AÑO I N.º 12

MICROHOBBY

AMSTRAD Semanal

REVISTA INDEPENDIENTE PARA USUARIOS DE ORDENADORES AMSTRAD

AÑO I N.º 12

150 Ptas.

Canarias 160 ptas.

TRATAMIENTO DE CADENAS: FUNCIONES AVANZADAS

Interrupciones desde código máquina: extraer el máximo partido del Amstrad simulando multitarea

EL RATON DE AMX CONVIERTE AL TECLADO EN UNA PIEZA DE MUSEO

RESOLUCION DE ECUACIONES DE SEGUNDO GRADO

SOFTWARE

Béisbol y fútbol, frente a frente

HOBBY PRESS, S.A.

COMPUTIQUE

Te da más

GARANTIA

AMSTRAD ESPAÑA

Al comprar tu Amstrad te regalamos

- Estuche con ocho programas originales:
  - Fruit Machine
  - Procesador Texto
  - Alfabeto Graf
  - Oh Mummy
  - Plaga Galáctica
  - Amsdraw
  - Laberinto Sultán
  - Animal, Vegetal, Mineral
- Joystick Gunshot I
- Un estupendo libro de Basic
- El mejor programa deportivo: Decathlon
- Guía de referencia del programador
- y además te obsequiamos con un curso de introducción al Basic.

VENTA A PLAZOS HASTA 36 MESES

Nuevo Amstrad CPC6128: 109.500 ptas. (F. Verde)

COMPUTIQUE

Servimos a tiendas
Abrimos sábados por la tarde

Embajadores, 90 Tfno. 2270980
28012 Madrid

MICROHOBBY
AMSTRAD
sumario

Año 1 • Número 12 • 9 al 25 de Noviembre de 1985
150 ptas. (sobretasa Canarias, 10 ptas.)

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

Edita
HOBBY PRESS S.A.

Presidente
María Andrino
Consejero Delegado
José I. Gómez-Centurión

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

Secretaría de Dirección
Marisa Cogorro

Suscripciones
M.ª Rosa González
M.ª del Mar 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

AMSTRAD

AMSTRAD PCW 8256

- 256K RAM
- Pántalla A. R. fósforo
- 90 columnas, 32 líneas
- Disco 3" 180K (incluído)
- CP/M Plus
- Impresora. 90 cps (incluída)
- Soft (Procesador Texto) (incluído)

* GARANTIA AMSTRAD ESPAÑA

AMSTRAD CPC 6128

- 128K RAM
- 48K ROM
- CP/M 2.2 - CP/M Plus
- GSX
- Disco 3" (180 Kb.)
- Obsequio (6 prog. Disco)
- VERSION MONITOR COLOR O F. VERDE

* GARANTIA AMSTRAD ESPAÑA

AMSTRAD CPC 464

- 64K RAM
- 32K ROM
- Versión M. color o F. verde
- Obsequio (8 cintas + Manual de Referencia)

* GARANTIA AMSTRAD ESPAÑA

PROXIMA APERTURA EN:

Paseo de la Castellana, 126 Madrid

SPEECH SYNTHETIZER

Permita que le hable su AMSTRAD con el sintetizador de voz

AMPLIACION MEMORIA

64K - 256K

¡Aumenta la potencia de tu AMSTRAD!

AMSTRAD RS 232

Interface vía Serie

MODULADOR-ALIMENTADOR

Permite conectar el AMSTRAD 464 - 664 - 6128 a T.V. Color

UNIDAD DE DISCO ADICIONAL

¡Multiplica las posibilidades de tu AMSTRAD!

AMSTRAD SOFTWARE CASSETTE

- SELECCION JUEGOS
  - Exploding Fist (Karate) ..... 2.300 Pts.
  - Beach Head ..... 2.700 Pts.
  - Fighter Pilot ..... 2.200 Pts.
  - Grand Prix Rallye II ..... 1.900 Pts.
  - World Cup Football ..... 1.800 Pts.
  - Decathlon ..... 2.200 Pts.
  - Pijamarama ..... 2.000 Pts.
  - Boxing ..... 2.200 Pts.
  - Bay Scauf ..... 1.900 Pts.
  - Traffic ..... 1.900 Pts.
  - Sorcery ..... 1.900 Pts.
  - Alien ..... 1.900 Pts.
  - Hockey ..... 2.000 Pts.
  - Airwolf ..... 2.000 Pts.
  - Gotscrosher ..... 1.900 Pts.
  - Flipper-Bumper ..... 1.900 Pts
- PROFESIONALES-UTILIDADES
  - Base de Datos-Etiquetas
  - Stock-Aid
  - Invostat
  - Masterfile
  - Mastercalc
  - Mascopy-Tasprint
  - Font Editor
  - Mini Office (B. datos, Graphics, Calc, Text)
  - Diseñador de P.
  - Amsword
  - Pascal
  - Contabilidad Personal
  - Sistema X (Ampliación Basic)
  - Backup
  - Transmat (Transf. cinta/disco)
- JUEGOS
  - Roland en el Espacio
  - Sorcery
  - Pijamarama
  - Airwolf
  - Rally II
- PROFESIONALES-UTILIDADES
  - Ensamblador Z
  - Sistema X (Ampliación Basic)
  - Printer Pack (Volcados, etc.)
  - Transmat (Transf. cinta/disco)
  - Contabilidad General
  - Control Stock
  - Vencimientos
  - Projet Planner
  - Decisión Maker
  - Planetarium
  - Copitoles/Ríos

LIBROS

- C. Autodidacta Basic I ..... 2.900 Pts.
- C. Autodidacta Basic II ..... 2.900 Pts.
- Programando con AMSTRAD ..... 2.400 Pts.
- T. Programación Gráficos ..... 1.950 Pts.
- Código Máquina (princip.) ..... 2.100 Pts.
- Juegos Sensacionales AMSTRAD ..... 1.950 Pts.
- Manual de Firmware (Inglés) ..... 3.900 Pts.

Envíos gratis a provincias
Tenemos la mayor cantidad de periféricos
Extenso surtido en impresoras
Pedidos por correo a:
Puerto Rico, 21 28016 MADRID

Madrid:
C/. Puerto Rico, 21
Tfno: 250 74 04

Valladolid:
C/. Juan de Juni, 3
Tfno: 33 40 00

Bilbao:
C/. Alameda de Urquijo, 63
Tfno. 431 96 67

Zaragoza:
C/. León XIII, 2 y 4

Chips & Tips

C/. Puerto Rico, 21. Madrid

Primera plana

NUEVA RED LOCAL DE IBM

Hace un año, IBM anunció la creación de un nuevo sistema de cableado que permitiría enlazar ordenadores en lo que se denomina una red local, reduciendo al mínimo los problemas de pérdida de información o corrupción de los datos.

La compañía americana ha revelado ahora la verdadera magnitud de su sistema, en el que viene trabajando a lo largo de los últimos 20 años en sus tres laboratorios de investigación de todo el mundo.

En el campo de las comunicaciones, las injerencias causadas por ruidos extraños constituían un enemigo difícil, llegando a entorpecer e incluso imposibilitar la transferencia de información. El nuevo sistema desarrollado en el laboratorio de Ruschlikon (*Zurich*) permite una fiabilidad casi absoluta.

Los 260 equipos, IBM o compatibles, que pueden conectarse a la red, lo hacen a través de un adaptador que a su vez está unido a una multiestación, la cual revisa la información antes de transmitirla al siguiente equipo.

Por otra parte, la posibilidad de unir máquinas no IBM facilitan la inclusión en la red de otras marcas sin que sus fabricantes tengan que invertir dinero en conseguirlo.

TANDY COMPUTERS VENDERA AMSTRADS

Probablemente bajo el lema secreto de **«al sol que más calienta»**, Tandy, una compañía fabricante de ordenadores se ha subido al tren de **Amstrad** sin pensárselo mucho. A partir de ahora, los venderá a mansalva, incluyendo paquetes de software de su propia creación.

Las navidades, que se acercan.

Tandy declaró que espera, a través de **Amstrad**, aumentar su presencia e importancia en el mercado de los **«home computers»**.

RATON PARA LA GAMA CPC

Uno de los periféricos más interesantes en la historia de los ordenadores caseros ha sido adaptado y comercializado para la gama CPC de **Amstrad**.

Se trata del ratón de AMX, creado por Advanced Memory Systems.

Ha sido elegido periférico del año en Inglaterra, vendiendo más de 10.000 unidades de la versión adaptada al BBC Micro en los primeros nueve meses.

La versión **Amstrad** ofrece a los usuarios un entorno de programación y desarrollo enteramente nuevo, convirtiendo al teclado poco menos que obsoleto.

El paquete incluye AMX Art, un programa de dibujo asistido por ordenador basado en ventanas en pantalla, iconos y menús del tipo **«pull-down»**.

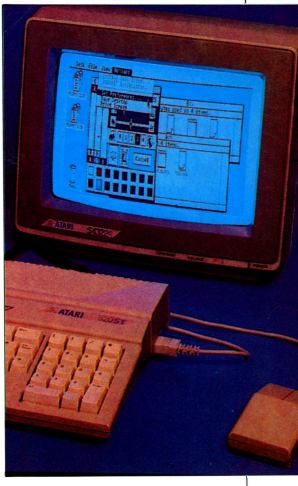
Otro programa incluido, el **«Pattern Designer»**, podríamos traducirlo como **«diseñador de formatos»**, nos capacita para crear un ilimitado número de diseños que pueden ser guardados en disco y posteriormente empleados con el AMX Art.

El usuario puede crear un entorno tipo ratón en sus propios programas mediante AMX Control, que extiende adecuadamente el set de comandos del **Amstrad** Basic para reflejar el nuevo entorno de programación.

También, **«Icon Designer»**, **«diseñador de iconos»**, es una utilidad prevista para que el usuario cree y use los iconos en sus programas.

Nick Pearson, de AMX, aseguró que podían adaptar el ratón y su software a cualquier ordenador, pero que habían elegido **Amstrad** con vistas a la campaña de Navidad, donde se espera que «barra», al menos en Inglaterra.

Tarde o temprano, las ideas del Apple MacIntosh tenían que llegar a nuestra casa. AMS ha sido el primero en hacerlo posible.



ATARI 520ST

Investrónica, la empresa española que distribuye los productos de Sinclair Research, Spectrum y QL, comercializará en nuestro país el Atari 520ST, conocido como Jackintosh en honor de su creador financiero y de su similitud con el Apple MacIntosh.

Al parecer, Investrónica quiere venderlo lo más rápidamente posible (Navidad de nuevo), y, si no fabricarlo en España, al menos ensamblarlo aquí (*Eurohard sabe mucho de esto*).

Se desconocen fechas, precios y software con el que el nuevo ordenador estará dotado.

AVANZAMOS EN LAS CADENAS

Es el momento de examinar una serie de funciones para el tratamiento de cadenas de una mayor importancia y complejidad, basándonos en lo anteriormente expuesto en estas series.

M. Barco

Si vuelve sobre el artículo anterior recordará que nos ocupamos de las cadenas. En particular aprendimos algo sobre las funciones Basic, CHR$, ASC y LEN.

Por tanto, si queremos encontrar el LEN de una concatenación de CHR$ esperamos que comprenderá que estamos **«Asciiando»** para usted.

Y si ha entendido todo esto, el programa I —la semana pasada programa X— no le debe causar problemas.

