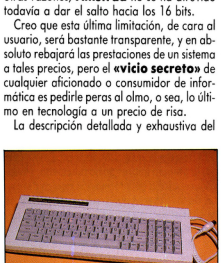
Sin embargo, ese dinero, vox populi, es «una pasta», y queda por ver el veredicto emitido por los profesionales liberales y por los simples aficionados que deseen tener un equipo potente, compacto y versátil como éste.

En cuanto a que el PCW8256 sea una revolución, también en mi opinión depende del punto de vista con que se le juzgue: a pesar de las **«dudas»** expresadas más arriba, respecto al precio pienso que sí; es auténticamente revolucionario. No obstante, la tecnología empleada en el ordenador no lo es, mucho más en un mundo donde empiezan a reinar los ordenadores de 16 bits.

¡Cuidado!, esto no implica que el PCW8256 funcione mal o sea **«un cacharro»**, sino simplemente que, bien en aras del coste o por otras razones, **Amstrad** no se ha atrevido todavía a dar el salto hacia los 16 bits.

Creo que esta última limitación, de cara al usuario, será bastante transparente, y en absoluto rebajará las prestaciones de un sistema a tales precios, pero el **«vicio secreto»** de cualquier aficionado o consumidor de informática es pedirle peras al olmo, o sea, lo último en tecnología a un precio de risa.

La descripción detallada y exhaustiva del PCW8256 requeriría una extensión de texto prohibitiva, por lo que vamos a comentar lo que nos ha parecido más relevante del ordenador, siempre teniendo en cuenta que esto ha sido solamente una **«toma de contacto»** con el ordenador y lo que le acompaña.



Punto por punto

El monitor, como puede verse en las fotos, es de unas dimensiones respetables, bastante mayor que el de sus antecesores. Tiene una excelente resolución, y el texto se ve en él con toda claridad y nitidez, sin desagradables **«temblequeos»**. Posee un único mando, el control de brillo, y más de 80 caracteres por línea, concretamente 90 por línea y 32 líneas, es decir, un 35 por 100 más de superficie visual que un IBM, por ejemplo.

En él se encuentra integrada la unidad de disco, sólo que colocada en sentido vertical. La operación de introducir y extraer discos resulta fácil y cómoda, y uno se acostumbra rápidamente a distinguir cuál es la cara 1, la 2 de un disco colocado en vertical. Por lo demás, es idéntica a la de los modelos CPC en cuanto a capacidad. Debajo se encuentra un espacio dedicado a una segunda unidad de disco cubierto con una tapa. Esta segunda unidad tendría una capacidad de un Megabyte, sin formatear, dando 720 K por cara al ser formateada. Es de doble cara, doble densidad o sea, el PCW puede acceder a la información de ambas caras del disco sin tener que sacarlo de la unidad.

En la parte posterior del monitor se encuentra el enchufe de la impresora y la interface Centronics, que también se conecta a la impresora.

El teclado se conecta a la extensión situada en la parte derecha del monitor y es, como se dijo antes, tipo máquina de escribir, un teclado profesional que llaman. Se puede separar a una distancia aceptable del monitor, detalle que se echa en falta en otros modelos **Amstrad**, y el tacto y respuesta de las teclas es excelente.

En su parte posterior parece tener unos orificios preparados para colocarle unas **«patitas»**, que permitan elevarlo, pero el equipo que estuvo en nuestras manos no las tenía. Esto no representa un problema serio a la hora de teclear, no obstante hay que tener cuidado con la orientación del teclado con respecto a la luz, porque las teclas, al ser cóncavas,

reflejan la luz de la habitación y brillan de forma bastante molesta *(por este detalle suponemos que los orificios traseros están provistos para incorporar soportes que aumenten la elevación del teclado)*.

También mentamos que está preparado para el proceso de textos en castellano, pues posee la ñ y las vocales acentuadas pueden obtenerse mediante determinadas combinaciones de teclas. Estas están distribuidas en un solo bloque, con la suficiente separación entre ellas para que sea difícil pulsar dos a la vez accidentalmente.

Del Hardware, sólo queda hablar un poco de la impresora. Es de tamaño y peso reducido, muy manejable, y no posee un solo mando de los habituales en las impresoras excepto el que permite mover manualmente el papel. Todo lo demás, avance de línea, de página, etc., se realiza por programa, bien bajo Locoscript o bajo Basic.

Por supuesto, es de matriz de puntos (¡el coste!), no de margarita, y posee un juego completo de tipos de letra (según Indescomp, unos 400), también controlados por programa, además de distintas «densidades» de impresión y tamaño de letra *(incluyendo el famoso espaciado proporcional)*.

Está dotada de dos velocidades de impresión: la normal, 90 caracteres por segundo, y la de **«calidad carta»**, 20 caracteres por segundo. Esta última denominación es un tanto optimista, comparada con lo que produce una impresora de margarita, pero puede servir. También se podría objetar que la velocidad de impresión **«normal»** no es nada del otro mundo, pero hay que tener en cuenta que, bajo Locoscript, el procesador de textos, podemos editar y corregir un documento mientras imprimimos otro, con lo que no se notará demora a la hora de imprimir excepto en situaciones excepcionales. La impresora puede manejar papel continuo, gracias a su mecanismo tractor, así como hojas sueltas. Para este propósito está dotada de un sistema de carga automática de papel.

EL Software

Amstrad entrega con el PCW8256 2 discos: en uno de ellos se encuentra el CP/M Plus, sistema operativo que, en parte por ser bastante conocido y en parte por la complejidad que entraña, demanda un estudio más profundo y un espacio más amplio del que podemos dedicarle aquí. Baste decir que incluye, bastantes mejoradas, las utilidades necesarias para la manipulación de ficheros en disco y otras más dedicadas a los programadores, como ensambladores, librerías, etc.

El Logo de Digital Research también es conocido de nuestros lectores, porque hay versiones implementadas para disco en el 664 y en el 6128. Sospechamos que la revisión del PCW8256 es prácticamente análoga a la del 6128, y digo sospechamos porque el espacio dedicado a él en los manuales es exiguo y la explicación pobre *(existen en el lenguaje comandos que ni siquiera aparecen en el manual)*, al menos en la versión inglesa que cayó en nuestras manos. Aprovecho para aclarar aquí que todos los manuales y Locoscript, siempre según Indescomp, estarán traducidos cuando el equipo llegue al comprador.

El intérprete de Basic, el Mallard Basic, en un dialecto clónico del Basic Microsoft, o sea, dedicado a la gestión y manejo de ficheros en disco. Nada de comandos de gráficos, ni de sonido, ni de ventanas.

El Basic de Mallard incorpora tratamiento de ficheros secuenciales y aleatorios: normal. Pero también posee una facultad extremadamente rara en ordenadores de este tipo: el manejo y creación de ficheros indexados, es decir, regidos por claves de acceso a cada campo. En el manual se habla de que este tipo de ficheros están pensados para permitir incluso aplicaciones multiusuario. Lo que está claro es que los ficheros indexados constituyen un método de almacenar la información muy potente y flexible. Bien por el Basic.

A propósito, el Basic **si** que se explica con gran detalle en el manual, comando por comando y con abundantes ejemplos.

La vedette del PCW

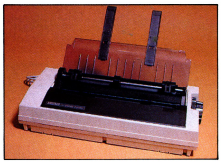
En cuanto al software, queda por hablar del programa que, sin duda, es la estrella de todos los que se dan con el ordenador y la razón de ser del mismo: el Locoscrit, uno de los procesadores de texto de mayor potencia que hemos visto hasta ahora, incluyendo muchos de los que corren en ordenadores de 16 bits como el IBM PC.

Locoscript se carga en memoria por el sencillo procedimiento de encender el ordenador e introducir en el disco que contiene el programa.

No es necesario llamar al CP/M ni nada parecido.

Debo admitir que, cuando el programa cargó, la primera impresión no fue buena. El monitor rebosaba de información acerca del disco, de los ficheros que había en él y de otras muchas cosas sin aparente sentido. Mis esperanzas de poder manejar Locoscript, al menos un poco, sin recurrir al manual se desvanecieron como el humo.

Sin embargo, para ser justo, y después de superada la decepción inicial, debo reconocer que el manual es necesario, no sólo por



la cantidad de posibilidades y opciones que Locoscript ofrece, sino porque está estructurado con gran habilidad, introduciendo al usuario novato en el uso del programa paso a paso, de lo más sencillo a lo más complejo, y con tal abundancia y detalle en los ejemplos, que es prácticamente imposible equivocarse.

En fin, el manual es extenso, pero por una vez, con motivo justificado. Imagino que la versión española, si no experimenta las habituales **«dificultades»** en la traducción, aún será más sencilla. Esperemos que Indescomp se supere a sí misma y nos sorprenda suministrando unos manuales del PCW8256 y de Locoscript medianamente legibles, en contra de lo que tiene por costumbre.

Pensando en el usuario

Para mayor facilidad, el disco de Locoscript incluye ficheros de texto que se usan, en el manual, a modo de ejemplo constantemente. Con ellos podemos practicar todas las funciones del programa, hasta la más avanzadas, desde el primer momento.

Tanto si nuestros lectores conocen las funciones **«standard»** de un procesador de textos, como si no, podemos asegurar que con Locoscript y un documento puede hacerse de todo, absolutamente de todo lo que se le pue-

FICHA DEL PCW8256

HARDWARE. Incluye:
- Monitor monocromo.
- Teclado y unidad central con 256 K RAM.
- Unidad de disco 169 K formateada.
- Impresora.
- Microprocesador Z80.

SOFTWARE. Incluye:

Programa	Facilidad de uso	Potencia
CP/M plus	regular	buena
CP/M utilidades	regular	?
LOGO	excelente	buena
Mallard Basic	buena	excelente
Locoscript	excelente	excelente

COMERCIALIZADO POR: **Indescomp.**
P.V.P.: **169.900 ptas.**

da ocurrir. El trío ordenador, programa e impresora se compenetran a la perfección.

Locoscript está regido por una estructura jerárquica de menús, a los cuales se accede normalmente pulsando las techas de función del teclado, cada una asociada en la pantalla a su correspondiente etiqueta.

El menú principal es el llamado **«gestión del disco»**, en el cual tenemos las opciones típicas: editar un documento, crearlo, imprimirlo directamente, y manipular los ficheros de disco, esto es, borrar, cambiar de nombre, mover, etc., junto con la curiosa opción de **«impresión directa»**, o sea, usar la impresora como una máquina de escribir.

Partiendo de aquí, por una maraña de submenús, accedemos al resto de las opciones. Por destacar alguna, podemos elegir el tipo de letra para cada sección de texto, podemos subrayar palabras, diseñar distintos formatos para un documento o parte del mismo, etc.

No podemos describirlas todas.

Siempre que se elige una opción, aparece una ventana en la pantalla donde se encuentra la **«sugerencia»** por defecto del programa *(normalmente acierta)* respecto a lo que queremos hacer.

Por ejemplo, si yo cambio la paginación de un documento, he terminado y quiero salir, Locoscript presenta una ventana en la que se encuentran una serie de posibilidades de salida, entre las cuales está, naturalmente, la de conservar y fijar la paginación que acabo de elegir para ese documento. Pues bien, ésta aparece iluminada; simplemente pulsando ENTER, la nueva paginación queda aceptada.

Si por alguna razón cambio de idea y quiero anularla o hacer otra cosa, basta tan sólo desplazarse por la ventana con las teclas del cursor, elegir la opción que sea y pulsar ENTER. Es así de fácil con todas las cosas que se pueden hacer con Locoscript, algunas de ellas verdaderamente complejas.

Menús o comandos: como usted guste

Se ha tenido en cuenta al usuario experimentado, aquel que no necesita recorrer menú tras menú de opciones para hacer las cosas porque lleva ya tiempo usando el programa, de dos formas igualmente ingeniosas: resumiendo todas las opciones importantes en dos pequeñas ventanas, sin agruparlas por categorías, e incluso suprimiendo los menús completamente, porque cada opción tiene una combinación de letras, un comando, que la identifica, por lo que si la incluimos en un texto, el programa sabe que se trata de un comando y obra en consecuencia.

Para elegir el documento con él que queremos trabajar, nos desplazamos por la pantalla de «gestión de disco» con los cursores hasta llegar al que interesa, que aparecerá en vídeo inverso. Luego, pulsar ENTER, y el texto es traído del disco inmediatamente a la memoria.

El programa hace un uso eficiente de la memoria del PCW8256, porque reserva y usa 112 K como disco virtual, es decir, es como si tuviéramos una segunda unidad de disco en el ordenador, con la ventaja de que las operaciones de lectura/escritura son muchísimo más rápidas, y la desventaja de que si hay un corte de luz, pues, perdemos lo almacenado en el disco virtual.

El manual recomienda, con toda razón, el uso de esta zona como área transitoria de documentos y/o datos solamente.

Esta técnica del disco virtual también es usada por CP/M, y los programas CP/M probablemente serán capaces de utilizarla.

El control de la impresora, a través de Locoscript, es total: podemos alterar tipos de letra, tamaño de página, pie y cabecera, calidad de letra, impresión o no y lugar de impresión del número de página, reimpresión de un documento a partir de determinada página. etc. Sin problemas.

A mí, personalmente, el programa me ha impresionado. Creo que aprovecha al máximo las posibilidades del PCW8256 y permite todo tipo de libertades con cualquier documento de cualquier extensión, aunque después de dos días de manejarlo resulta difícil hacerse una idea exacta de las posibilidades de este programa, dada la potencia que exhibe.

Cuando estábamos acabando de escribir este artículo, Indescomp presentó oficialmente a los medios de comunicación el PCW8256, e informó de una serie de nuevas características que no pudimos observar antes por poseer la versión inglesa del ordenador, y programas prototipo en lugar de la versión definitiva.

Así, nuestro Locoscript tenía fallos; concretamente, al realizar una operación en la que se definía la densidad de escritura, el ordenador se bloqueaba siempre, tanto con copias del programa como con el disco original.

Tuvimos la ocasión de ver que:

— La versión definitiva de Locoscript en castellano, completamente traducida, tanto manuales como el de los propios mensajes del programa, esta libre de errores, al menos de éste. Indescomp nos aseguró que ha sido exhaustivamente revisada y depurada por representantes suyos altamente cualificados en colaboración con Locomotive, creadora de Locoscript.

— El CP/M Plus y el DR Logo también han sido traducidos al castellano, al igual que los manuales y toda la documentación del ordenador.

Nota: para comentar la facilidad de uso y la potencia de las utilidades de programación de CP/M, nos hemos basado en nuestro conocimiento del CP/M Plus. Los manuales de ***Amstrad*** *no dan información suficiente sobre ellas para juzgarlo adecuadamente. Se limitan a remitir al lector a un libro publicado al efecto, por lo que sabemos no disponible en España.*

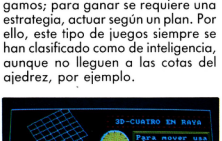
CUATRO EN RAYA 3D

COMPATIBLE
CPC 464
CPC 664
CPC 6128

Mucho me temo que este juego apenas necesite presentación. Pocas personas habrá que en **«pubes»** o sitios parecidos no hayan visto un tablero de madera agujereado, puesto en posición vertical, con unas ranuras en su parte superior por la que se introducen fichas de colores.

El juego consiste en colocar cuatro fichas, normalmente del mismo color, en **«raya»**, bien sea horizontal, vertical, o en diagonal.

Esto no es demasiado fácil que digamos; para ganar se requiere una estrategia, actuar según un plan. Por ello, este tipo de juegos siempre se han clasificado como de inteligencia, aunque no lleguen a las cotas del ajedrez, por ejemplo.



La simulación de juegos de **«pensar»** en ordenadores, siempre ha tenido el doble atractivo de jugar con un compañero infatigable e infinitamente paciente, y de estudiar, desde el punto de vista del programador, cómo puede modelizarse una estrategia o método de previsión en una máquina.

Así que, tanto el jugador curioso, como el programador ávido de aprender a simular **«inteligencia»**, encontrarán materia de estudio más que sobrada en este programa.

Por otra parte, el autor del juego lo ha complicado de veras: ha construido una estructura tridimensional formada por cuatro tableros. Jugar contra el ordenador no es fácil, sobre todo en los últimos niveles de los que dispone *(9)*.

La presentación del programa, a mi juicio, es impecable, y su manejo se explica por sí mismo: todas las instrucciones necesarias están en pantalla.