La pega que tiene este programa, diseñado para admitir sólo palabras de cuatro letras, es que acepta como dato de entrada 1234. Esperamos que haya sido capaz de ver que una línea semejante a:

```
55 IF ASC (entrada$) < 65 THEN
      GOTO 40
```

resuelve en parte el problema. Ahora el programa no aceptará «entrada$» que comience con números pero aceptará cosas tales como q234. Seremos capaces de tratar con esto cuando nos encontremos con más comandos de manejo de cadenas en un futuro artículo.

En esta ocasión echaremos un nuevo vistazo a las variables literales o cadenas y veremos comó pueden cambiarse a números y viceversa. Y también sabremos por qué tenemos que verlo antes de nada.

Con o sin comillas

Antes de empezar, sin embargo, es importante que comprendamos la diferencia entre un número y una cadena. Si no puede explicar los diferentes resultados obtenidos con:

```
PRINT 2+2
```

y

```
PRINT «2+2»
```

entonces será mejor que vuelva a leer alguno de los artículos del comienzo de la serie. De todas formas por ahora todo lo que vemos es que el primer PRINT toma el 2+2 como números, los suma y da el resultado.

El segundo comando PRINT encuentra las comillas y escribe, sin cambiarlo, todo lo que sigue hasta que encuentra otras comillas.

Resumiendo, 2+2 se trata como una expresión, o suma, mientras **«2+2»** se maneja como una cadena, las comillas son delimitadores. Como recordará, una cadena es precisamente una colección de letras, signos de puntuación, espacios y números, tratada como un todo. Puede que no sea una descripción exquisitamente técnica, pero funciona.

No debemos pensar en las cadenas únicamente como una colección de letras semejante a:

nombre$ = **«Pedro»**

donde la variable literal «nombre$» contiene las letras que forman un nombre de pila. Puesto que, la combinación de letras y números como:

nombre$ = **«C3PO»**

es válida y hasta las cadenas que consisten solamente en cifras como:

nombre$ = **«666»**

también son correctas. Vamos a centrar nuestra atención esta vez en las cadenas hechas a base de combinaciones de letras y números, y las funciones Basic que tratan con ellas.

Conversión numérica-alfanumérica

La primera función que veremos sera STR$. Se usa para convertir un número en una cadena.

Primeros pasos

Para transformar el número 3 en una variable literal llamada **«número$»** utilizamos:

número$ = STR$ (3)

y lo escribimos con:

PRINT número$

encontrándonos que ahora la cadena contiene un 3. Tengamos en cuenta, sin embargo, que es una variable literal. El **Amstrad** no admitirá que intentemos hacer matemáticas con ella.

PRINT número$ * 3

nos da un mensaje de error **«type mismatch»** (*incongruencia de tipos*), advirtiéndonos que estamos intentando utilizar una cadena como si fuera un número.

STR$ no trabaja sólo con números, también lo hará con variables numéricas (*variables que contengan números*) y también con expresiones. Por lo tanto:

número = 24
número$ = STR$ (*número*)

coloca 24 en la cadena «número$», mientras que:

resultado$ = STR$ (*72/12*)
PRINT resultado$

nos muestra que la variable de cadena **«resultado$»** contiene el caracter 6.

Observemos que en el último caso la expresión que está entre paréntesis se evalúa antes que STR$ convierta al conjunto en una cadena.

Usar STR$ con numerosos negativos nos ayuda a resaltar la diferencia entre cadenas y números. No nos cogerá de sorpresa encontrar que haciendo:

abc = —1.23
número$ = STR$ (*abc*)

resulta que la variable literal **«número$»** contiene la cadena —1.23.

Es un punto a considerar, cuando utilizamos STR$ para convertir números en cadenas —los espacios cuentan como caracteres. Si averiguamos la longitud de **«número$»** con:

PRINT LEN (*número$*)

nos devuelve cinco. Esto tiene sentido porque la cadena considera al signo negativo y al punto decimal como cifras. Todo cuenta como caracteres.

No obstante, formemos una cadena con:

abc = 123
número$ = STR$ (*abc*)

y averigüemos la longitud de **«número$»** con:

PRINT LEN (número$)

La función STR$ se comporta de un modo extraño: ¡cuidado!

El resultado es 4, y no 3 como cabría esperar. Esto ocurre porque STR$ coloca un espacio delante de cualquier número positivo al que convierte en una cadena. Quien mejor conoce la causa de esto es Locomotive, la que ha escrito el Basic. Por lo tanto la longitud de **«número$»** es 4, formada por un espacio seguido de tres cifras. Este invisible caracter extra puede traernos problemas si no somos cuidadosos cuando utilizamos STR$.

Y si nos ha seguido todo el tiempo, intente explicarse el resultado de:

PRINT LEN (*abc*)

El lector más reflexivo, o quizá inconformista, puede preguntarse cuál es la importancia de todo esto. Después de todo, podemos colocar un número en una cadena sin usar STR$. Puede usarse un INPUT, como muestra el programa II.

Este programa asombroso y trivial, nos pide el nombre y el número de una calle y escribe en pantalla la dirección. Difícilmente quedará como una pieza épica de la programación pero nos contentamos con encontrar un par de puntos interesantes. Vamos a verlo de cerca:

La línea 20 coloca el nombre de la calle en **«calle$»**.

La línea siguiente pregunta por el número de la casa y lo almacena en la cadena **«número$»**. Aquí tenemos un ejemplo de un número convertido en cadena gracias a INPUT.

La línea 40 concatena (junta) las dos, introduciendo el resultado en **«dirección$»**. El CHR$ (32) intercalado entre ellos es precisamente el que separa el nombre del número.

Fijémonos en que **«dirección$»** contiene dos partes de información —el nombre de la calle y el número de la casa— en una sola variable. Volveremos sobre esto más tarde.

Como descubriremos al probar:

```
PRINT número$ + 2
```

mientras **«número$»** puede contener un número, no podemos hacer sumas con él. El programa anterior parece ser correcto pero para ejemplos más complicados puede que necesitemos utiliar el número de la casa en nuestros cálculos.

El programa III es semejante a éste. Pide una calle y un número como antes, pero ahora el bucle FOR... NEXT se encarga de escribir las 25 direcciones siguientes a la que metemos y se encuentran en el mismo lado de la calle.

Es muy similar al programa II, pero observemos que ahora el número de la casa está contenido en la variable numérica «número». Debido a que utilizaremos «número» para determinar los valores que toma la variable literal del bucle.

Intentemos cambiar, en todos los lugares donde esté **«número»** por **«número$»** y veamos qué pasa. El programa hace **«crash»** porque hemos intentado hacer cálculos con cadenas. No importa que la cadena pueda contener una número, nunca podemos utilizarla como tal, sólo es una cadena.

En la línea 50 hemos utilizado STR$ por primera vez en un programa. Aquí toma el valor del bucle (*incrementándose en 2 cada vez que gira*) y lo mete en una cadena mediante STR$. Rápidamente se une con un espacio y «calle$», como antes, se almacena en **«dirección$»**. La línea 60 escribe la información guardada en **«dirección$».**

Las cadenas son una buena forma de almacenar informaciones

Desde luego que no es un programa muy práctico, pero se le podría sacar mucho jugo. Hay que tener sólo un poco de imaginación para ver que podemos utilizarlo como base de un programa más largo.

En lugar de sacar en pantalla todas las direcciones, pueden imprimirse o, mejor, guardarse en cinta o en disco para su futura utilización.

Antes de dejar el programa III queremos hacer dos puntualizaciones sobre él. La primera es que, a diferencia del programa II, el número de la calle se guarda en una variable numérica **«número»** y no en una variable literal, **«número$»**.

Aunque en este caso no se nota demasiada diferencia, es preferible la manera de organizar las variables del programa III. Después de todo, esperamos encontrar un número en una variable numérica, mientras que una cadena podría contener todo tipo de caracteres.

Esto no debe ser un problema en los ejemplos anteriores, pero a la larga, en programas complicados, guardando los números en variables numéricas y los no números en variables literales, podemos salvar algunos errores elementales pero que nos hacen perder el tiempo.

El segundo punto es que después que hemos hablado de guardarlos por separado, hemos usado STR$ para colocar **«número»** ¡en cadena! Puede parecernos un poco contradictorio, pero francamente no lo es.

Mientras somos partidarios de guardar números en variables numéricas y no números en cadenas, al hacer uso de ellas, no hay inconveniente en convertirlos en cadenas ya que de este modo pueden almacenarse con mayor eficacia.

En el programa III **«dirección$»** maneja el contenido de la información de dos variables (*«número», «calle$»*) en una sola variable. Puede ser bastante económico, como muestra el programa IV.

Esta joya del arte de la programación calcula los sueldos de tres empleados usando una fórmula un poco extraña. El sueldo es de 1.000 ptas. por cada año de vida.

No debemos tener dificultades en seguir su forma de trabajo. El cuerpo principal del programa es un bucle FOR... NEXT que realiza ciclos de tres vueltas. Cada vez que el bucle gira se anota el nombre de un empleado y se almacena en «nombre$». Del mismo modo la edad del empleado está contenida en «edad». La línea 50 calcula la paga multiplicando «edad» por 1.000 y colocando el resultado en **«sueldo»**. La línea 60 lo saca en pantalla detalladamente.

¿Puede organizar el resultado de un modo más esmerado, quizá en columnas con encabezamientos?

Como veremos si lo ejecutamos —y si no lo hemos hecho tendremos que hacerlo— el programa funciona. Si es así y nos ha hecho caso, podrá contarnos todo sobre él. **¿Son realmente necesarias tres variables separadas?**

La línea 60 es la que, precisamente, saca en pantalla un sencillo mensaje, pero busca en las tres variables para encontrar los datos que necesitamos. El programa V hace este trabajo, pero de un modo diferente.

Aprendiendo el concepto de registro

Este programa sólo utiliza una variable, «registro$» para guardar toda la información sobre el empleado. Cada vez que el bucle FOR... NEXT pasa por las líneas 40 y 50 utiliza INPUT para almacenar los datos del empleado en **«nombre$»** y **«edad»**. La línea siguiente calcula la paga como antes guardándolo en **«sueldo»**. La línea 70 es la que marca la diferencia.

Aquí STR$ se utliza para convertir las variables numéricas **«edad»** y **«sueldo»** en variables literales. Estas concatenan inmediatamente con **«nombre$»** y todo el lote se almacena en **«registro$»**. De esta manera los tres bloques de información están contenidos en una sola variable en lugar de en tres como antes.

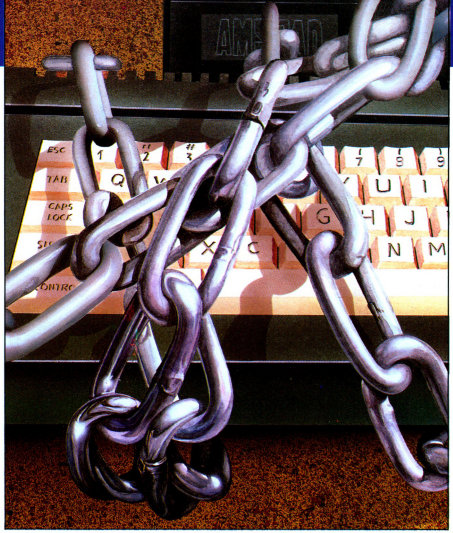
Esto es un modo mucho más eficaz de almacenar datos, aunque podríamos tener una pequeña dificultad con el formato que tienen al salir en pantalla. Y pese a su eficacia, debemos tener cuidado con lo que sigue.