Desde el punto de vista del programador, yo destacaría las rutinas en las que el programa decide su movimiento, valorando la posición, y aquellas en las que se maneja los dos relojes que cuentan el tiempo transcurrido entre cada jugada, uno para el jugador humano y otro para la máquina.

```
10 MODE 1:CLS
20 PEN 3:PAPER 0:PRINT CHR$(22)+CHR
$(0);:PRINT CHR$(23)+CHR$(0);
30 DIM XY(144,2),TABLE(361,4),GRID(
144)
40 RANDOMIZE TIME:CLS
60 GOSUB 2100:REM DEFINE CARACTERES
70 PRINT CHR$(23)+CHR$(0)
80 PRINT CHR$(22)+CHR$(0):CLS
100 top=1:Xorigin=40:Yorigin=240:GO
SUB 1840:REM dibuja la cuadricula s
uperior
110 top=0:Xorigin=84:Yorigin=152:GO
SUB 1840:REM dibuja la segunda cuad
ricula
120 Xorigin=130:Yorigin=64:GOSUB 18
40:REM dibuja la tercera cuadricula
130 Xorigin=176:Yorigin=24:GOSUB 1
840:REM dibuja la cuadricula inferi
or
150 ORIGIN 0,0:GOSUB 1150
160 PEN 1:LOCATE 26,22:PRINT"ELIGE
NIVEL:";:PEN 3
170 A$=INKEY$:IF A$="" OR (VAL(A$)<
1) OR (VAL(A$)>9) THEN GOTO 170
180 PRINT A$:LEVEL=VAL(A$)
190 LOCATE 29,24:PRINT"NIVEL: "+A$
200 FOR I=1 TO 2000:NEXT:GOSUB 720
210 PEN 2:LOCATE 27,22:PRINT "DESEA
S MOVER":LOCATE 27,23:PRINT"PRIMERO
? S/N"
220 A$=INKEY$:A$=UPPER$(A$):IF A$="
" OR (A$<>"S" AND A$<>"N") THEN GOT
O 220
230 IF A$="S" THEN START=0:PEN 3:LO
CATE 28,24:PRINT " ** SI **" ELSE S
TART=1:PEN 3:LOCATE 28,24:PRINT " *
* NO **"
240 FOR I=1 TO 2000:NEXT:GOSUB 720
250 EXIT=0:CLOCK=1:SEC=1:MIN=0
280 X=334:Y=290:CX=19:CY=12:MM=3:GO
SUB 4200
290 X=40:Y=200:CX=2:CY=17:MM=3:GOSU
B 4200
300 EVERY 49,1 GOSUB 4160
310 WHILE EXIT=0
320 IF START=1 THEN START=255:GOTO
350
330 GOSUB 3000:REM tu jugada
340 IF EXIT=1 THEN GOTO 360
350 GOSUB 3410:REM jugada del orden
ador
360 WEND
370 PRINT CHR$(22)+CHR$(0)
380 IF SCORE>HISCORE THEN HISCORE=S
CORE:SCORE=0:PEN 2:LOCATE 10,19:PRI
NT USING "###";HISCORE:LOCATE 10,21
:PRINT USING "###";SCORE
390 PEN 2:LOCATE 27,22:PRINT "OTRO
JUEGO"
400 LOCATE 30,23:PRINT"(S/N)?"
410 A$=INKEY$:A$=UPPER$(A$):IF A$="
" OR (A$<>"S" AND A$<>"N") THEN GOT
O 410
420 IF A$="S" THEN PEN 3:LOCATE 28,
24:PRINT " ** SI **" ELSE PEN 3:LOC
ATE 28,24:PRINT " ** NO **"
430 FOR I=1 TO 2000:NEXT:GOSUB 720
440 IF A$="S" THEN GOSUB 3340:GOTO
160:REM limpia la pantalla para otr
a partida
450 GOTO 2410
460 REM chequeo para ver si hay gan
ador
490 FINISHED=0:CLOCK=0
510 PEN:PRINT CHR$(22)+CHR$(0);:LOCA
TE 28,22:PEN 2:PRINT" CHEQUEO ":EI
:COUNT%=1
530 WHILE FINISHED=0 AND COUNT%<360
540 IF GRID(TABLE(COUNT%,1))=PLAYER
 AND GRID(TABLE(COUNT%,2))=PLAYER A
ND GRID(TABLE(COUNT%,3))=PLAYER AND
 GRID(TABLE(COUNT%,4))=PLAYER THEN
FINISHED=1:EXIT =1 ELSE COUNT%=COUN
T%+1
550 WEND
560 IF FINISHED<>1 THEN &B0
570 CLOCK=0
580 TAGOFF:PRINT CHR$(22)+CHR$(0);
590 IF FINISHED=1 AND PLAYER=COMPUT
ER THEN PEN 3:LOCATE 27,21:PRINT"**
**********":LOCATE 27,22:PRINT"*
          *":LOCATE 27,23:PRINT"******
******":PEN 2:LOCATE 29,22:PRINT"YO
 GANO!"
600 IF FINISHED=1 AND PLAYER=HUMAN
```

```
THEN PEN 3:LOCATE 27,21:PRINT"*****
*******":LOCATE 27,22:PRINT"*
    *":LOCATE 27,23:PRINT"*********
***":PEN 2:LOCATE 29,22:PRINT"TU GA
NAS"
610 FOR I=1 TO 2000:NEXT I:GOSUB 72
0
620 PRINT CHR$(23)+CHR$(1);
630 SEC=HSEC+1:IF SEC>59 THEN SEC=0
640 MTN=MTN:X=334:Y=290:CX=19:CY=1
2:MM=2:GOSUB 4200
650 SEC=CSEC+1:IF SEC>59 THEN SEC=0
660 MTN=CMTN:X=60:Y=200:CX=2:CY=17:
MM=3:GOSUB 4200
670 CSEC=0:HSEC=0:CMTN=0:HMTN=0
680 IF FINISHED=0 THEN CLOCK=1
690 RETURN
700 REM limpia la ventana de mensaj
es
720 TAGOFF:PRINT CHR$(22)+CHR$(0);
730 LOCATE 26,21:PRINT STRING$(14,"
")
740 LOCATE 26,22:PRINT STRING$(14,"
")
750 LOCATE 26,23:PRINT STRING$(14,"
")
760 LOCATE 26,24:PRINT STRING$(14,"
");
770 LOCATE 26,21
780 PRINT CHR$(22)+CHR$(1);
790 RETURN
800 REM obtiene y analiza la jugada
 devolviendo un valor de 1-16 en la
 variable NUM
820 NUM=1
830 TAGOFF
840 PRINT CHR$(22)+CHR$(1);
850 PRINT CHR$(23)+CHR$(1);
860 DI:MOVE -1,-1:DRAWR 0,0,1
870 FOR U=0 TO 10B STEP 36:NN=U+NUM

890 TAG:MOVE XY(NN,1),XY(NN,2):PRIN
T CURS$;:MOVE XY(NN,1),XY(NN,2)-16:
PRINT CURS1$;:TAGOFF:NEXT U
910 EI
920 A$=INKEY$:IF A$="" THEN 920
930 DI:MOVE -1,-1:DRAWR 0,0,1
940 FOR U=0 TO 10B STEP 36:NN=U+NUM

960 TAG:MOVE XY(NN,1),XY(NN,2):PRIN
T CURS$;:MOVE XY(NN,1),XY(NN,2)-16:
PRINT CURS1$;:TAGOFF:NEXT U
980 EI
990 IF A$=CHR$(&F3) THEN NUM=NUM-1
1000 IF A$=CHR$(&F2) THEN NUM=NUM+1
1010 IF A$=CHR$(&F1) THEN NUM=NUM+6
1020 IF A$=CHR$(&F0) THEN NUM=NUM-6
1030 IF A$=CHR$(&E0) THEN GOTO 1130
1040 IF NUM>36 THEN NUM=NUM-36
1050 IF NUM < 1 THEN NUM=NUM+36
1060 DI:MOVE -1,-1:DRAWR 0,0,1
1070 FOR U= 0 TO 10B STEP 36:NN=U+N
UM
1090 TAG:MOVE XY(NN,1),XY(NN,2):PRI
NT CURS$;:MOVE XY(NN,1),XY(NN,2)-16
:PRINT CURS1$;:TAGOFF:NEXT U
1110 EI
1120 GOTO 920
1130 TAGOFF:RETURN
1140 REM dibuja pantalla
1150 ORIGIN 0,0
1160 LOCATE 20,2:PEN 3:PRINT STRING
$(21,"*")
1170 LOCATE 20,3:PRINT "*":LOCATE 4
0,3:PRINT "*"
1180 LOCATE 20,4:PRINT STRING$(21,"
*")
1190 PAPER 0
1200 LOCATE 22,3:PEN 1:PRINT "3D-CU
ATRO EN RAYA"
1210 MOVE 390,10:DRAW 390,330,2
1220 DRAW 630,330:DRAW 630,10:DRAW
390,10
1230 MOVE 394,14:DRAW 394,326,2
1240 DRAW 626,326:DRAW 626,14:DRAW
394,14
1250 MOVE 394,80:DRAW 626,80
1260 MOVE 394,84:DRAW 626,84
1270 MOVE 0,0:DRAW 0,399,1:DRAW 639
,399:DRAW 639,0:DRAW 0,0
1280 PEN 1:LOCATE 26,6:PRINT"Para m
over usa"
1290 LOCATE 27,7:PRINT"los cursores
"
1300 PRINT CHR$(22)+CHR$(1)
1310 PEN 3:LOCATE 26,6:PRINT"______
_____"
1320 LOCATE 27,7:PRINT"___________
_____"
1330 ZONE 1
1340 SCORE=0:HISCORE=0
1350 PRINT CHR$(22)+CHR$(0)
1360 PEN 2:LOCATE 30,9:PRINT"SUBIR"

1370 LOCATE 30,19:PRINT"BAJAR"
1380 LOCATE 38,13:PRINT " D":LOCATE
 38,14:PRINT " E":LOCATE 38,15:PRIN
T" R"
1390 LOCATE 26,13:PRINT"I"+CHR$(8)+
CHR$(10)+"Z"+CHR$(8)+CHR$(10)+"D"
1400 PEN 1:LOCATE 31,14:PRINT"COPY"
1410 PEN 3
1420 LOCATE 28,14:PRINT CHR$(242)+C
HR$(154)
1430 LOCATE 36,14:PRINT CHR$(154)+C
HR$(243):PEN 3
1440 PEN 3
1450 LOCATE 32,11:PRINT CHR$(240)+C
HR$(8)+CHR$(10)+CHR$(149)
1460 LOCATE 32,16:PRINT CHR$(149)+C
HR$(8)+CHR$(10)+CHR$(241)
1470 PEN 2:LOCATE 3,23:PRINT"HUMAND
 ";:PEN 1:PRINT CHR$(242)+CHR$(243)
:PEN 2:PRINT " "+CHR$(143)
1480 PEN 2:LOCATE 2,24:PRINT"MAQUIN
A ";:PEN 1:PRINT CHR$(242)+CHR$(243
);:PEN 3:PRINT " "+CHR$(143);
1490 PEN 1:LOCATE 2,19:PRINT "RECOR
D:";LOCATE 10,19:PRINT USING "###";
HISCORE
1500 LOCATE 2,21:PRINT "PUNTOS:";LO
CATE 10,21:PRINT USING "###";SCORE
1510 PRINT CHR$(22)+CHR$(1)
1520 X=334:Y=290:R=45:C=1:GOSUB 410
0
1530 X=334:Y=290:R=41:C=2:GOSUB 410
0
1540 FOR I=0 TO 60 STEP 5:MOVE X+(
SIN(I*PI/30)*R-5)),Y+(COS(I*PI/30)