Cuando la línea 80 escribe «registro$» el nombre del empleado, edad y paga son cuidadosamente separados por un espacio. Esto ocurre porque la función STR$ de la línea 70 introduce un espacio delante de cada cantidad. En este caso funciona correctamente, pero si utilizamos el método de introducir los datos en una cadena, seamos cuidadosos con estos espacios.

Si concatenamos dos cadenas normales estarán **«pegadas»** una a la otra y no habrá un espacio que indique dónde se han **«juntado»**.

Si no acaba de ver lo que queremos decir, cambie la línea 70 a:

```
70 registro$ = nombre$ + STR$ (edad)
+ «unacadena» + STR$ (sueldo) +
«unacadena»
```



Ahora cuando ejecutemos el programa V veremos que la información está desordenada. Es difícil ver los finales de las cantidades de la edad y del sueldo. Los campos de algunos registros, como hemos visto, están unidos.

Así que hay que tener en cuenta que añadir algunos espacios durante la concatenación puede hacer las cosas más comprensibles. Relacionadas con este campo, más adelante usaremos las funciones LEFT$, RIGHT$ y MID$.

Mientras el programa V discurre de un modo más eficaz que el programa IV, sin embargo no tiene flexibilidad. Con el programa IV era fácil sacar en pantalla los datos de un empleado *(edad, paga, nombre)* en un orden establecido por nosotros mismos. Solamente cambiando la forma de la línea 60, dejando intacta la estructura principal del programa.

Con el programa V, sin embargo, tenemos que cambiar la línea 70, alterando el modo de almacenamiento de los datos en **«registro$»**. Este problema puede superarse utilizando algunas técnicas de manejo de cadenas que no hemos descubierto todavía, pero aun así los cambios no son tan fáciles.

Eficacia frente a flexibilidad

De modo que aunque el programa es eficaz, no es flexible. Encontrará que, al crecer su experiencia en programación, estas opciones de elegir entre flexibilidad y eficacia le darán muchos quebraderos de cabeza. Al programador le ocurre esto cada vez que ha de hacer una selección, sin embargo, como los computadores actúan rápidamente y tienen una memoria grande y barata sospechamos que la flexibilidad se impondrá sobre la eficacia.

De momento, no obstante, observamos cómo se ha utilizado STR$ para convertir números en cadenas y almacenarlas en otras cadenas.

La información de varias variables se une en una variable literal. Más tarde aprenderemos a buscar en estas cadenas datos para nuestra información.

Por ahora, veamos si podemos cuidar la forma de expresar los datos del programa V. ¿Hay algún modo de variar el formato no muy claro en que se muestra el contenido de «registro$»? Una solución se encuentra en los caracteres de control como tratamos la última vez. La siguiente parte añadida al programa V hará las cosas esmeradas. **¿Puede explicarnos cómo trabajan?**

Ahora vamos a ver el programa VI, por el momento una versión más eficaz de los programas IV y V. Es el más corto y utiliza menos variables que los otros dos, y sin embargo, hace el mismo trabajo. Como podemos ver, no tiene la variable **«nombre»**, en cambio se utiliza **«registro$»**. Es muy similar, pero tiene una línea distinta y que varía para calcular y almacenar el sueldo, se utiliza STR$ con la expresión **«edad»** * 1.000 en la línea 50.

Aunque podría ser un programa más eficaz, no parece que lo sea demasiado. No sólo ha perdido flexibilidad, sino que es mucho más difícil de entender. Omite las variables **«nombre$»** y **«sueldo»** cuidándose de ocultar dónde están ocurriendo las cosas.

Cuando revise el programa, la semana que viene como muy tarde, pregúntenos, **¿dónde están calculados los sueldos?** Y si le empezamos a dar vueltas, **¿nos daremos cuenta de que «registro$» de la línea 30 contiene una información completamente diferente de la de «registro$» de la línea 50?**

Parece que de los tres, preferimos el programa V. Hace su trabajo, como nos interesa, tiene un aceptable nivel de eficacia, flexibilidad y comprensibilidad. Además, sigue este método de almacenar números en cadenas que nos conduce hacia el próximo tema.

Hasta aquí hemos estado ocupados convirtiendo números en cadenas usando STR$. Así hemos visto que podemos cambiar un grupo de variables numéricas y literales en una larga cadena llena de información.

Pero, una vez que tenemos nuestros números convertidos en cadenas, **¿qué ocurre si queremos volverlos a números otra vez? ¿Hay alguna forma de convertir cadenas, o parte de ellas, otra vez en números?** Es una buena pregunta.

Conversión alfanumérica-numérica

Desde luego, la función ASC que tratamos la última vez podría convertir una cadena en un número, pero no es esto lo que necesitamos. Aunque:

PRINT ASC(«A»)

puede devolvernos 65, lo mismo que

```
PRINT ASC(«AB»)
```

y

```
PRINT ASC(«ABC»)
```

De un modo semejante

```
PRINT ASC(«1»)
```

nos da el mismo resultado que:

```
PRINT ASC(«12»)
```

Casi no se utiliza esta función para sacar números de las cadenas donde los tenemos almacenados. Lo que necesitamos es la función llamada con acierto VAL. Esta función convierte la parte numérica de una cadena otra vez en número, así:

```
PRINT VAL(«12»)
```

devuelde 12 y, nos demuestra que realmente lo ha transformado en un número y que podemos realizar operaciones matemáticas con él. Probemos:

```
PRINT VAL(«12»)*12
```

y nos da 144.

Para que VAL trabaje correctamente con una cadena ésta ha de comenzar con un signo más o un signo menos o con un número. Si comienza con cualquier otra cosa, obtenemos un 0 para nuestro pesar. Así:

```
PRINT VAL(«—56.67»)
```

devuelve —56.67, mientras que:

```
edad$ = «+7»
edad = VAL(edad$)
PRINT edad
```

nos da 7. A propósito, este último ejemplo demuestra que VAL puede trabajar con una variable literal.

De todas formas tengamos cuidado con las cadenas que contienen expresiones, porque VAL sólo trabaja con el primer número. Por lo tanto:

```
suma$ = «25+45»
PRINT VAL(suma$)
```

sólo da 25 como resultado. Y recuerde que es necesario comenzar con un número o con el signo más o el signo menos. Intentemos:

```
nono$ = «*123»
vuelta = VAL(nono$)
PRINT vuelta
```

y todo lo que nos devolverá será 0. Igualmente:

```
PRINT VAL(«chirivitel»)
```

nos da 0 porque comienza con una letra.

```
10 REM PROGRAMA I
20 entrada$=""
30 WHILE LEN(entrada$)<>4 OR VAL(en
trada$)<>0
40 PRINT "Teclee una palabra de cua
tro letras ";
50 INPUT entrada$
60 WEND
70 PRINT entrada$
```

```
10 REM PROGRAMA II
20 INPUT "Nombre de la calle? ",cal
le$
30 INPUT "Casa numero? ",numero$
40 direccion$=calle$+CHR$(32)+numer
o$
50 PRINT"La direccion es: Calle "di
reccion$
```

```
10 REM PROGRAMA III
20 INPUT "Nombre de la calle? ",cal
le$
30 INPUT "Casa numero? ",numero
40 FOR bucle=numero TO numero+50 ST
EP 2
50 direccion$=calle$+CHR$(32)+STR$(
bucle)
60 PRINT"La direccion es: Calle "di
reccion$
70 NEXT bucle
```

```
10 REM PROGRAMA IV
20 FOR bucle=1 TO 3
30 INPUT "Nombre? ",nombre$
40 INPUT "Edad? ",edad
50 sueldo=edad*1000
60 PRINT nombre$,edad,sueldo
70 NEXT bucle
```

```
10 REM PROGRAMA V
20 FOR bucle=1 TO 3
40 INPUT "Nombre? ",nombre$
50 INPUT "Edad? ",edad
60 sueldo=edad*1000
70 registro$=nombre$+STR$(edad)+STR
$(sueldo)
80 PRINT registro$
90 NEXT bucle
```

```
10 REM PROGRAMA VI
20 FOR bucle=1 TO 3
30 INPUT "Nombre? ",registro$
40 INPUT "Edad? ",edad
50 registro$=registro$+STR$(edad)+S
TR$(edad*1000)
60 PRINT registro$
70 NEXT bucle
```

```
15 registro$=""
70 registro$=registro$+CHR$(13)+CHR
$(10)+nombre$+STR$(edad)+STR$(sueld
o)
80 CLS
100 PRINT registro$
```

Veamos la diferencia existente entre:

```
PRINT VAL(«Pedro34»)
```

y

```
PRINT VAL(«34Pedro»)
```

La primera devuelve 0 porque la cadena empieza con una letra, mientras la segunda devuelve 34. VAL toma todos los números que puede e ignora el resto de caracteres.

La función VAL permite otras soluciones parciales al problema planteado por el programa I. Utilizando una línea como:

```
55 IF VAL(entrada$) < 0 THEN GO-
   TO 40
```

se soluciona el problema más eficazmente que usando ASC.

Y esto es todo sobre lo que investigamos por esta vez. Hemos visto cómo se utiliza STR$ para convertir números en cadenas y cómo VAL hace lo inverso. También hemos dado con un modo sumamente compacto de almacenar información, como una variable que estará compuesta por un grupo de ellas.

Veamos si podemos emplear lo que hemos tratado hace un par de semanas para crear nuestro propio programa almacenando y sacando información. Y, si le queda tiempo libre después de esto, **¿puede depurar los programas anteriores y quizá hacer al programa VI más eficaz y menos extenso?**

Deberá continuar atareado hasta el próximo capítulo, cuando dejemos por un momento las cadenas y READ todo sobre DATA.

¡EXCLUSIVA MUNDIAL!

Editado por:
SINTAX, S.A.

YOUR COMPUTER

La Revista de ordenadores de mayor venta en toda Europa

¡SE PUBLICA DESDE AHORA EN ESPAÑA, EN FORMA DE CASSETTE!

Si, ya está confirmada la sensacional noticia. Muy pronto estará ***en los quioscos*** *de toda España una selección de los mejores juegos y utilidades publicados por la prestigiosa Revista británica «YOUR COMPUTER», editados en cassette de alta calidad y con instrucciones en castellano.*

El ***prestigio*** *alcanzado por Your Computer, tanto en Inglaterra como en España y otros países, se debe, de una forma muy especial, a la* ***gran calidad de los programas*** *que publica, la mayor parte de ellos en Código Máquina, y con la utilización de rutinas y técnicas de programación muy depuradas.*

Ahora, a un precio inmejorable, podéis tener acceso a estos programas, ***evitandoos*** *la difícil tarea de* ***teclearlos*** *en vuestro ordenador.*

¡Y ***cada mes*** *estará en la calle una nueva cinta!*

Si no encuentras la cassette de **«Your Computer»** en tu quiosco o tienda de informática, solicítala a nuestras oficinas:

SINTAX, S. A.
«YOUR COMPUTER»
Paseo de la Castellana, 268
28046 Madrid

Envía tus señas completas, teléfono **y marca de ordenador** e incluye **talón bancario,** o remite **Giro Postal** por el importe.