*(R-5)):DRAW X+(SIN(I*PI/30)*R),Y+(
COS(I*PI/30)*R),3:NEXT I
1580 X=60:Y=200:R=45:C=3:GOSUB 4100
1590 X=60:Y=200:R=41:C=3:GOSUB 4100
1600 FOR I=0 TO 60 STEP 5:MOVE X+(
SIN(I*PI/30)*R-5)),Y+(COS(I*PI/30)
*(R-5)):DRAW X+(SIN(I*PI/30)*R),Y+(
COS(I*PI/30)*R),1:NEXT I
1640 PRINT CHR$(23)+CHR$(1):R=35
1660 HSEC=0:HMTN=0:CSEC=0:CMTN=0
1670 RETURN
1680 REM dibuja la pieza del jugado
r en su sitio correspondiente
1710 TAGOFF:DI
1720 IF PLAYER=HUMAN THEN COL=2 ELS
E COL=3
1730 PRINT CHR$(22)+CHR$(1);:PRINT
CHR$(23+COL);
1740 MOVE -1,-1:DRAWR 0,0,COL
1750 IF (I>=37 AND I<=41) OR (I>=73
 AND I<=77) OR (I>=109 AND I<=113)
THEN S$=EDGE$:S1$=EDGE1$:GOTO 1770
1760 S$=BLK$:S1$=BLK1$
1770 TAG:MOVE XY(I,1),XY(I,2)+PRINT
 S$;:MOVE XY(I,1),XY(I,2)-16:PRINT
S1$;:TAGOFF:EI
1780 RETURN
1800 REM dibuja cada cuadricula en
la posicion obtenida de Xorigin e Y
origin
1840 ORIGIN Xorigin,Yorigin:X=2.8:Y
=2.8
1850 X=2.8:Y=2.8
1860 MOVE X+10,Y+40
1870 DRAW X+5,Y+38,1:DRAW X+29,Y+7:
DRAW X+75,Y+28:DRAW X+52,Y+56
1880 MOVE X+45,Y+52:DRAW X+67,Y+25
```


```
1890 MOVE X*3B,Y*49:DRAW X*60,Y*21.
5
1900 MOVE X*31,Y*46:DRAW X*53,Y*18
1910 MOVE X*23,Y*42:DRAW X*45,Y*14.
5
1920 MOVE X*15,Y*39:DRAW X*37,Y*11
1930 MOVE X*9,Y*33:DRAW X*53,Y*53
1940 MOVE X*12,Y*2B:DRAW X*57,Y*4B
1950 MOVE X*17,Y*23:DRAW X*62,Y*45
1960 MOVE X*20,Y*1B:DRAW X*66,Y*3B
1970 MOVE X*25,Y*13:DRAW X*70,Y*33
1980 IF topo$="MEN GOTO 2050
1990 MOVE X*10,Y*41:DRAW X*51,Y*59:
DRAW X*53,Y*54
2000 MOVE X*45,Y*52:DRAW X*43,Y*55
2010 MOVE X*3B,Y*49:DRAW X*36,Y*51
2020 MOVE X*31,Y*46:DRAW X*29,Y*4B
2030 MOVE X*23,Y*42:DRAW X*21,Y*45
2040 MOVE X*15,Y*39:DRAW X*13,Y*41
2050 DRAWR 0,0,3
2060 RETURN
2070 REM inicializa los caracteres
definidos por el usuario,las conden
adas X-Y,introduce la combinacion y
ganadora en la matriz,llena la mat
riz que representan las cuadriculas
 con ceros e inicializa el contador
 de sprites
2100 SYMBOL AFTER 179
2110 SYMBOL 180,0,0,0,0,3,15,15,31
2120 SYMBOL 181,15,15,15,3,0,0,0,0
2130 SYMBOL 182,0,0,0,0,0,0,192,192,2
24
2140 SYMBOL 183,224,192,192,0,0,0,0
,0
2150 SYMBOL 184,0,0,0,0,0,0,0,0
2160 SYMBOL 185,7,15,15,3,0,0,0,0
2170 SYMBOL 186,0,0,0,0,0,0,192,224
2180 SYMBOL 187,0,0,0,1,7,31,127,25
5
2190 SYMBOL 188,255,127,63,31,14,0,
0,0
2200 SYMBOL 189,0,0,0,192,224,240,2
48,252
2210 SYMBOL 190,252,248,224,128,0,0
,0,0
2220 CURS$=CHR$(187)+CHR$(189):CURS
1$=CHR$(188)+CHR$(190)
2230 BLK$=CHR$(180)+CHR$(182):BLK1$
=CHR$(181)+CHR$(183)
2240 EDGE$=CHR$(184)+CHR$(186):EDGE
1$=CHR$(185)+CHR$(183)
2250 FOR I=1 TO 144:READ XY(I,1),XY
(I,2):NEXT I
2260 ZONE 10
2270 INK 0,0:BORDER 27:INK 1,26:INK
 2,24:INK 3,6
2320 FOR I%=1 TO 54:READ TABLE(I%,1
),TABLE(I%,2),TABLE(I%,3),TABLE(I%,
4):NEXT I%
2330 FOR I%=1 TO 54:TABLE(I%+54,1)=
TABLE(I%,1)+36:TABLE(I%+54,2)=TABLE
(I%,2)+36:TABLE(I%+54,3)=TABLE(I%,3
)+36:TABLE(I%+54,4)=TABLE(I%,4)+36:
NEXT I%
2340 FOR I%=1 TO 54:TABLE(I%+108,1)
=TABLE(I%,1)+72:TABLE(I%+108,2)=TAB
LE(I%,2)+72:TABLE(I%+108,3)=TABLE(I
%,3)+72:TABLE(I%+108,4)=TABLE(I%,4)
+72:NEXT I%
2350 FOR I%=1 TO 54:TABLE(I%+162,1)
=TABLE(I%,1)+108:TABLE(I%+162,2)=TA
BLE(I%,2)+108:TABLE(I%+162,3)=TABLE
(I%,3)+108:TABLE(I%+162,4)=TABLE(I%
,4)+108:NEXT I%
2360 FOR I%= 217 TO 360:READ TABLE(
I%,1),TABLE(I%,2),TABLE(I%,3),TABLE
(I%,4):NEXT I%
2370 FOR I%=1 TO 144:GRID(I%)=0:NEX
T I%
2380 HUMAN=10:COMPUTER=20:CLOCK=0

2400 RETURN
2410 TAGOFF
2420 PRINT CHR$(22)+CHR$(0)
2430 PRINT CHR$(23)+CHR$(0)
2440 INK 1,26:PEN 1:MODE 2
2460 END
2470 RETURN
2540 REM
2600 REM datas conteniendo todas la
s cordenadas X-Y que permiten coloc
ar las piezas en cualquier posicion

2650 DATA 162,407,142,398,122,389,1
01,380,79,370,58,360,174,394,153,38
4,133,374,112,365,90,355,68,345,186
,379,165,370,145,361,125,351,102,34
1,80,331,49,321
2660 DATA 177,356,157,346,135,337,1
12,327,91,317,209,351,189,342,168,3
33,147,323,125,313,102,303,220,336,
200,327,179,318,159,308,136,298,114
,288
2670 DATA 206,318,186,310,166,300,1
45,290,125,280,102,271,218,306,197,
296,177,286,156,276,135,267,113,258
,220,297,199,287,179,277,169,263,14
7,254,124,243
2680 DATA 242,277,221,268,201,258,1
80,249,159,239,134,229,255,263,233,
254,212,245,191,235,168,225,146,215
,264,249,244,239,223,230,202,220,18
0,211,158,200
2690 DATA 253,231,232,222,212,212,1
91,202,170,193,147,183,264,218,244,
208,222,199,201,180,181,179,159,170
,276,203,255,193,235,184,215,176,19
3,165,170,155
2700 DATA 207,189,267,180,247,170,2
26,161,204,151,181,141,299,175,279,
166,259,157,237,147,215,137,192,127
,310,161,352,152,269,142,248,132,22
6,122,203,112
2710 DATA 298,142,278,134,258,124,2
37,114,215,105,190,181,179,159,170
,20,269,111,248,102,226,124,204,82,
32
2,115,302,106,281,97,260,87,238,77,
216,67,333,101
2720 DATA 313,92,293,82,272,73,249,
63,227,53,345,87,325,78,303,69,282,
59,260,49,238,39,356,72,336,63,316,
54,294,44,272,34,252,24
2730 REM todas las combinaciones ga
nadoras posibles
2760 DATA 1,2,3,4,2,3,4,5,3,4,5,6,7
,8,9,10,8,9,10,11,12,13,14,
15,16,14,15,16,17,15,16,17,18,19,20
,21,22,20,21,22,23,21,22,23,24,25,2
6,27,28,26,27,28,29
2770 DATA 27,28,29,30,31,32,33,34,3
2,33,34,35,33,34,35,36
2780 DATA 1,7,13,19,7,13,19,25,13,1
9,25,31,2,8,14,20,8,14,20,26,14,20,
26,32,3,9,15,21,9,15,21,27,15,21,27
,33,4,10,16,22,10,16,22,28,16,22,28
,34,5,11,17,23
2790 DATA 7,13,19,25,13,19,25,31,8,
14,20,26,14,20,26,32,9,15,21,27,15,
21,27,33,4,10,16,22,10,16,22,28,16,
22,28,34,5,11,17,23
2800 DATA 18,23,28,33,12,17,22,27,1
7,22,27,32,16,21,26,31,15,21,26,1,6
,21,26,31,5,10,15,20,10,15,20,25,4
,9,14,19,13,20,27,34,7,14,21,28,14,
21,28,35
2810 DATA 1,8,15,22,8,15,22,29,15,2
2,29,36,2,9,16,23,9,16,23,30,3,10,1
7,24
2820 DATA 1,37,73,109,2,38,74,110,3
,39,75,111,4,40,76,112,5,41,77,113,
6,42,78,114,7,43,79,115,8,44,80,116
,9,45,81,117,17,53,89,125,18,54,90,
119,12,48,84,120
2830 DATA 13,49,85,121,14,50,86,122
,15,51,87,123,16,52,88,124,17,53,89
,125,18,54,90,126,19,55,91,127,20,5
6,92,128,21,57,93,129,22,58,94,130,
23,59,95,131
2840 DATA 24,60,96,132,25,61,97,133
,26,62,98,134,27,63,99,135,28,64,10
0,136,29,65,101,137,30,66,102,138,3
1,67,103,139
2850 DATA 32,68,104,140,33,69,105,1
41,34,70,106,142,35,71,107,143,36,7
2,108,144
2860 DATA 1,38,75,112,2,39,76,113,3
,40,77,114,4,39,74,109,5,40,75,110,
6,41,76,111,7,44,81,116,8,45,82,119
,9,46,83,120,10,45,80,115,11,46,81,
116,12,47,82,117
2870 DATA 13,50,87,124,14,51,88,125
,15,52,89,126,16,51,86,121,17,52,87
,122,18,53,88,123,19,56,93,130,20,5
7,94,131,21,56,95,132,22,57,92,127,
23,58,93,128
2880 DATA 24,59,94,129,25,62,99,136