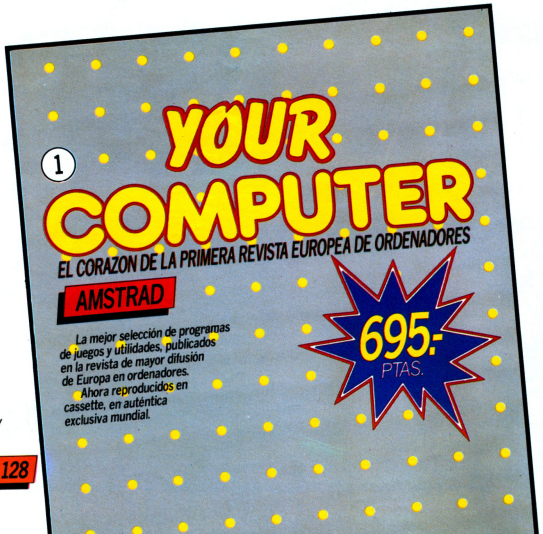
No te cobraremos gastos por el envio.

Si prefieres pagar **contra reembolso,** entonces incluye, junto a tu pedido, dos sellos de 50 ptas. cada uno para gastos de envio.

TAMBIEN DISPONIBLE PARA

COMMODORE 64 y

SPECTRUM 48, PLUS, 128



DISPONIBLE PARA ZX SPECTRUM
AMSTRAD

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

Sound-on-Sound

JUEGA CON EL FUTURO

SOFTWARE

¡¡¡NO LO SUEÑES!!! ¡JUÉGALO!
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

OTELO

Serie Oro

PROGRAMA REALIZADO POR EL LECTOR:
José Planelles Segui *(Alicante)*

COMPATIBLE
CPC 464
CPC 664
CPC 6128

El «Otelo» es uno de los juegos de estrategia e inteligencia más conocidos. Sorprende por la simplicidad de sus reglas y la profundidad y larga vista que hay que tener para jugar con perfección.

El programa, en mi opinión, está bastante logrado: la presentación es aceptable, sencilla y elegante, y el algoritmo mediante el cual el ordenador juega es bastante potente. Por lo menos a nosotros, no nos ha resultado fácil en absoluto ganarle. Desde el punto de vista del programador, pienso que el algoritmo de juego es mejorable *(¿cuál no?)*, pero recurriendo ya a técnicas y representaciones más propias de la inteligencia artificial.

No dudo ni por un momento que algún lector se planteará ese reto.

OTELO: GUIA DEL USUARIO

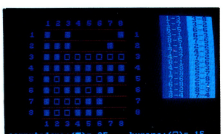
Tras cargar el juego, el programa pregunta el orden de las jugadas. Si contestamos con «s», jugaremos primero. La ventana azul oscuro de la derecha indica las jugadas del ordenador. La azul claro, las jugadas propias.

El ordenador te pregunta, cuándo te toca mover, tu jugada. Debes responder con un número compuesto por las coordenadas x *(columna)* y la coordenada y *(fila)*. Así, la ficha correspondiente a la x=3, y=4, será el número 34.

Debes tener en cuenta que un error en la jugada supondría la pérdida de movimiento, excepto si ésta es mayor de 88, la posición está ocupada y no es 99. Si no puedes tirar con introducir el número 99 basta.

JUEGO

El objetivo del juego consiste en tener más fichas que el contrario; para ello te vales de lo siguiente. Debes «volcar» las fichas del contrario, situando una tuya en una posición, de forma que todas las fichas del contrario que haya entre esa ficha nueva y otra también tuya pasan a ser tuyas. No puedes tirar si no puedes volcar ninguna ficha enemiga. El juego termina cuando uno de los dos contrincantes no tiene fichas, el tablero está lleno o cuando ninguno de los jugadores puede poner una ficha más.



SUBRUTINAS

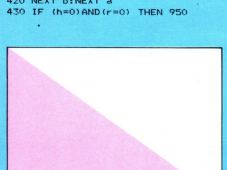
LINEAS	COMENTARIOS
110	Definición de las pantallas, tintas y símbolos
270	Cálculo de la jugada del ordenador
420-460	Ponderación de las jugadas: selección de la mejor jugada
530	Introducción del movimiento del jugador
720	Impresión del tablero, fichas y tanteo
880	Cálculo del movimiento válido para el ordenador y jugada válida para el jugador
1020	Títulos de final de partida y rutina de comienzo de nueva partida
1080	Iniciación de variables, definición del tablero y colocación de fichas iniciales
1240	Dibujo de las líneas del tablero

VARIABLES

NOMBRE	FUNCION
S	Contiene ficha contraria
O	Contiene carácter 128: jugador
T	Contiene ficha propia
X	Contiene caracter 143: ordenador
H	Contiene máx. jugada y contador
A	General
B	General
Q	Contiene jugada actual: valor
C	General de contador
D	General
K	Contador
F	General
G	General
M	Pos x de la max y jugada propia
N	Pos y de la max jugada y jugada propia
TH$	Recoje pulsación del teclado
R	Jugada propia
POSV	Indica si la jugada es válida
A$	General
A	Matriz: tablero
W$	General
I	General

Serie Oro


```
10 REM **************************
20 REM          O T H E L O
30 REM   JOSE I PLANELLES SEGUI
40 REM **************************
50 REM  DEFINICION DE PANTALLAS Y C
ARACTERES
60 MODE 1
70 WINDOW #0,1,29,1,21
80 WINDOW #2,1,40,22,25
90 WINDOW #1,30,34,1,15
100 WINDOW #3,35,39,1,15
110 PAPER #1,2:PEN #1,3:CLS #1
120 PAPER #3,3:PEN #3,2:CLS #3
130 PAPER #2,0:PEN #2,3:PAPER 0:PEN
 2
140 SYMBOL AFTER 127
150 SYMBOL 128,255,129,129,129,129,
129,129,255
160 INK 0,0
170 BORDER 0
180 INK 1,6
190 INK 2,2
200 INK 3,20
210 REM OTELO.CALCULO DE LA JUGADA
DEL ORDENADOR:PESOS
220 GOTO 1050
230 SOUND 1,45,10,7
240 LOCATE #2,1,4:PRINT #2,"MI MOVI
MIENTO.............";
250 s=n:t=n:h=0
260 FOR a=2 TO 9:FOR b=2 TO 9
270 IF a(a,b)<>32 THEN 420
280 q=0
290 FOR c=-1 TO 1:FOR d=-1 TO 1
300 k=0:f=a:g=b
310 IF a(f+c,g+d)<>s THEN 340
320 k=k+1:f=f+c:g=g+d
330 GOTO 310
340 IF a(f+c,g+d)<>t THEN 360
350 q=q+k
360 NEXT d:NEXT c
370 IF (a=2) OR (a=9) OR (b=2) OR (
b=9) THEN q=q*4
380 IF (a=3) OR (a=8) OR (b=3) OR (
b=8) THEN q=q/2
390 IF ((a=2) OR (a=9))AND((b=3) OR
 (b=8))OR((a=3) OR (a=8))AND((b=2)
OR (b=9)) THEN q=q/3
400 IF (q<h) OR ((RND<0.3) AND (q=h
)) THEN 420
410 h=q:m=a:n=b
420 NEXT b:NEXT a
430 IF (h=0)AND(r=0) THEN 950
```



```
440 IF h=0 THEN 460
450 GOSUB 830:PRINT #1," ";CHR$(48+
g-1);"-";CHR$(48+f-1)
460 GOSUB 660
470 REM MOVIMIENTO DEL HUMANO
480 SOUND 1,15,10,7
490 LOCATE #2,1,4:PRINT #2,"CUAL ES
 TU MOVIMIENTO ?";
500 th$=INKEY$:IF VAL(th$)=0 THEN 5
00 ELSE PRINT #2,th$;
510 r=VAL(th$)
520 th$=INKEY$:IF VAL(th$)=0 THEN 5
20 ELSE PRINT #2,th$;
530 r=r+VAL(th$)*10
540 REM SELECCIONA O PARA PASAR
550 s=x:t=o:REM LETRA O
560 IF r=99 THEN 640
570 IF (r<11) OR (r>88) THEN 480
580 PRINT #3," ";CHR$(48+r MOD 10)
;"-";CHR$(48+r \ 10)
590 r=r+11
600 m=INT(r/10)
610 n=r-10*m
620 IF a(m,n)<>32 THEN 480
630 GOSUB 830
640 GOSUB 660
650 GOTO 230
660 REM IMPRIME EL TABLERO
670 c=0:h=0
680 LOCATE 1,1
690 LOCATE 8,2:PRINT "1 2 3 4 5 6 7
 8"+GOSUB 1490
700 FOR B=2 TO 9:LOCATE 4,b*2:PRINT
 b-1;CHR$(9);
710 FOR D=2 TO 9
```


```
720 PRINT CHR$(A(B,D));CHR$(9);
730 IF A(B,D)=X THEN C=C+1
740 IF A(B,D)=O THEN H=H+1
750 NEXT D
760 PRINT B-1:PRINT
770 NEXT B
780 PRINT TAB(8);"1 2 3 4 5 6 7 8"
790 LOCATE #2,1,1:PRINT #2,"computa
dora:(";CHR$(x);")=";c;"   humano:(
";CHR$(o);")=";h
800 IF (h=0)OR(c=0)OR((h+c)=64) TH
EN 970
810 RETURN
820 REM  CALCULO DE MOVIMIENTO VALI
DO
830 POSv=0:FOR c=-1 TO 1:FOR d=-1 T
O 1
840 f=m:g=n
850 IF a(f+c,g+d)<>s THEN 880
860 f=f+c:g=g+d
870 GOTO 850
880 IF a(f+c,g+d)<>t THEN 920
890 a(f,g)=t:IF (f<>m)OR(g<>n) THEN
 POSv=-1
900 IF m=f AND n=g THEN 920
910 f=f-c:g=g-d:GOTO 890
920 NEXT d:NEXT c
930 IF NOT POSv THEN a(m,n)=32
940 RETURN
950 GOSUB 660
960 REM  FINAL DE LA PARTIDA
970 CLS:LOCATE 10,5:PRINT "YO=";C,"
TU=";H:LOCATE 10,10
```


```
980 IF c>h THEN PRINT "SOY EL CAMPE
ON!":SOUND 1,230,50,5
990 IF h>c THEN PRINT "ERES EL MEJO
R!":SOUND 1,330,50,5
1000 IF h=c THEN PRINT "ES UN CLARO
 CASO DE EMPATE!":SOUND 1,430,50,5
1010 a$=INKEY$:IF a$="" THEN 1010 E
LSE RUN
1020 REM INICIALIZACION DE VARIABLE
S Y TABLEROS
1030 CLS
1040 x=143:o=128
1050 DIM a(10,10)
1060 FOR b=1 TO 10:FOR c=1 TO 10
1070 IF (b<>1) AND (c<>1) AND (b<>1
0) AND (c<>10) THEN a(b,c)=ASC(" ")
1080 NEXT c:NEXT b
1090 a(5,5)=x:a(6,6)=x
1100 a(6,5)=o:a(5,6)=o
1110 p=0
1120 PRINT "QUIERES MOVER TU PRIMER
O?"
1130 PRINT "S/N"
1140 w$=INKEY$:IF w$="" THEN 1140 E
LSE CLS
1150 GOSUB 660
1160 IF (w$="s")OR(w$="S") THEN GOT
O 480
1170 GOTO 230
1180 REM RUTINA DE DIBUJO DEL TABLE
RO
1190 FOR i=1 TO 9
1200 PLOT 70+i*32,358,1
1210 DRAW 70+i*32,358-256
1220 NEXT
1230 FOR i=1 TO 9
1240 PLOT 102,358-(i-1)*32
1250 DRAW 70+9*32,358-(i-1)*32
1260 NEXT
1270 RETURN
```

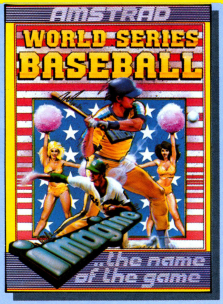
Para que tus dedos, no realicen el trabajo duro, M.H. AMS TRAD lo hace por ti. Todos los listados que incluyan este logotipo se encuentran a tu disposición en un cassette mensual, solicítanoslo.