,26,63,100,137,27,64,101,138,28,63,
98,133,29,64,99,134,30,65,100,135,3
1,68,105,142,32,69,106,143,33,70,10
7,144
2890 DATA 34,69,104,139,35,70,105,1
40,36,71,106,141
2900 DATA 1,43,85,127,2,44,86,128,3
,45,87,129,4,46,88,130,5,47,89,131,
6,48,90,132,7,49,91,133,8,50,92,134
,9,51,93,135,10,52,94,136,11,53,95,
137,12,54,96,138
2910 DATA 13,55,97,139,14,56,98,140
,15,57,99,141,16,58,100,142,17,59,1
01,143,18,60,102,144,19,49,79,109,2
0,50,80,110,21,51,81,111,22,52,82,1
12,23,53,83,113
2920 DATA 24,54,84,114,25,55,85,115
,26,56,86,116,27,57,87,117,28,58,88
,118,29,59,89,119,30,60,90,120,31,6
1,91,121,32,62,92,122,33,63,93,123,
34,64,94,124
2930 DATA 35,65,95,125,36,66,96,126
2940 DATA 1,44,87,130,2,45,88,131,3
,46,89,132,4,45,86,127,5,46,87,128,
6,47,88,129,7,50,93,136,8,51,94,137
,9,52,95,138,10,53,96,139,11,54,97,
134,12,53,94,135
2950 DATA 13,56,99,142,14,57,100,14
3,15,58,101,144,16,57,98,139,17,58,
99,140,18,59,100,141,19,50,81,112,2
0,51,52,113,21,52,83,114,22,51,80,1
09,23,53,82,110
2960 DATA 24,53,82,111,25,56,87,118
,26,57,88,119,27,58,89,120,28,57,86
,115,29,58,87,116,30,59,88,117,31,6
2,93,124,32,63,94,125,33,64,95,126,
34,63,92,121
2970 DATA 35,64,93,122,36,65,94,123
2980 REM esta rutina procesa nuestr
a jugada y la devuelve solo si es v
alida
3000 FINISHED=0
3010 DI
3020 H=HSEC+1+MTN+MTN:IF SEC>59
THEN SEC=0:MTN=MTN+1
3030 PLAYER=HUMAN
3040 EI
3050 TAGOFF
3060 WHILE FINISHED=0
3070 DI
3080 PRINT CHR$(22)+CHR$(0);
3090 PEN 3:LOCATE 27,21:PRINT"*****
******":LOCATE 27,22:PRINT"*
          *":LOCATE 27,23:PRI
NT"*****  *****":PEN 2:LOCATE 30,22
:PRINT"ILEGAL" ELSE FINISHED=1:GOTO
 3150
3100 FOR I=1 TO 2000:NEXT:GOSUB 720
:GOTO 3090
3150 PLAYER=HUMAN
3160 TAGOFF:PRINT CHR$(22)+CHR$(1);
3170 X=MUM:GOSUB 1710
3180 IF NUM=36 >144 THEN GOTO 3200
3190 IF GRID(NUM+36)=0 THEN I=NUM:8
OSUB 1710:NUM=NUM+36:GOTO 3170
3200 WEND
3210 FOR T%=1 TO 10:SOUND 1,60,10,1
5:FOR DELAY=1 TO 250:NEXT DELAY:SOS
UB 1710:NEXT T%
3220 GRID(NUM)=HUMAN
3230 PLAYER=HUMAN:GOSUB 490:GOSUB 7
20
3250 TAGOFF
```

```
3260 IF NUM>=109 AND NUM<=144 THEN
SCORE=SCORE+LEVEL
3270 IF NUM>=73 AND NUM<=108 THEN S
CORE=SCORE+(2*LEVEL)
3280 IF NUM>=37 AND NUM<=72 THEN SC
ORE=SCORE+(3*LEVEL)
3290 IF NUM>=1 AND NUM<=36 THEN SCO
RE=SCORE+(4*LEVEL)
3300 DI:PRINT CHR$(22)+CHR$(0):LOCA
TE 10,21:PRINT USING "###";SCORE:PR
INT CHR$(22)+CHR$(1);:EI
3310 RETURN
3340 FOR I=1 TO 144:IF GRID(I) <> 0
 THEN PLAYER=GRID(I):GRID(I)=0:GOSU
B 1710
3350 NEXT I
3370 RETURN
3380 REM rutina de jugada de la maq
uina segun nivel de juego
3410 FINISHED=0
3420 DI
3430 SEC=CSEC+1:NTN=CNTN:IF SEC>59
 THEN SEC=0:NTN=NTN+1
3440 PLAYER=COMPUTER
3450 EI
3460 NUM=0:VAR=20
3480 TAGOFF
3490 PRINT CHR$(22)+CHR$(0);
3500 DI
3510 PEN 3:LOCATE 27,21:PRINT"*****
*******":LOCATE 27,22:PRINT"*
   *":LOCATE 27,23:PRINT"*********
***":PEN 2:LOCATE 29,22:PRINT"PENSA
NDO"
3520 EI
3530 WHILE VAR<>0 AND FINISHED<>1
3540 RESULT=3#VAR
3550 FOR I1%=0 TO 162 STEP 54
3560 IF INT(RND(1)#LEVEL)=0 AND VAR
=20 THEN I1%=I1%+54:GOTO 3650
3570 FOR I%=1 TO 54:NX%=I1%
3590 IF GRID(TABLE(NX,1))+GRID(TABL
E(NX,2))+GRID(TABLE(NX,3))+GRID(TAB
LE(NX,4))<> RESULT THEN GOTO 3640
3600 IF GRID(TABLE(NX,1))=VAR AND G
RID(TABLE(NX,2))=VAR AND GRID(TABLE
(NX,3))=VAR AND GRID(TABLE(NX,4))=0
 THEN FINISHED=1:NUM=TABLE(NX,4)+I%
=54:I1%=162:GOTO 3640
3610 IF GRID(TABLE(NX,2))=VAR AND G
RID(TABLE(NX,3))=VAR AND GRID(TABLE
(NX,4))=VAR AND GRID(TABLE(NX,1))=0
 THEN FINISHED=1:NUM=TABLE(NX,1)+I%
=54:I1%=162:GOTO 3640
3620 IF GRID(TABLE(NX,1))=VAR AND G
RID(TABLE(NX,2))=VAR AND GRID(TABLE
(NX,4))=VAR AND GRID(TABLE(NX,3))=0
 THEN FINISHED=1:NUM=TABLE(NX,3)+I%
=54:I1%=162:GOTO 3640
3630 IF GRID(TABLE(NX,1))=VAR AND G
RID(TABLE(NX,3))=VAR AND GRID(TABLE
(NX,4))=VAR AND GRID(TABLE(NX,2))=0
 THEN FINISHED=1:NUM=TABLE(NX,2)+I%
=54:I1%=162:GOTO 3640
3640 NEXT I%
3650 NEXT I1%
3660 IF FINISHED=1 THEN GOTO 3750
3670 FOR I1%=216 TO 324 STEP 36
3680 IF INT(RND(1)#LEVEL)=0 THEN I1
%=I1%+36:GOTO 3730
3690 FOR I%=1 TO 36:NX%=I1%+I1%
3710 IF GRID(TABLE(NX,2))=VAR AND G
RID(TABLE(NX,3))=VAR AND GRID(TABLE
(NX,4))=VAR AND GRID(TABLE(NX,1))=0
 THEN FINISHED=1:NUM=TABLE(NX,1)+I%
=36:I1%=324
3720 NEXT I%
3730 NEXT I1%
3740 VAR=VAR-10
3750 WEND
3760 TAGOFF:PRINT CHR$(23)+CHR$(1);
3770 IF FINISHED=1 THEN GOTO 3980
3780 IF LEVEL < 5 THEN GOTO 3950
3790 VAR=10
3800 WHILE VAR<>30 AND FINISHED<>1
3810 RESULT=2#VAR
3820 FOR NUMBER = 1 TO 360
3830 IF GRID(TABLE(NUMBER,1))+GRID(
TABLE(NUMBER,2))+GRID(TABLE(NUMBER,
3))+GRID(TABLE(NUMBER,4))<>RESULT T
HEN GOTO 3910
3840 IF GRID(TABLE(NUMBER,1))=VAR A
ND GRID(TABLE(NUMBER,2))=VAR THEN F
INISHED=1:NUM=TABLE(NUMBER,3):VAR=3
0:GOTO 3910
3850 IF GRID(TABLE(NUMBER,2))=VAR A
ND GRID(TABLE(NUMBER,3))=VAR THEN F
INISHED=1:NUM=TABLE(NUMBER,4):VAR=3
0:GOTO 3910
3860 IF GRID(TABLE(NUMBER,3))=VAR A
ND GRID(TABLE(NUMBER,4))=VAR THEN F
INISHED=1:NUM=TABLE(NUMBER,2):VAR=3
0:GOTO 3910
3870 IF GRID(TABLE(NUMBER,1))=VAR A
ND GRID(TABLE(NUMBER,4))=VAR THEN F
INISHED=1:NUM=TABLE(NUMBER,3):VAR=3
0:GOTO 3910
3880 IF GRID(TABLE(NUMBER,1))=VAR A
ND GRID(TABLE(NUMBER,3))=VAR THEN F
INISHED=1:NUM=TABLE(NUMBER,2):VAR=3
0:GOTO 3910
3890 IF GRID(TABLE(NUMBER,2))=VAR A
ND GRID(TABLE(NUMBER,4))=VAR THEN F
INISHED=1:NUM=TABLE(NUMBER,3):VAR=3
0
3900 IF FINISHED=1 THEN NUMBER=360
3910 NEXT NUMBER
3920 IF FINISHED=1 THEN GOTO 3940
3930 VAR=VAR+10:IF LEVEL<7 THEN VAR
=30
3940 WEND
3950 WHILE FINISHED=0
3960 NUM=INT(RND(1)*16)+1:IF GRID(N
UM)=0 THEN FINISHED=1
3970 WEND
3980 PLAYER=COMPUTER
3990 I=NUM:GOSUB 1710
4000 IF NUM<36 > 144 THEN GOTO 4020
4010 IF GRID(NUM+36)=0 THEN I=NUM:G
OSUB 1710:NUM=NUM+36:GOTO 3990
4020 FOR T%=1 TO 10:SOUND 1,30,10,1
5:FOR DELAY=1 TO 250:NEXT DELAY:GOS
UB 1710:NEXT T%
4030 GRID(NUM)=COMPUTER
4040 PLAYER=COMPUTER:GOSUB 490
4050 GOSUB 720
4060 TAGOFF
4070 RETURN
4080 rellena un circulo de coordena
das X=Y,radio "R" y color "C"
4100 FOR QX=Y-R TO Y+R STEP 2
4110 INC=INT(SQR((R*R)-((Y-QX)*(Y-Q
X)))+0.5)
4120 MOVE X-INC,QX:DRAW X+INC,QX,C
4130 NEXT QX
4140 RETURN
4150 REM moviento del reloj
4160 IF CLOCK=0 THEN GOTO 4250
4170 IF PLAYER=HUMAN THEN HSEC=SEC:
HMTN=MTN:X=334:Y=290:C=1:R=12:MM
=2 ELSE CSEC=SEC:CMTN=MTN:X=60:Y=20
0:CX=2:CY=17:MM=3
4180 TAGOFF:MOVE X,Y:DRAW X+(SIN(SE
C#PI/30)#R),Y+(COS(SEC#PI/30)#R),MM
4190 SEC=SEC+1:IF SEC>59 THEN SEC=0
:MTN=MTN+1
4200 MOVE X,Y:DRAW X+(SIN(SEC#PI/30
)#R),Y+(COS(SEC#PI/30)#R),MM
4210 ZONE 1
4220 PRINT CHR$(22)+CHR$(0);
4230 LOCATE CX,CY:PEN MM:PRINT USIN
G "##";MTN;SEC
4240 PRINT CHR$(22)+CHR$(1);
4250 RETURN
```

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

ORDENACION ALFABETICA

Análisis

Una de las principales misiones de un ordenador, es como su nombre indica la ordenación de datos.
En esta ocasión Análisis aborda el tema de la ordenación con un programa en el cual una vez introducidas todas las palabras las saca en orden alfabético.
El programa también puede usarse con números, ordenando éstos de menor a mayor, o también utilizando palabras y números a la vez, en cuyo caso aparecerán los números en orden creciente en un primer bloque y las palabras ordenadas alfabéticamente en un segundo bloque.

10,20 Nombre del programa y línea de asteriscos que lo separan del resto de instrucciones.

30 Limpia la pantalla en caso de que tubiésemos algún mensaje anterior en ella.

40 El LOCATE, coloca el cursor de texto en la décima columna de la quinta fija.

Procediendo después a situar el mensaje del INPUT, donde nos pide el número de palabras a situar, el cual es almacenado en la variable PALABRAS.

50 Dimensiona una matriz con tantos elementos como palabras deseemos ordenar.

60-100 Ciclo FOR... NEXT con tantas oscilaciones como elementos contiene la variable PALABRAS.

70 Sitúa el cursor de texto en la posición 13,13, colocando después el mensaje palabra no. indicándonos el número de la palabra que vamos a teclear.

80 Coloca una serie de espacios en blanco encima de la palabra tecleada anteriormente, produciendo su borrado.

90 Sitúa el cursor de texto debajo del mensaje palabra no. De forma que la palabra tecleada queda bajo éste, siendo almacenada en la variable alfanumérica a$ con su número de orden.

100 El NEXT cierra el ciclo.

110-150 Dos ciclos anidados que son los responsables de todo el trabajo de ordenación.

Su funcionamiento es el siguiente:

El primer ciclo toma una palabra con un número de orden t, la cual es comparada en el segundo ciclo con la palabra del siguiente número de orden, es decir t+1.

Si la t+1 es mayor que la t, el programa pasa a compararla con la siguiente palabra incrementando j en una unidad, repitiendo el proceso hasta que encuentra una palabra menor, o completa el ciclo FOR... NEXT 120-140, es decir, todas las palabras que quedan sin ordenar.

En cambio si t+1 es menor que t, el programa cambia de orden las dos palabras de forma que ahora queda delante la palabra menor *(el ordenador entiende por menor la palabra que en orden alfabético está delante).*

Completado este ciclo el programa pasa al principal 110-150, para proceder con la siguiente palabra. De esta forma las que tengan un número menor que t, quedan ordenados definitivamente, continuando el bucle la ordenación con los elementos desordenados que se encuentran detrás.

120 Compara dos elementos de forma que si el primero es menor que el segundo, almacena éste en la variable b$, pasa a la a$ (t) el valor de a$ (j) y asigna a a$ (j) el contenido de b$, consiguiendo de esta forma cambiar el orden entre las dos, de forma que ahora se encuentra delante la menor.

160 Una vez realizado todo el proceso de ordenación limpia la pantalla para proceder después al listado de todas las palabras.

170-200 FOR... NEXT, responsable de sacar por pantalla la lista de palabras ordenadas.

180 Representa la palabra de número de orden marcado por el ciclo.

190 La senencia IF, compara si el número de palabras representadas en pantalla es múltiplo de 24, en este caso aparece el mensaje ** Pulse una tecla para continuar **, el ciclo WHILE... WEND se ocupa de mantener en pantalla la lista de 24 palabras hasta que pulsemos una tecla. Pulsada ésta nos representará las siguientes 24, repitiéndose el proceso hasta acabar con toda la lista de palabras.

Para volver a ver la lista de palabras, basta con teclear: GOTO 170 (ENTER).

Y ésta aparecerá de nuevo en pantalla desde la primera hasta la última, en bloques de 24 elementos.

COMPUTER CENTER

COMANDANTE ZORITA, 13
28003 MADRID

TELS.: (91) 233 07 35
(91) 233 07 81

AMSTRAD 464 Verde	**59.900** ptas.
AMSTRAD 6128	**99.900** ptas.
DISKETTE 3"	**1.050** ptas.
INTERFACE DISCO 5,25"	**5.900** ptas.
CINTA C-15 ESPECIAL ORDEN	**85** ptas.

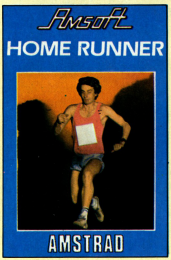
SOFTWARE ENTRETENIMIENTO (CASSETTE)

COMBAT LINX	**1.925** ptas.
EXPLODING FIST	**2.095** ptas.
ALIEN-8	**1.875** ptas.
KNIGHT LORE	**1.875** ptas.
GREMLINS	**2.095** ptas.
JUMP JET	**2.695** ptas.
MIGHWAY ENCOUNTER	**1.875** ptas.
FRUIT MACHINE	**995** ptas.
FIGHTER PILOT	**2.050** ptas.
AJEDREZ 3-D	**2.340** ptas.

LIBROS

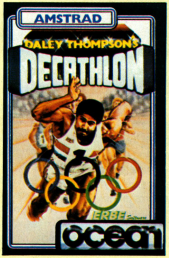
CURSO AUTODIDACTICO BASIC AMSTRAD (Contiene manual y dos cassettes)	**2.695** ptas.
HACIA LA INTELIGENCIA ARTIF.	**1.300** ptas.
40 JUEGOS EDUCATIVOS	**1.800** ptas.
MUSICA Y SONIDO PARA AMSTRAD	**1.300** ptas.
PROGRAMANDO CON AMSTRAD	**1.900** ptas.
CODIGO MAQUINA PARA PRINCIPIANTES CON AMSTRAD	**1.900** ptas.
TECNICAS DE PROGRAMACION DE GRAFICOS EN EL AMSTRAD	**1.800** ptas.

Tu pedido lo puedes recibir contra-reembolso *(libre de gastos)*, llamando a los teléfonos **(91) 233 07 35** y **(91) 233 07 81.**



1.495 Ptas.

1.975 Ptas.



1.975 Ptas.

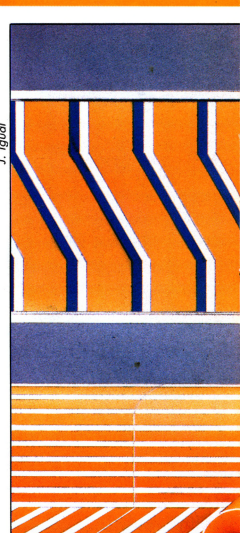
1.495 Ptas.

1.795 Ptas.

2.395 Ptas.

FLAGS

Si hay algo que se use con frecuencia en un programa, sea lenguaje máquina o no, es la toma de decisiones en función de que se cumplan ciertas condiciones. Esto, en código máquina, se hace mediante la inspección de ciertos indicadores, los flags.



J. Igual

Recuerda que es un flag? Si, un flag es un indicador que nos permite conocer el estado de las cosas en nuestros programas. Para nosotros es una guía tan vital como el radar para un piloto o el aparato de tomar la tensión para un doctor.

Pese a su importancia, los flags son en esencia bastante simples. Podemos imaginarlos como una variable numérica primitiva que sólo nos permite tener dos valores: 1 cuando el flag está activado y/o cuando esta desactivado.

Un flag, un bit

Otra forma de verlos es como un registro de un solo bit, puesto que un bit puede tomar solamente los valores 0 ó 1.

Dependiendo de lo que ocurra en un programa, el micro activa o desactiva los diversos flags. Encontramos el primer flag cuando sumamos números. Ya que el número más grande que se puede almacenar en un byte es 255, cuando tenemos dos números que juntos suman más de 255 nos encontramos con problemas.

Como hemos visto, sobrepasando 255 en un byte —*en registro o en memoria*— es algo parecido a cuando el cuentakilómetros de un coche da una vuelta completa: vuelve a comenzar desde cero.

Hasta aquí el registro A (el registro de trabajo de ADD y SUB) debe preocuparse de:

```
255+1=0
255+2=1
255+3=1
```

y así sucesivamente. Algo muy parecido ocurre cuando intentamos ir por debajo de cero en una sustracción.

Si no nos cree pruebe los programas I y II. El primero hace:

```
254+3 (&FE+&03)
```

y el segundo:

```
3 — 4 (&03 — &04)
```

En ambos casos la respuesta esta almacenada en &2FFB, el primer byte del espacio de trabajo del Hexer.

Por supuesto, el Z80 no hace caso de los resultados aberrantes cuando las cosas dan una vuelta de reloj *(se sobrepasa en un sentido o en otro el valor límite)*: activa lo que conocemos como el Carry flag. Esto es, el Carry flag se hace 1. A la inversa, si no hemos sobrepasado por encima *(o por debajo)* del límite, el flag está desactivado, a 0.

Así que, el micro se ocupa de hacer:

```
255+1=0 activa el Carry flag
254+1=255 desactiva el Carry flag
0 — 0=0 desactiva el Carry flag
3 — 4 =255 activa el Carry flag
```

El Carry flag nos permite saber cuando las cosas van mal.

El propio Amstrad se encarga

No hay forma de que, como programadores, podamos examinar un flag en particular directamente, pero el micro está bastante atento para tener en cuenta el valor de un flag.

Como hemos visto en artículos anteriores, dependiendo de que el Carry flag esté activado o desactivado, podemos saltar a diferentes partes de nuestros programas con instrucciones tales como:

```
JP C (opcode &DA)
```

que es salto con el Carry activado y:

```
JP NC (opcode &D2)
```

que es salto si el Carry flag NO está activado.

El programa III efectúa un sonido de una forma muy simple. Lo que hace es sumar dos números ?1 y ?2. Por supuesto, son etiquetas *(labels)* solamente —queremos decir que hay que meter aquí los dos bytes que necesitamos sumar— y si queremos hacer un POKE con ellos, sus posiciones de memorias son &3001, &3003, respectivamente.

Las primeras dos líneas del programa III hacen la suma. La instrucción ADD activará o desactivará el Carry flag convenientemente para avisarnos si nos hemos excedido de 255.

```
LD (&2FFB), A
```

entonces coloca el resultado en el siguiente byte del espacio de trabajo del hexer, para su examen posterior.

Atención a esto: la instrucción LD no afecta nunca a ningún flag. *(Hay solamente dos excepciones sin importancia, LD, A,R y LD A,I con las que probablemente nunca tropezaremos.)* Deja los flags sin tocar para nada. Esto es una parte de información muy a tener en cuenta por su utilidad.

Así que después de este LD, el estado del Carry dependerá del resultado del ADD.

Podemos probarlo con:

```
JP NC, &300F
```

Esto comprueba si el Carry flag está activado. Si no lo está —es decir, si nuestra solución no excede el límite del byte— el programa salta a la posición de memoria &300F, la cual es simplemente el final de las rutinas en código máquina, ya que contiene &C9, RET.

Obsérvese que por el momento saltamos avanzando e ignoraremos así algunas instrucciones máquina si el Carry no está activado.

No obstante, si el Carry está activado —o sea, que hemos excedido 255 en nuestro resultado— nos lleva directamente al siguiente bit del código máquina después de JP, que es:

```
LD A,&07
CALL CharOut
```

que, como seguramente recuerda, provoca un pitido. Entonces nos encontramos con RET.

DIRECCIÓN	OPCODE	MNEMONICO
3000	3E FE	LD A, &FE
3002	C6 03	ADD A, &03
3004	32 F8 2F	LD (&2FF8), A
3007	C9	RET

Programa 1

El resultado de todo esto es que nuestro programa será un pitido producido por las sumas que provoquen la activación del Carry flag —las que nos dan resultados **«erróneos»**— sin embargo, deja pasar las otras sumas sin problemas.