¡NUEVO!

SIEMPRE LOS PRIMEROS EN TENER LO ULTIMO

circulo de soft
MICROAMIGO S.A.

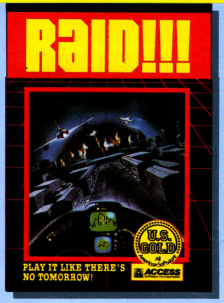
P.º de la Castellana, 268, 3.º C. 28046-MADRID.
Tel.: (91) 733 25 00



BASEBALL

Impresionante simulación en 3' dimensiones. Se puede competir contra el ordeñador o contra otro jugador. No es necesario conocer el beisbol. Hay un modo de demostración. Pantallas gigantes para ver de cerca la acción.

P.V.P.: 2.100 ptas.
Precio C. de Soft: 1.890 ptas.



RAID

Defiéndete con tu escuadrilla de aviones del ataque nuclear que han lanzado sobre ti. Tu viaje estará lleno de peligros hasta que llegues a las bases de lanzamiento de misiles enemigas. Tendrás que destruirlos para salvar a tu país de una catástrofe nuclear. Gráficos y acción sensacionales.

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

MATCH DAY

¡Ahora para Amstrad! No se trata de un juego de fútbol cualquiera. Fantástica acción en 3 dimensiones y animación total que dan vida al fútbol. Quedarás maravillado con el control del balón y desarrollarás tu destreza y técnica jugando contra otro jugador o contra el ordenador.

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

¡¡¡...Y LOS TRES PROGRAMAS POR SOLO 5.400 PTAS!!!

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

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

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

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

TITULO	P.V.P.	ORDENADOR

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

Nombre _____ Apellidos _____ Edad _____

Domicilio _____ Teléfono _____

Localidad _____ C.P. _____ Provincia _____

TRATAMIENTO DE INTERRUPCIONES

La semana pasada comenzamos hablando de sprites y concluimos descubriendo un método de programación mucho más general, apenas esbozado en el Amstrad Basic: el tratamiento de interruptores.
Es hora de plantearnos algo mucho más ambicioso: su uso desde código máquina.

Una interrupción es una señal enviada al Z80, el microprocesador que rige al **Amstrad,** informándole que se requiere su atención inmediatamente para llevar a cabo una tarea importante.

Cuando el Z80 recibe una señal de este tipo, detiene la tarea que en ese momento esté ejecutando *(recuérdese que el microprocesador siempre está haciendo algo),* atiende a la llamada, la procesa y retorna a lo que estuviera haciendo antes a partir del punto exacto donde lo dejó.

Como ha habido un programa principal momentáneamente cancelado (interrumpido), el nombre genérico de rutinas de interrupción dado a los procesos que demandan atención inmediata del ordenador está más que justificado.

Utilidad de las interrupciones

Las interrupciones se usan para muchas cosas; quizá la más obvia sea lo relacionado con el reloj interno del **Amstrad,** el cual puede leerse desde Basic mediante la pseudovariable TIME.

Cada trescientoavo de segundo, el reloj es incrementado.

También, cada cincuenteavo de segundo se genera una interrupción que permite actualizar la memoria de pantalla *(«refrescarla» para que no se borre)* y examinar el teclado para ver si se ha pulsado alguna tecla.

Por último o casi, cada 1/100 segundos se procesan sonidos y sus envolventes, si es que las hay.

Como se puede ver, las interrupciones permiten al **Amstrad** realizar una serie de tareas de manera que parece que se llevan a cabo todas a la vez.

Nosotros no nos percatamos de estos **«programas de fondo»** (en inglés background tasks) porque el **«gestor de interrupciones»** del ordenador está cuidadosamente estudiado, permitiendo que muchas tareas de fondo se ejecuten rápida y eficientemente, para no afectar al programa principal *(foreground task)* que el **Amstrad** ejecute.

Como comentábamos en el artículo de la semana pasada, el **Amstrad** Basic posee la rara habilidad de permitir manipular interrupciones desde Basic. Nos aprovechamos de ello para mostrar una simple técnica de creación y movimiento de sprites.

Por supuesto, que esto está muy bien, pero **¿cómo podríamos manipular interrupciones desde código máquina?**

Si aprendiéramos a hacerlo de forma general, podríamos aplicar el método a cualquier cosa que se nos ocurra, incluidos los programas de juegos tipo arcade.

Aquéllos que conozcan o provengan de otras máquinas, como programadores, dirán con toda la razón que el tratamiento de interrupciones es un asunto muy complejo.

Gracias al firmware

Sin embargo, en el **Amstrad** el panorama es completamente diferente. Una vez más hay que dar las gracias a los diseñadores del firmware, porque el sistema operativo proporciona un método para las interrupciones muy fácil de usar, dándole la for-

TABLA UNO

Byte 0	Dirección ***«cercana»*** (byte = 1)/*«lejana»* (byte = 0).
Bytes 1-4	Prioridad de sucesos síncronos.
Byte 5	Debe ser cero.
Byte 6	Si el suceso es ***«express»*** *(atendido inmediatamente)* debe ser 1. En nuestro caso está puesto a cero.
Byte 7	Vale 1 si el suceso es asíncrono.

ma predefinida de lo que se conoce como SUCESO *(en inglés, EVENT)*.

Se permiten tres **«fuentes»** de sucesos y cada una de ellas posee una cola de espera asociada, para ser procesada en el momento que el sistema operativo considere oportuno, y así no interferir con otros procesos catastróficamente.

Estas fuentes se denominan:

— Fast Ticker
— Ticker
— Interrupciones **«frame flyback»**

Utilizaré los nombres ingleses pensando en los lectores del libro del firmware, ya que, por otra parte, estas denominaciones se refieren al número de veces por segundo que los sucesos pueden ejecutarse y no tienen mayor significación.

Los sucesos **«fast ticker»** son requeridos cada 1/300 segundos *(el reloj interno)*.

Los otros dos cada 1/50 segundos.

Tipos de sucesos

1. Sucesos asíncronos de ejecución inmediata (express asynchronous events): como su nombre indica, son llamados y ejecutados inmediatamente durante el procesamiento de interrupciones. Normalmente no se usan.

2. Sucesos asíncronos **«normales»** *(asynchronous events):* son el tipo más flexible. Se colocan en su cola de espera y el sistema operativo les atiende cuando ha finalizado sus propias interrupciones. Tienen pocas restricciones y la rutina a ejecutar puede ser tan larga como sea necesario.

3. Sucesos síncronos *(synchronous events):* estos sucesos tienen una prioridad de ejecución asociada y se colocan en una cola de espera especial de acuerdo con ella. El programa principal la examina para ver si existen y los procesa.

Nosotros vamos a ocuparnos de los sucesos asíncronos normales *(tipo 2)*, intentando aprender cómo se crean y usan.

Cómo hacerlo en máquina

El sistema operativo del ordenador necesita un pequeño bloque de memoria como espacio de trabajo para cada suceso. En este caso, 13 bytes *(de 0 a 12)*.

El bloque se divide en dos partes:

a) bloque del reloj *(ticker block)*: primeros 6 bytes;

b) bloque del suceso *(event block)*: 7 bytes restantes.

Cada uno de estos bytes cumple la siguiente función:

Bytes 0/1: almacenan la DIRECCION del siguiente bloque de reloj, si lo hay.

Bytes 2/3: un contador decrementado cada 1/50 segundos. Cuando alcanza el valor cero, la rutina es ejecutada.

Bytes 4/5: aquí se almacena un valor que se pone a cero cuando la rutina ha sido ejecutada.

Bytes 6/7: almacena la posición del suceso en la cola de espera.

Byte 8: cuenta el número de veces que la rutina ha sido ejecutada.

Byte 9: clase de suceso.

Bytes 10/11: dirección donde se encuentra la rutina (ROM o RAM).

Byte 12: selección de ROMs *(cuáles están habilitadas y cuáles no)*.

Como siempre, hay que advertir al sistema operativo de que vamos a habilitar una interrupción; para ello, hay que inicializar el bloque completo llamando a la rutina firmware &BCEF.

Esta rutina necesita la dirección del bloque del suceso en HL, la dirección de la rutina que hay que ejecutar en DE, la selección de ROMs en C y la clase de suceso en B.

```
                ORG &8000
                INICIALIZA SUCESO
8000:21 28 80   LD HL,eventblock
8003:06 81      LD B,%10000001
8005:11 1D 80   LD DE,event
8008:C3 EF BC   JP &BCEF
                AÑADIR A LA LISTA
800B:21 22 80   LD HL,tickblock
800E:11 32 00   LD DE,50
8011:01 32 00   LD BC,50
8014:C3 E9 BC   JP &BCE9
                BORRAR DE LA LISTA
8017:21 22 80   LD HL,tickblock
801A:C3 EC BC   JP &BCEC
                RUTINA
801D:           .event
801D:3E 07      LD A,7
801F:C3 5A BB   JP &BB5A
                ESPACIO TRABAJO
8022:           .tickblock
8022:           DEFS 6
8028:           .eventblock
8028:           DEFS 7
802F:           END
```

Programa 1

Como nuestra rutina está en RAM, podemos ignorar la selección ROM.

En cuanto a la clase de suceso, la figura 1 muestra el significado de los 8 bytes de este byte.

La rutina va a ir colocada en los 32 k centrales de RAM, por lo que el byte 0 vale 1 *(esta dirección se considera cercana)*.

El suceso es asíncrono, por lo que bytes 0-4 no interesan.

Además de ser asíncrono, el suceso es **«normal»**, no requiere atención **«caiga quién caiga»:** byte 6 a 0.

El byte 7 se pone a 1 para indicar que el suceso es asíncrono.

Las cuatro primeras líneas del programa 1 muestran el código para inicializar nuestro bloque.

Las cuatro siguientes indican cómo un suceso de este tipo puede ser añadido a la lista de espera: HL almacena la dirección del bloque de reloj, DE y BC los temporizadores, el número de veces que la rutina será ejecutada por segundo. Como vale 50, será una vez por segundo exactamente (el contador se decrementa cada 1/50 segundos).

La rutina de interrupción propiamente dicha es lo más simple del mundo: **«imprime»** un CHR$ (7) en pantalla llamando a &BB5A.

Por tanto, al activar la interrupción, el **Amstrad** hará un sonido cada segundo, sin importar el programa que el ordenador esté ejecutando *(probadlo y veréis)*.

Sin embargo, el programa 2 muestra una rutina de interrupción un poco más compleja.

El proceso de inicialización es idéntico, aparte de unos valores de temporización distintos *(5)*.

La rutina examina el teclado y detiene el programa principal que se esté ejecutando, sea Basic o máquina, si se pulsa la tecla TAB. El programa principal continuará si se pulsa la tecla CAPS LOCK.

Esta es una rutina muy, pero que muy útil. No está mal detener nuestro juego favorito cuando suena el teléfono o nos vamos a tomar un algo.

Si la rutina de interrupción se oculta el buffer de las teclas de función *(&B460 más o menos)*, será utilizable con la mayor parte del software comercial. Entonces, será posible recorrer el programa parte a parte y ver cómo funciona.

Para los que no tengan un ensamblador, el programa 3 resuelve la papeleta, colocando el código a partir de &8000.

Para deshabilitar una rutina de interrupción, basta con cargar HL con la dirección del bloque de suceso y llamar a &BCEC.

Después de ensamblar ambas rutinas (programas *1* y *2*), CALL &8000 y CALL &8030 inicializan los sucesos.

CALL &800B y CALL &803B los activan y CALL &8017 y CALL &8047 los desactivan.

El programa 4 puede servir para estudiar los bloque de sucesos. La única condición que debe cumplirse es que la variable «i» en la línea 60 debe apuntar al comienzo del bloque.

Se imprimirán en pantalla los contenidos de los bloque de reloj y de suceso, para que se pueda observar cómo cambian a medida que el sistema operativo ejecuta las rutinas de interrupción.

Recuerde: rutinas de interrupción equivale a aprovechar al máximo la potencia del **Amstrad**, sobre todo para la posibilidad del multitarea y/o gráficos.

PROGRAMAS

```
10 REM PROGRAMA III
20 FOR i=0 TO 90
30 READ byte$
40 POKE &8000+i,VAL("&"+byte$)
60 NEXT
70 DATA 21,28,80,06,81,11,1D,80,C3
80 DATA EF,BC,21,22,80,11,32,00,01
90 DATA 32,00,C3,E9,BC,21,22,80,C3
100 DATA EC,BC,3E,07,C3,5A,BB,00,00
110 DATA 00,00,00,00,00,00,00,00,00
120 DATA 00,00,00
130 REM
140 DATA 21,61,80,06,81,11,4D,80,C3
150 DATA EF,BC,21,5B,80,11,05,00,01
160 DATA 05,00,C3,E9,BC,21,5B,80,C3
170 DATA EC,BC,3E,44,CD,1E,BB,C8,3E
180 DATA 46,CD,1E,BB,28,F9,C9
```

```
10 REM PROGRAMA IV
20 MODE 1:INK 0,0:BORDER 0
30 PEN 1:PAPER 3:LOCATE 9,3
40 PRINT " BLOQUES DE SUCESO Y RELO
J "
50 PAPER 0
60 i=&8022:j=i+2
70 LOCATE 1,7
80 PRINT TAB(9)"Dir. sig. bloque";:
GOSUB 190
90 PRINT TAB(9)"Contador1";:GOSUB 1
90
100 PRINT TAB(9)"Contador2";:GOSUB
190
110 PRINT TAB(9)"Pos_en_cola";:GOSU
B 190
120 PRINT TAB(9)"Llamadas";:GOSUB 1
80
130 PRINT TAB(9)"Clase";:GOSUB 180
140 PRINT TAB(9)"Dir. rutina";:GOSU
B 190
150 PRINT TAB(9)"ROM";:GOSUB 180
160 PEN 3
170 LOCATE 28,9:PRINT "&";HEX$(PEEK
(j)+256*PEEK(j+1),4):GOTO 170
180 GOSUB 200:PRINT "&";HEX$(PEEK(i
),2):PRINT:i=i+1:PEN 1:RETURN
190 GOSUB 200:PRINT "&";HEX$(PEEK(i
)+256*PEEK(i+1),4):PRINT:i=i+2:PEN
1:RETURN
200 PEN 2:WHILE POS(#0)<28:PRINT ".
";:WEND:PEN 3:RETURN
```

```
                        ORG &8030
                        INICIALIZA SUCESO
8030:21 61 80           LD HL,eventblock
8033:06 81              LD B,%10000001
8035:11 4D 80           LD DE,event
8038:C3 EF BC           JP &BCEF
                        AÑADIR A LA LISTA
803B:21 5B 80           LD HL,tickblock
803E:11 05 00           LD DE,5
8041:01 05 00           LD BC,5
8044:C3 E9 BC           JP &BCE9
                        BORRAR DE LA LISTA
8047:21 5B 80           LD HL,tickblock
804A:C3 EC BC           JP &BCEC
                        RUTINA
804D:                   .event
804D:3E 44              LD A,68
804F:CD 1E BB           CALL &BB1E
8052:C8                 RET Z
8053:                   .loop
8053:3E 46              LD A,70
8055:CD 1E BB           CALL &BB1E
8058:28 F9              JR Z,loop
805A:C9                 RET
                        ESPACIO TRABAJO
805B:                   .ticblock
805B:                   DEFS 6
8061:                   .eventblock
8061:                   DEFS 7
8068:                   END
```

Programa 2

MATCH DAY

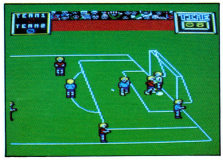
Mr. Joystick se siente hoy deportivo a tope y presenta dos programas de juego en equipo y alta rivalidad. El fútbol, el deporte rey en toda Europa es uno de ellos, el otro es el deporte más popular en USA, el béisbol.

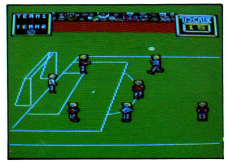
El estadio está lleno hasta los topes, el público enfervorizado espera la aparición de los jugadores en el terreno de juego, los equipos salen de los vestuarios, el partido va a empezar.

El Amstlético de Software, tiene en posesión el balón, su delantero centro, Pepitobit hace un par de regates, dejando clavados a sus contrarios, centra hacia el extremo izquierdo a Gustavoleft, el cual tira sobre la hoya donde agazapado espera nuestro tigre Edugool, que se eleva entre la maraña de defensores, conectando uno de los mejores cabezazos de su vida. **¡GOOL!,** el público ruge en los graderíos, el estadio se viene abajo.

Match Day, es un programa de fútbol, pero de fútbol de verdad, nuestra primera opción es elegir el nombre y color de los equipos.

Las posibilidades de juego permiten jugar contra el ordenador o contra otra persona, con la inclusión de una competición de Copa en la que se pueden enfrentar hasta ocho jugadores distintos, incluyendo los sorteos de los cuartos de final y semifinales.





Estos son pequeños detalles que dicen mucho de un programa, pero lo más importante viene ahora; las técnicas futbolísticas.

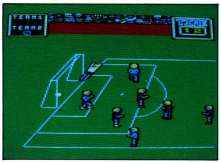
En efecto, este juego refleja todas las posibilidades del deporte rey:

Regate, centro, parada del balón, tiro de cabeza, chut, saque de banda, córner, saque de centro y control del portero.

Regate, se realiza automáticamente cuando un jugador está en posesión del esférico, por supuesto un jugador con balón corre menos que uno que no lo lleva.

Centro, para centrar a un compañero, existen dos posibilidades: a ras de suelo, cuando el jugador está parado y bombeado; hay ocho direcciones distintas de tiro.

Parar el balón, importantísimo a la hora de recoger los centros de nuestros compañeros, la recepción del balón debe hacerse en el momento justo, más tarde éste nos rebotaría en las piernas.



Tiro de cabeza, es una utilización ventajosa de la anterior técnica, aprovechando el rebote de éste en la cabeza del jugador.

Chut, con ocho direcciones posibles y tres grados de fuerza, dependiendo del impulso que lleve el jugador.

Saque de banda, se efectúa con las manos (como de verdad) con la posibilidad de ocho direcciones de tiro.

Córner, con la posibilidad de tiro raso y bombeado.

Saque de centro, donde el balón debe cruzar la línea de centro. Control de portero, cuando a éste se le cambien el color de las botas, está en tu poder, puedes hacerle tirarse a derecha o izquierda.

Con todos estos controles y técnicas de juego a utilizar, el programa refleja fielmente lo que constituye el dominio del balón y el conocimiento del fútbol, está muy bien estudiado y el manejo de jugadores se realiza mediante teclado o joystick, lo que permite manejar todas las técnicas fácilmente cuando se tiene un poco de práctica. Adelante futboleros.

Compatible: CPC464, CPC664 y CPC6128.

BASEBALL

Mr. Joystick

De la vieja Europa pasamos a Norteamérica, la tierra del progreso, la civilización, tecnología y el feudo de Mr. Dólar.

Al otro lado del Atlántico, cambian las costumbres y por supuesto, también los deportes; sin ninguna duda el béisbol es el deporte más popular, siendo la liga nacional la más prestigiosa en el mundo entero.

¿Qué es el béisbol? Un juego de pelota en el que un equipo ataca intentando batear, es decir, golpear la bola con un palo y el otro defiende intentando que el bateador no golpee la pelota o atrapando ésta al vuelo o centrándola a las bases para intentar eliminar a los contrarios que se encuentran en una base.

El objetivo es apuntar el máximo número de carreras posibles.

Para obtener una carrera el jugador debe recorrer las tres bases, llegando al Home sin ser eliminado.

Los no entendidos en béisbol, no tienen que preocuparse de reglamentos ni mecánica de juego, solamente observar a la máquina y darle fuerte a la pelota en la fase de ataque, intentando atraparla al vuelo y lanzándola a las bases antes de que lleguen los jugadores contrarios en la de defensa.

Sofisticaciones como robar bases y lanzamientos de bola con efecto, se aprenden con la práctica y leyendo las instrucciones del programa.

Las técnicas de juego:

Pitcher, en los lanzamientos de bola se puede graduar la fuerza y la altura, pudiendo utilizar hasta 8 tiros distintos: bola baja, alta, lenta, rápida, alta y rápida, alta y lenta, baja y rápida, baja y lenta.

Bateador, en el momento de batear se puede dirigir la bola, en caso de que acertemos a golpearla hacia el lado del campo que más nos interese, con sólo retardar levemente el momento del golpe o acelerarlo.

Bases, los jugadores de las bases deben recibir la bola en ellas, para intentar eliminar a los contrincantes que corren hacia su base.

La bola puede circular desde los jugadores de campo hacia cualquier base, desde el pitcher hacia una base, de una base a otra y desde el catcher al pitcher o a cualquier base. Todas las posibilidades están reflejadas.

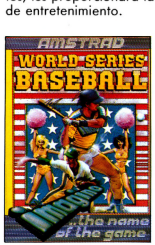
Jugadores de campo, deben atrapar la pelota al vuelo para eliminar al bateador y luego pasarla a las bases o al pitcher.

Corredores, deben intentar una vez la pelota esté en juego, alcanzar la siguiente base sin ser eliminados, cabe la posibilidad de robar bases cuando el pitcher está distraído o lanza la pelota.

Strike, fallo del bateador que al intentar golpear la pelota no le acierta o que da por mala una pelota que era buena (una pelota es buena cuando pasa por encima del diamante del Home y a una altura comprendida entre las rodillas y los hombres del bateador, tres strikes traen como consecuencia la eliminación del bateador.