Sencillo, pero útil

Está bien, un pitido de advertencia no es algo muy impresionante, pero nos ilustra sobre cómo podemos detectar la señal del Carry flag y actuar en consecuencia. En la práctica utilizaremos esta ramificación técnica para hacer algunos cálculos aritméticos más, antes de tener la respuesta correcta.

Si experimentamos cambiando los valores sumados —*en las posiciones de memoria &3001, &3003*— veremos si tenemos el salto esperado o no.

Reemplacemos nuestro siguiente:

```
JP NC, &300F
```

por un

```
JP C, &300F
```

cambiando la cuarta línea del programa III a:

```
3007 DA OF 30 JP C, &300F
```

¡Cuidado al reemplazar con estos tres bytes los anteriores!

En este momento saltamos cuando el Carry está activado. Lo cual sólo ocurre cuando la suma tiene una solución de mayor valor que la extensión del byte, así que no se hace un **«beep»** para valores fuera de rango.

Si la solución queda dentro de los límites y el Carry es desactivado, no realizamos el salto, y escuchamos el **«bepp»**.

De modo que cambiando un solo byte, el código de operación para JP NC —&D2— al có-

DIRECCION	OPCODE	MNEMONICO
3000	3E 03	LD A, &03
3002	D6 04	SUB &03
3004	32 F8 2F	LD (&2FF8), A
3007	C9	RET

Programa 2

digo de operación para JP C —&DA— tenemos un efecto del programa completamente contrario.

De cualquier manera, lo esencial de la demostración es que cuando sumamos (ADD) dos bytes sencillos, si la solución excede de 255 el Carry flag se activa para avisarnos.

Le puede extrañar que le llamemos Carry flag *(indicador de acarreo)*. Bueno, si puede dar marcha atrás en sus recuerdos, en la escuela lo usábamos para señalar que nos **«llevábamos»** a la columnas de las decenas desde la columna de las unidades cuando hacíamos sumas ¡y lo que nos «llevábamos» era muy importante!.

Si teníamos que sumar:

```
  DU
  16
+  9
  --
```

nuestro metódico pensamiento funcionaba: 9 más 6 es 15, que es demasiado grande para la columna de las unidades, así que pondré 5 debajo de las unidades y **«llevaré»** una decena a la columna de las decenas.

Trabajábamos de un modo muy parecido al código máquina. En este momento nuestra **«columna de unidades»**, *un byte sencillo* puede contener hasta el número 255, después de esta cantidad tenemos que **«llevar»**, o añadir uno, a un byte que representa la segunda columna de nuestra suma (columna de las decenas en la escuela).

Por lo tanto, es muy útil que nuestro flag se active automáticamente cuando pasamos de 255. Precisamente cuando necesitamos «llevarnos uno» nuestro flag se activa convenientemente a uno, por eso lo llamamos Carry flag. Explicaremos exactamente cómo trabaja todo el proceso en las series posteriores.

Como hemos mencionado, el flag también se activa cuando el resultado de un cálculo está por debajo de cero. El programa IV es una variante del programa III adaptado a la substracción.

Una vez más ?1 y ?2 son solamente etiquetas, queremos decir, que colocamos nuestros propios números aquí en las posiciones de memoria &3001, &3003.

El programa trabaja del mismo modo que el programa III. Si el Carry NO se activa, el JP NC sencillamente salta a la terminación RET. Si hay un Carry, se realiza la instrucción del pitido.

Como habrá descubierto mientras experimentaba, el pitido ocurre cuando el segundo número es mayor que el primero. Es decir, cuando intentamos substraer un número mayor que el que ya está en el registro A.

Piénselo. Si el número que cogemos es mayor que el número que hemos tomado anteriormente, debemos cruzar el importante «límite cero», volviendo otra vez como un reloj y activando el Carry flag.

Lo siguiente es una demostración de lo que pasa cuando restamos de 2 números cada vez más grandes, veamos:

```
2—0=2   Carry desactivado.
2—1=1   Carry desactivado.
2—2=0   Carry desactivado.
2—3=255 Carry activado.
2—4=254 Carry activado.
```

DIRECCION	OPCODE	MNEMONICO
3000	3E ?1	LD A, ?1
3002	C6 ?2	ADD A, ?2
3004	32 F8 2F	LD (&2FF8), A
3007	D2 0F 30	JP NC, &300F
300A	3E 07	LD A, &07
300C	CD 5A BB	CALL CharOut
300F	C9	RET

Programa 3

De modo que el Carry flag se activa cuando el segundo número es mayor que el primero. O si lo desea, hacemos una especie de comparación, cotejando el número que hay en el registro A con el que le restamos. Si el segundo es mayor, se activa el Carry flag.

La importancia de las comparaciones

Puede asombrarnos la cantidad de veces que necesitamos comparar bytes en código máquina, pero el hacerlo como hasta ahora —*usando SUB*— no es muy usual. Veamos, aunque podemos utilizar el Carry flag con SUB de este modo para hacer una comparación rudimentaria entre un byte y lo que hay en el registro A, el contenido del registro A desde luego cambiará en el proceso. Después de todo, le hemos restado un número.

SUB n resta el número n de lo que hay en el registro A y almacena el resultado devuelto también en el registro A.

DIRECCION	QPCODE	MENMONICO
3000	3E ?1	LD A, ?1
3002	D6 ?2	SUB ?2
3004	32 F8 2F	LD (&2FF8), A
3007	D2 0F 30	JP NC, &300F
300A	3E 07	LD A, &07
300C	CD 5A BB	CALL CharOut
300F	C9	RET

Programa 4

Esto lo representamos como:

A < ---A-n

De todas formas, los diseñadores del Z80 nos proporcionan una instrucción de comparación que salva este problema CP n. Esta instrucción coloca al número en el registro A, le resta n, activa los flags convenientemente, pero NO devuelve el resultado a A. Sencillamente, lo descarta, deja sin cambiar el número en A.

DIRECCION	OPCODE	MNEMONICO
3000	3E ?1	LD A,?1
3002	FE ?2	CP ?2
3004	32 F8 2F	LD (&2FF8), A
3007	D2 0F 30	JP NC, &300F
300A	3E 07	LD A, &07
300C	CD 5A BB	CALL CharOut
300F	C9	RET

Programa 5

Por tanto, nos encontraremos con que los bytes de memoria y los registros no son afectados por CP n —sólo los flags son activados y desactivados convenientemente—. Es decir:

— Si el número comparado con A es mayor que el de A, se activa el Carry flag.

— Si el número comparado con A es menor o igual que el de A, se desactiva el Carry flag.

Matemáticamente:

A < n------ > Carry activado.

A > =n------ > Carry desactivado.

El programa V es esencialmente el programa IV con el SUB reemplazado por un CP. De nuevo, colocaremos nuestros propios números en las posiciones de memoria &3001, &3003 y veremos si podemos predecir los pitidos exactamente.

DIRECCION	OPCODE	MNEMONICO
3000	3E 20	LD A, &20
3002	CD 5A BB	CALL CharOut
3005	C6 01	ADD A, 1
3007	D2 02 30	JP NC, &3002
300A	C9	RET

Programa 6

Hace falta mirar bien &2FF8 después de ejecutar el programa. Debe probarnos que el contenido de A realmente no ha sido cambiado por CP.

Para ilustrar cómo podemos usar la instrucción CP, veamos el programa VI. La encontraremos enseguida —de hecho la encontraremos en el primer bucle.

Como puede ver, escribe todos los caracteres con código comprendido entre &20 y &FF, cargando A con &20, escribiendo el caracter con este código Ascii, incrementando en uno lo que hay en A, saltando atrás con un JP NC para escribir A otra vez, incrementarlo de nuevo, y así sucesivamente.

Finalmente se escribe el caracter correspondiente a &FF y el valor de A incrementando en uno, llega **«como el reloj»** a cero, activa el Carry flag y sale del bucle.

DIRECCION	OPCODE	MNEMONICO
3000	3E 1F	LD A, &1F
3002	C6 01	ADD A, 1
3004	CD 5A BB	CALL CharOut
3007	FE FF	CP &FF
3009	DA 02 30	JP C, &3002
300C	C9	RET

Programa 7

Lo que hace es aprovecharse del hecho de que el Carry se activa cuando sobrepasamos el límite de &FF (255). Alternativamente, podemos usar CP para controlar el último caracter del bucle y tiene algunas ventajas más. El programa VII muestra la idea.

Fíjese que esta vez hemos añadido uno al registro A «antes» de escribirlo. Entonces lo comparamos directamente con &FF, ya que es el último caracter que queremos imprimir. Esto quiere decir, que cargamos el registro A inicialmente con &1F, pero inmediatamente es incrementado a &20 por la instrucción ADD A, 1.

DIRECCION	OPCODE	MNEMONICO
3000	3E 5A	LD A, &5A
3002	CD 5A BB	CALL CharOut
3005	D6 01	SUB 1
3007	FE 41	CP &41
3009	D2 02 30	JP NC, &3002
300C	C9	RET

Programa 8

Ahora hasta que el registro A tenga &FF, el número que comparamos con A es mayor que el que hay en A, así que el Carry flag está activado, de forma que saltamos atrás a la cabeza del bucle otra vez con JP C.

Una vez que el registro A consigue &FF, los números que estamos comparando son iguales y el Carry flag se desactiva.

Podríamos preguntarnos por qué no añadimos CP o después del ADD A,1 del programa VI. Bueno, si lo piensa no podemos seguir el camino del reloj restando o de un número, puesto que el Carry no se activará nunca.

Lo más agradable del uso de CP en bucles es que podemos variar el número final fácilmente. Por ejemplo, si queremos escribir hasta Z —Ascii &5A— necesitamos cambiar el CP &FF por CP &5A. Y, si queremos comenzar por A —Ascii &41— cargamos nuestro acumulador inicial reemplazando LD A, &1F por LD A, &40. Recordemos que enseguida le estamos añadiendo uno al registro.

Otra ventaja de usar CP para activar el Carry mejor que recurriendo al camino del reloj es que nos permite hacer cosas tales como escribir el alfabeto hacia atrás, como en el programa VIII.

Como último ejemplo del uso de CP vamos a utilizarlo como filtro de entradas fuera de rango.

A menudo en un programa dejamos que la gente de su opinión elija entre cinco opciones de un menú. Ellos responden pulsando el número comprendido entre 1 y 5, correspondiente a su preferencia. Si pulsan una tecla equivocada se ignora y el programa espera hasta que se hace la elección válida.

Menú en código máquina

Así que vamos a escribir un programa que espera los números 1, 2, 3, 4 ó 5 como entradas y los saca en pantalla.

Usaremos CharIn para dar entrada a un dato pulsando una tecla y entonces controlará si está en el rango con dos comparaciones.

DIRECCION	OPCODE	MNEMONICO
3000	CD 18 BB	CALL CharIn
3003	FE 31	CP &31
3005	DA 00 30	JP C, &3000
3008	FE 36	CP &36
300A	D2 00 30	JP NC, &3000
300D	CD 5A BB	CALL CharOut
3010	C9	RET

Programa 9

Código máquina

Hay que recordar que CharIn coloca el valor Ascii del caracter cuya tecla se ha pulsado en el registro A, así que necesitaremos revisar qué valores están en el rango &31 a &35, los códigos Ascii de 1 a 5.

En otras palabras, el número en el registro A debe ser mayor o igual a &31, nuestra primera comparación. También debe ser menor que &36, nuestra segunda comparación.

El programa IX lo pone en práctica:

El CP &31 comprueba que el número en el registro A es &31 o mayor. Si no, se activa el Carry y saltamos atrás, al comienzo para esperar otra entrada, puesto que el caracter elegido era **«demasiado pequeño».**

Si conseguimos pasar este control, se compara el número con &36. Necesitamos que el número que hay en el registro A sea menor que éste, así que en el momento en que necesitemos el Carry se activa. Si no es así el valor de la tecla es «demasiado grande» así que, saltaremos atrás para conseguir otra entrada.

Si, no obstante, el Carry se activa, sencillamente continuamos. Si el programa ha llegado hasta este punto es que el número en A está comprendido en el rango que queremos así que CALL CharOut y RET.

Una vez que hemos visto cómo trabaja ¿por qué no intentamos escribirlo aceptando solamente las letras mayúsculas del alfabeto, &41 a &5A? No nos resultará demasiado difícil.

Bueno, pues esto es todo de momento. Hay muchas más cosas que pueden hacerse con CP y bucles en general, lo veremos próximamente.

COMPUTICUE

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

COMPUTICUE

Embajadores, 90 28012 Madrid Tfno. 2270980

En algunos programas aparece el guarismo «^», sin embargo no conseguimos localizarlo en el teclado. ¿Tiene algún sustituto? ¿Hay que hacer un programa que nos lo dibuje? ¿Cómo, una vez dibujado, lo prodremos introducir en el listado del programa que lo precise?

Alvarez Márquez Montoro (Granada).

El guarismo al que te refieres es el carácter que se obtiene pulsando la tercera tecla de la fila superior del teclado del ***Amstrad*** *empezando por la derecha, y corresponde al signo de exponenciación.*

Sucede que la mayoría de las impresoras interpreta dicho carácter como tú indicas, y así aparece en los listados.

¿Cuántas Josticks se pueden «enchufar» al port trasero del **Amstrad**?

¿Pueden salir modelos nuevos del **Amstrad**?

¿La gama de periféricos para el **Amstrad** irá en aumento?

¿Es el **Amstrad** un ordenador con futuro?

Raúl López ***I Madrid.***

1. Se le pueden enchufar dos Joysticks.

2. Sí que pueden salir. Hasta ahora, el parque de ***Amstrad*** *incluye el CPC464, CPC664, CPC6128 y PCW8256. No tenemos noticias de la aparición de nuevos modelos a corto plazo.*

3. ***Amstrad*** *ya posee los periféricos más importantes de uso común en ordenadores, como unidades de disco e impresoras. Si te refieres a cosas como modems, plotters, digitalizadores de imagen, lápices ópticos, etc., el futuro dirá. Lo probable, como en el caso de las demás máquinas, es que vayan saliendo poco a poco.*

4. A esta pregunta debes responder tú, los lectores de nuestra revista y todas las personas que tengan, o piensen tener, un ***Amstrad****. En nuestra opinión, sí.*

Estimados amigos del **Microhobby Amstrad**:

1. ¿De cuántos pixel está compuesto un carácter?

2. ¿Cuántos colores se pueden poner en un carácter? ¿Cuántos gráficos se pueden definir?

3. ¿Tiene sprites?

4. ¿El sonido sale por el monitor? y si no es así, ¿sale con el volumen bastante alto?

5. Me parece que el monitor está muy cerca del ordenador, ¿se podría separar a un metro y medio?

Gracias.

Miguel Acedo (Madrid).

1. Un carácter está compuesto de 64 pixels, distribuidos en una retícula de 8×8. Sin embargo, el aspecto del carácter cambia con el modo de pantalla, alcanzando su tamaño máximo en el modo 0 (20 por línea) y el mínimo en el modo 2 (80 por línea). 40 por línea en el modo 1.

2. También depende del modo: 16 en el 0, 4 en el 1 y 2 en el 2.

3. Sale del ordenador, pero existe salida stereo.

4. Puedes separar el monitor del ordenador hasta que dé el cable. Si eso no te parece suficiente, añade más cable.

¿A partir de qué posición de memoria está mapeada la pantalla en memoria?

Yo sé programar en ensamblador para el Intel 8080, pero no sé cómo hacerlo con el ensamblador y el debugger del disco de CP/M.

¿Me pueden indicar cómo se utilizan?

¿Qué comandos hay que poner para ensamblar un programa? ¿Hay que montarlo? ¿Y para ejecutar?

Javier Planellas Giner (Barcelona)

La memoria de pantalla normalmente abarca de las posiciones &C000 a &FFFF, es decir, solapa con el intérprete Basic.

Decimos normalmente porque una de las cosas más curiosas de la pantalla del ***Amstrad*** *es que se «traslada» a través de la memoria por Hardware.*

En el firmware del ordenador existe una rutina que informa, entre otras cosas, del comienzo de la memoria de pantalla, pero requiere el conocimiento del lenguaje ensamblador del Z80.

Su dirección es &BC0B y devuelve en HL el offset de pantalla y en el acumulador el byte más significativo de la dirección de comienzo. La verdadera dirección se obtiene realizando un OR lógico entre el acumulador y el registro H, de esta forma:

```
CALL &BC0B
OR H
LD H,A
RET
```

Cuando esta rutina retorne, la dirección de la primera posición de pantalla estará en el registro HL.

Respecto a tu pregunta del ensamblador y debugger CP/M, estos programas son complicados de manejar y tienen una gran cantidad de comandos. Por razón del escaso espacio que disponemos, no podemos listarlos todos e indicar además cómo se usan.

Cualquier libro, de las docenas que existen sobre CP/M podrá ayudarte.

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.

Seréis, semana a semana, los encargados de construir esta página con vuestras consultas. En más de una ocasión, aquello que os preocupa ya ha sido contestado antes a otro lector o, por el contrario, puede suceder que determinada consulta aclare muchos quebraderos, de cabeza de otros aficionados.

Las cartas **«sin duda alguna»,** nos servirán de gran ayuda. Gracias a ellas podremos ir evaluando vuestras necesidades y, de este modo, modificando el contenido de MICROHOBBY AMSTRAD acorde con ello. **¡Os esperamos!**

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!

Intercambio programas para **Amstrad,** tanto juegos como utilidades. Interesados escribir a: Fco. Javier Punta Sánchez. Prol. Maestro Guridi, 3 - 2.º D. 41010 Sevilla. ¡Mandad lista!

Vendo Amstrad CPC-664 monitor verde con disco Logo-CPC-Demo, garantía Indescomp 07/85. Por 90.000 ptas. Tel (93) 699 17 52. Tardes. Manuel Miras Más. Justicia, 12 - 1. Rubí *(Barcelona)*.

Desearía contactar con usuarios de **Amstrad** CPC-464, para intercambio de ideas, libros, programas y cosas referentes al firmware, periféricos, etc... Francisco Javier García Casado. San Pedro Bautista, 16. 05005 Avila.

Desearía contactar con usuarios del CPC-464 de toda España y en especial de Vigo para intercambio de información y programas. Interesados llamar al teléfono (986) 27 07 35 y preguntar por Javi.

Vendo:
Ordenador BBC Modelo B con muy poco uso; 60.000 ptas.
Ordenador **Amstrad** CPC464 comprado en mayo 1985 con 30 juegos originales; 65.000 ptas.
Impresora Seikosha GP-100A; 30.000 ptas.
Unidad de diskette Standard; 15.000 ptas.
Alfonso Oliva Delgado. Satsuma, 13. 41006 Sevilla. Tel. (954) 67 33 20.

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.

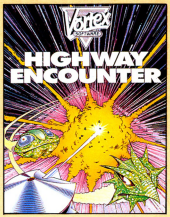
— Los autores de los programas seleccionados para su publicación, recibirán una comunicación escrita de ello en un plazo no superior a dos meses a partir de la fecha en que su programa llegue a nuestra redacción.

— **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)

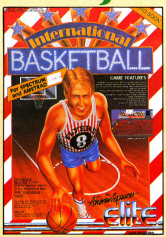


Dirige a tus cinco vortons y al lasertron hasta la «Zona Cero» por la única vía libre que queda en la tierra. Todo ello en un increíble escenario tridimensional que dan al programa un realismo sin límites.

Amstrad y ZX Spectrum: 1.900 ptas.

Fabuloso simulador de vuelo con despegue vertical desde un portaaviones, con el que podrá mantener un combate con los aviones enemigos, hacer ensayos de maniobras, practicar vuelo, ...

lo mejor de lo mejor



Fabuloso partido de baloncesto en el que podrás ensayar tus tiros favoritos contra el ordenador o frente a tu amigo

Amstrad: 2.100 ptas.
Spectrum: 2.100 ptas.

T.L.L. Lo que fue todo un éxito para el Zx Spectrum, ahora los tienes disponible para tu Amstrad

Zx Spectrum: 1.595 pts.
Amstrad: 1.900 pts.

Knigt Lore y Alien 8 con sus magníficos gráficos tridimensionales y su alto nivel de dificultad son, sencillamente, lo mejor para tu Amstrad.
P.V.P. unidad: 1.950 pts.

abc soft

Santa Cruz de Marcenado, 31
28015 MADRID
Teléfs. (91) 248 82 13
(91) 242 50 59

Si deseas recibir más información y propaganda de nuestros programas y de nuestras interesantes ofertas, por favor, rellena y envíanos este cupón.

Nombre ..... Edad .....

Dirección .....

Localidad-Provincia .....

Ordenador ..... Programas favoritos .....

DISPONIBLE PARA ZX SPECTRUM
AMSTRAD

Sound on Sound es una marca registrada producida y distribuida por Iberofón, s.a.
Telf. 671.22.00 / 04 (de 12 / 14)

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