Bola, cuando el pitcher lanza la pelota fuera de estos límites fijados para la bola buena, cuatro bolas implican el paso a primera base del bateador.

Home run, se produce cuando un bateador golpea la pelota lanzándola fuera de los límites del campo sin que ésta bote, equivale a una carrera por el bateador más una por cada hombre que se encuentre en una base. Los amantes y aficionados a este popular deporte americano tienen en este programa un pasatiempo sensacional, con multitud de matices que permiten reproducir las más sofisticadas tácticas de juego, que una vez dominada la técnica de movimiento de jugadores y lanzamientos, les proporcionará largas horas de entretenimiento.



Compatible: CPC1664, CPC1664 y CPC6128.

Análisis

ECUACIONES DE SEGUNDO GRADO

En nuestros años de estudio, todos nos hemos encontrado con las inefables ecuaciones. Resolver una no es difícil, pero el problema empieza a ser preocupante cuando tenemos que resolver unas 10 ó 15, o cuando algunas tienen soluciones imaginarias.

En esta ocasión Análisis te ayuda a resolver cualquier ecuación de segundo grado, bien tenga soluciones reales o imaginarias, con el único esfuerzo de introducir los términos en X2X, y el término independiente.

10,20 Título del programa.

30 Limpia la pantalla.

40 INPUT del coeficiente en X2, que se almacena en la variable a.

50 Condición que comprueba si el término introducido en la variable a es igual a cero, en ese caso el programa vuelve a la línea 30 pidiéndonos de nuevo este dato.

Una ecuación con el coeficiente de segundo grado en cero tiene soluciones infinitas.

60 INPUT del coeficiente en X, éste se aloja en la variable b.

70 INPUT del término independiente, su valor queda almacenado en la variable c.

Se han elegido las variables a, b, c, para reflejar la forma típica de una ecuación de segundo grado; $ax2+bx+c=0$.

80 Define una cadena alfanumérica h$ que combina los valores de a,b,c, con los términos X2+,X+,=0, de forma que al representar h$ aparezca la ecuación.

90 Escribe en pantalla la palabra ECUACION y la variable h$.

100 Calcula el discriminante de la ecuación; D=b2—4a=.

110 Crea una bifurcación del programa de forma que si el discriminante es mayor o igual que cero pasa a la rutina 160.

Si el discriminante es menor que cero, el programa pasa a la rutina de la línea 230.

Esta bifurcación se ha realizado para distinguir entre soluciones reales; discriminante $>=0$ o soluciones imaginarias; discriminante <0.

120 Representa en pantalla el mensaje ** Pulsar una tecla para otra ecuación **.

130 Llamada a una rutina del firmware que hace que el ordenador espere a que se pulse una tecla.

140 Una vez pulsada la tecla el programa vuelve a la línea 30, para resolver una nueva ecuación.

150 END que separa el programa principal de las rutinas.

160-120 Rutina que resuelve las ecuaciones con soluciones reales.

170 Calcula la primera solución $X1 = \frac{-b+ \quad DISCRIMINANTE}{2a}$

180 Calcula la solución $X2 = \frac{-b- \quad DISCRIMINANTE}{2a}$

190 Representa el mensaje ** Raíces reales **.

200,210 Escriben las soluciones X1, X2.

220 RETURN que devuelve el control a la línea 120.

230-290 Rutina para resolver las ecuaciones con raíces imaginarias.

240 Cambia el valor del discriminante por su valor absoluto; de esta manera podremos extraer su raíz cuadrada.

250 Calcula el término $z=-b/2*a$.

260 Escribe el mensaje ** Raíces imaginarias **.

270,280 Representan en pantalla las dos soluciones de la ecuación en el formato:

$X1 = \frac{-b}{2a} +$ discriminante. i/2a

$X2 = \frac{-b}{2a} -$ discriminante. i/2a

290 —Devuelve el control del programa a la línea 120.

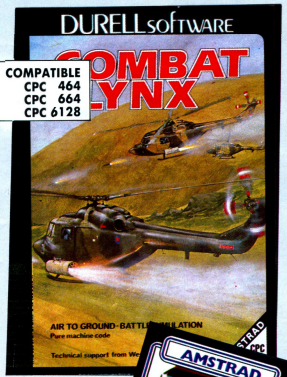
Suscríbete... y uno de estos tres sensacionales juegos será tuyo... ¡GRATIS!

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

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

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

OFERTA VALIDA SOLAMENTE PARA ESPAÑA



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

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

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

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

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

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

FRUTTIES

Frutties es un juego tipo arcade rápido y lleno de color. La acción tiene lugar en una factoría que ha sido invadida por unos monstruos llamados Fruttie. Por el momento, han destruido toda la maquinaria de la fábrica y ahora les ha llegado el turno a los trabajadores.

COMPATIBLE
CPC 464
CPC 664
CPC 6128

Todas las puertas y ventanas de las habitaciones han sido selladas para impedirles escapar y hacer más daño.

Nuestra tarea es aniquilarlos habitación por habitación.

Por razones desconocidas, a las frutties esta perspectiva no les hace mucha gracia, así que intentarán convertirte en su cena rápidamente: esto les parece mucho más lógico por razones esta vez obvias.

La única forma de acabar con un fruttie es cavar un agujero en su camino y atraerle para que caiga en él. ¡Cuidado!, si tú caes en él, no lo cuentas.

Existen varias pantallas con distinto número de bichos y niveles crecientes de dificultad.

La última advertencia: el programa incorpora una parte de código máquina, por lo que una vez tecleado íntegro, SALVALO EN CINTA/DISCO antes de ejcutarlo. Un error de tipografía sería fatal.

VARIABLES PRINCIPALES

x,y	Coordenadas del hombre
b (15,1)	Coordenadas de los monstruos
d(15)	Frutty vivo o muerto
dead	Hombre vivo o muerto
air	El aire restante en una habitación
screen	Número de pantalla
lives	Vidas que quedan
m (21,25)	Mapa de pantalla

RUTINA GRAFICA

Llamadas desde Basic: líneas 280, 370, 380, 390, 430, 620, 630. 1.000, 1.040.
Parámetros admitidos por la llamada CALL desde Basic: 6
Dirección de carga: &A000

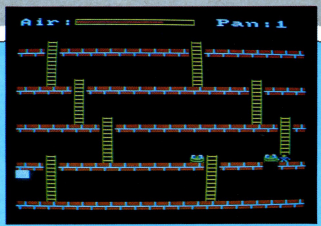
Dirección de arranque: &A000
Longitud en bytes: 80
Registros utilizados: 80
Registros modificados: todos

DIRECCION	OPCODES	MNEMONICOS
A000		ORG #A000
A000	26C0	LD H,#C0
A002	DD7E0A	LD A,(IX+10)
A005	3D	DEC A
A006	B7	ADD A,A
A007	B7	ADD A,A
A008	6F	LD L,A
A009	115000	LD DE,#0050
A00C	DD4608	LD B,(IX+8)
A00F	05	DEC B
A010	19	LA010: ADD HL,DE
A011	10FD	DJNZ LA010
A013	DD7E06	LD A,(IX+6)
A016	CD2FA0	CALL LA02F
A019	26C0	LD H,#C0
A01B	DD7E04	LD A,(IX+4)
A01E	3D	DEC A
A01F	B7	ADD A,A
A020	B7	ADD A,A
A021	6F	LD L,A
A022	115000	LD DE,#0050
A025	DD4602	LD B,(IX+2)
A028	05	DEC B
A029	19	LA029: ADD HL,DE
A02A	10FD	DJNZ LA029
A02C	DD7E00	LD A,(IX+0)
A02F	B7	LA02F: ADD A,A
A030	B7	ADD A,A
A031	B7	ADD A,A
A032	B7	ADD A,A
A033	B7	ADD A,A
A034	5F	LD E,A
A035	16A1	LD D,#A1
A037	060B	LD B,#0B
A039	1A	LA039: LD A,(DE)
A03A	77	LD (HL),A
A03B	13	INC DE
A03C	23	INC HL
A03D	1A	LD A,(DE)
A03E	77	LD (HL),A
A03F	13	INC DE
A040	23	INC HL
A041	1A	LD A,(DE)
A042	77	LD (HL),A
A043	13	INC DE
A044	23	INC HL
A045	1A	LD A,(DE)
A046	77	LD (HL),A
A047	13	INC DE
A048	7B	LD A,B
A049	01FD07	LD BC,#07FD
A04C	09	ADD HL,BC
A04D	47	LD B,A
A04E	10E9	DJNZ LA039
A050	C9	RET

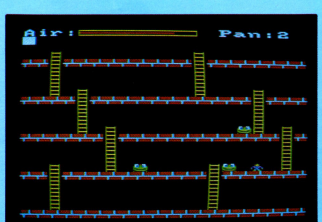
Nota: *las etiquetas hacen referencia a la posición de memoria donde están colocadas. Todas van precedidas de una «L».*


```
10 REM Frutties
20 REM By R.A.Waddilove
30 REM(c)AMSTRAD SEMANAL
40 GOSUB 910:REM Inicializa
50 GOSUB 1570:REM Instrucciones
60 WHILE NOT(TRUE AND FALSE)
70 lives=3:screen=0
80 WHILE lives
90 GOSUB 460:REM Dibuja Pantalla
100 GOSUB 990:REM Comienzo
110 WHILE NOT dead AND k<n AND air>
134
120 GOSUB 200
130 GOSUB 310
140 WEND
150 IF dead THEN GOSUB 1390 ELSE GO
SUB 1490
160 WEND
170 GOSUB 1940:REM Final Juego
180 WEND
190 END
200 REM ----- Mueve Hombre -----
210 PLOT air,390,0:PLOT air,392,0:a
ir=air-1:IF air<136 THEN dead=TRUE:
RETURN
220 IF m(x,y)=4 THEN dead=TRUE:RETU
RN ELSE IF m(x,y+1)=4 THEN xx=x:yy=
y+1:GOTO 280
230 IF INKEY(70)>-1 AND x>1 THEN LO
CATE x-1,y+1:PRINT " ":m(x-1,y+1)=4
:RETURN
240 IF INKEY(30)>-1 AND x<19 THEN L
OCATE x+1,y+1:PRINT " ":m(x+1,y+1)=
4:RETURN
250 xx=x+(INKEY(39)>-1)-(INKEY(31)>
-1):IF m(xx,y)<2 THEN xx=x
260 yy=y+(INKEY(69)>-1)-(INKEY(71)>
-1):IF m(xx,yy)<2 THEN yy=y
270 IF x=xx AND y=yy THEN RETURN
280 IF m(x,y)=2 THEN CALL &A000,x,y
,3,xx,yy,1 ELSE CALL &A000,x,y,4,xx
,yy,1
290 SOUND 130,500,10,10:x=xx:y=yy
300 RETURN
310 REM ----- Mueve Bichos -----
320 FOR i=0 TO n
330 IF d(i) THEN GOSUB 360
340 NEXT
350 RETURN
360 REM ----- Bichos -----
370 bx=b(i,0):by=b(i,1):IF bx=x AND
 by=y THEN dead=TRUE:CALL &A000,x,y
,5,x,y,5:RETURN
380 IF m(bx,by)=4 THEN k=k+1:CALL &
A000,bx,by,4,bx,by,4:d(i)=FALSE:RET
URN
390 IF RND>0.3 THEN CALL &A000,bx,b
y,0,bx,by,0:RETURN
400 IF m(bx,by+1)=4 THEN h=5:xx=bx:
yy=by+1:SOUND 132,50,100,10,0,1:GOT
O 430
410 h=0:SOUND 129,100*n,5,10:xx=bx+
(x<bx)-(x>bx):IF m(xx,by)<2 THEN xx
=bx
420 yy=by+(y<by)-(y>by):IF m(xx,yy)
<2 THEN yy=by
430 b(i,0)=xx:b(i,1)=yy:IF m(bx,by)
=2 THEN CALL &A000,bx,by,3,xx,yy,h
ELSE CALL &A000,bx,by,4,xx,yy,h
440 IF xx=x AND yy=y THEN dead=TRUE
:CALL &A000,x,y,5,x,y,5
450 RETURN
460 REM ----- Pantalla -----
470 screen=screen+1
480 MODE 0:BORDER 0
490 RESTORE 530:FOR i=0 TO 15:READ
j:INK i,j:NEXT
500 IF screen=3 THEN BORDER 1:INK 0
,1
510 IF screen=4 THEN BORDER 3:INK 0
,3
520 IF screen>4 THEN BORDER 4:INK 0
,4:INK 1,6:INK 3,3:INK 4,21:INK 11,
25
530 DATA 0,15,23,6,24,26,8,2,11,20,
13,18,0,0,0,16
540 PEN 5:PRINT "Air:";TAB(15)"Pan:
";MID$(STR$(screen),2)
550 MOVE 130,396:DRAW 398,396,4:DRA
W 398,386:DRAW 130,386:DRAW 130,396
:MOVE 134,392:DRAW 394,392,15:DRAW
394,390:DRAW 134,390
560 RESTORE 670
570 FOR i=3 TO 25
580 READ screen$
590 FOR j=1 TO 20
600 a$=MID$(screen$,j,1)
610 IF a$="." THEN m(j,i)=0
620 IF a$="=" THEN m(j,i)=1:CALL &A
000,j,i,2,j,i,2
630 IF a$="I" THEN m(j,i)=2:CALL &A
000,j,i,3,j,i,3
640 IF a$="*" THEN m(j,i)=3
650 NEXT
660 NEXT
670 DATA ....................
680 DATA **I**********I*******
690 DATA ==I=========I=======
700 DATA ..I.........I.......
710 DATA ..I.........I.......
720 DATA ..I.........I.......
730 DATA **I**I*******I***I***
740 DATA ====I==========I===
750 DATA ....I..........I...
760 DATA ....I..........I...
770 DATA ....I..........I...
780 DATA ****I*I**********I*I*
790 DATA ======I==========I=
800 DATA ......I..........I.
810 DATA ......I..........I.
820 DATA ......I..........I.
830 DATA **I***I******I****I*
840 DATA ==I==========I======
850 DATA ..I.........I.......
860 DATA ..I.........I.......
870 DATA ..I.........I.......
880 DATA **I**********I******
890 DATA ====================
```

```
900 RETURN
910 REM ------- Inicializa -------
920 MEMORY &9FFF
930 ENT 1,200,1,1
940 DEFINT a-z
950 DIM m(21,25),b(15,1),d(15)
960 TRUE=-1:FALSE=0
970 GOSUB 1080:REM Machine code
980 RETURN
990 REM ----- Comienzo -----
1000 x=5:y=24:CALL &A000,x,y,1,x,y,
1
1010 n=screen
1020 FOR i=0 TO n
1030 b(i,0)=1+INT(RND*20):b(i,1)=4+
INT(RND*4)*5
1040 CALL &A000,b(i,0),b(i,1),0,b(i
,0),b(i,1),0:d(i)=TRUE
1050 NEXT
1060 k=-1:air=394:dead=FALSE
1070 RETURN
1080 REM ----- Codigo -----
1090 RESTORE 1130
1100 FOR i=0 TO 80
1110 READ a$:POKE &A000+i,VAL("&"+a
$)
1120 NEXT
1130 DATA 26,C0,DD,7E,0A,3D,87,87,6
F,11,50,00,DD,46,08,05,19,10,FD,DD,
7E,06,CD,2F,A0,26,C0,DD,7E,04,3D,87
,87,6F,11,50,00,DD,46,02,05,19,10,F
D,DD,7E,00,87,87,87,87,5F,16,A1,
06,08,1A,77,13,23,1A,77,13,23,1A,77
,13,23,1A,77,13,78,01,FD
1140 DATA 07,09,47,10,E9,C9
1150 FOR i=0 TO 191:READ j:POKE &A1
00+i,j:NEXT
1160 REM alienigena
1170 REM Filas=8/Columnas=4
1180 DATA 80,228,216,160,216,237,22
2,228,218,237,222,229,207,207,207,2
07,138
1190 DATA 207,207,69,207,0,0,207,69
,207,207,138,138,0,0,69
1200 REM Hombre
1210 REM Filas=8/Columnas=4
1220 DATA 0,16,0,0,0,48,32,0,0,16,0
,0,20,124,60,0,40,124,40,40,32,65,0
,32,0
1230 DATA 130,130,0,69,138,207,0
1240 RETURN
1250 REM Ladrillo
1260 REM FIlas=8/Columnas=4
1270 DATA 192,192,132,192,192,192,1
```



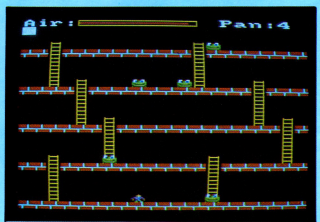

```
32,192,192,192,132,192,192,192,132,
192,12
1280 DATA 12,12,12,192,72,192,192,1
92,72,192,192,192,72,192,192
1290 REM Escalerilla
1300 REM FIlas=8/Columnas=4
1310 DATA 16,0,0,32,16,48,48,32,16,
0,0,32,16,0,0,32,16,0,0,32,16,48,48
,32,16
1320 DATA 0,0,32,16,0,0,32
1330 REM Espacio
1340 DATA 0,0,0,0,0,0,0,0,0,0,0,0,0
,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
0,0
1350 REM alien
1360 REM FIlas=8/Columnas=4
1370 DATA 80,228,220,160,216,237,22
2,228,218,237,218,229,207,207,207,2
07,207
1380 DATA 138,138,207,207,69,69,207
,69,207,207,138,138,0,0,69
1390 REM ----- Muerte -----
1400 lives=lives-1:screen=screen-1
1410 FOR i=0 TO 2000:NEXT
```



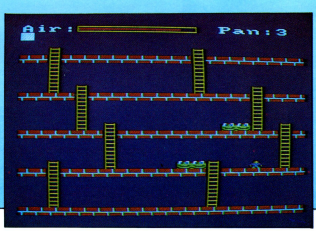

```
1420 RESTORE 1480
1430 FOR i=1 TO 6:READ j:SOUND 1,j,
20,12:NEXT
1440 SOUND 1,478,100,12
1450 IF lives=0 THEN LOCATE 7,12:PR
INT CHR$(22);CHR$(1);"GAME OVER";CH
R$(22);CHR$(0)
1460 FOR i=0 TO 3000:NEXT
1470 RETURN
1480 DATA 956,758,638,851,716,568
```

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.

```
1490 REM ----- Borrar Pantalla ----
-
1500 FOR i=0 TO 2000:NEXT
1510 RESTORE 1560
1520 FOR i=1 TO 15:READ j:SOUND 1,j
,20,12:NEXT
1530 SOUND 1,119,100,12
1540 FOR i=0 TO 3000:NEXT
1550 RETURN
1560 DATA 60,63,60,63,60,71,63,80,7
1,89,80,95,89,106,95
1570 REM ----- Instrucciones -----
1580 MODE 1:CALL &BC02:CALL &BB4E:I
```




```
NK 0,0:BORDER 0
1590 PEN 2:INK 2,0:LOCATE 1,25:PRIN
T "Frutties!":PLOT -10,-10,1
1600 FOR i=0 TO 140 STEP 2
1610 FOR j=0 TO 16 STEP 2
1620 IF TEST(i,j) THEN PLOT 180+i*2
,350+j*2:PLOT 180+i*2,352+j*2:PLOT
182+i*2,350+j*2:PLOT 182+i*2,352+j*
2
1630 NEXT
1640 NEXT
1650 LOCATE 1,25:PRINT SPC(10):INK
2,20
1660 MOVE 0,0:DRAW 0,398,1:DRAW 638
,398:DRAW 638,0:DRAW 0,0
1670 WINDOW #0,2,38,6,24
1680 PEN 2:PAPER 0:PRINT "Puedes li
brar la fabrica de Frutty  Monster
s? Parecen estar en cualquier sitio
 causando estragos."
```




```
1690 PRINT:PRINT "La fabrica se ha
cerrado para evitar su escapada, po
r lo que debes llevar bombona de ox
igeno para respirar."
1700 PRINT:PRINT "Para librarte de
un Frutty, cava un  agujero y el Fr
utty caera en el."
1710 PEN 3:PRINT:PRINT " A=Subir
 Z=Bajar    <=Izquierda   >=Derecha
":PRINT:PRINT "      Caps Lock=Cava
un agujero a la  Izquierda":PRINT:P
RINT"      ?=Cava un agujero y a la de
recha"
1720 GOSUB 1770
1730 WINDOW #2,2,10,2,24:WINDOW #3,
11,20,2,24:WINDOW #4,21,30,2,24:WIN
DOW #5,31,39,2,24
1740 LOCATE #3,1,23:LOCATE #5,1,23
1750 FOR i=1 TO 25:PRINT #2,CHR$(11
);:PRINT #4,CHR$(11);:PRINT #3,CHR$
(10);:PRINT #5,CHR$(10);:NEXT
1760 RETURN
1770 REM ----- Scroll -----
1780 RESTORE 1900
1790 PEN #1,1:PAPER #1,3
1800 a$="Microhobby Amstrad Semanal
.....  "
1810 b$=a$
1820 SOUND 2,239,10000,10:SOUND 4,3
19,10000,10
1830 WHILE INKEY$<>"":WEND
1840 WHILE INKEY$="":READ j:IF j=0
THEN RESTORE 1900:READ j
1850 SOUND 1,j,15,12:LOCATE #1,6,24
:PRINT #1,MID$(b$,1,30):b$=MID$(b$,
2):IF LEN(b$)<31 THEN b$=b$+a$
1860 WEND
1870 SOUND 129,0,0,0:SOUND 130,0,0,
0:SOUND 132,0,0,0
1880 x=FRE("")
1890 RETURN
1900 DATA 60,63,60,63,71,63,71,80,7
1,80,89,80,89,95,89,95,106,95,106,1
19,106
1910 DATA 239,213,190,179,159,142,1
27,119
1920 DATA 190,179,159,142,127,119,1
06,95, 159,142,127,119,106,95,89,80
1930 DATA 0
1940 REM -----Puntuacion-----
1950 MODE 1:INK 0,13:INK 1,26:INK 2
,20:INK 3,2
1960 DRAW 0,398,1:DRAW 638,398:DRAW
 638,0:DRAW 0,0
1970 LOCATE 16,5:PEN 2:PRINT "Bien
Hecho!":PEN 3:LOCATE 11,10:PRINT "P
antalla Numero";screen+1
1980 PEN 1:LOCATE 7,15:PRINT "Pulsa
 una tecla para continuar"
1990 GOSUB 1770:REM scroll
2000 RETURN
```

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 A

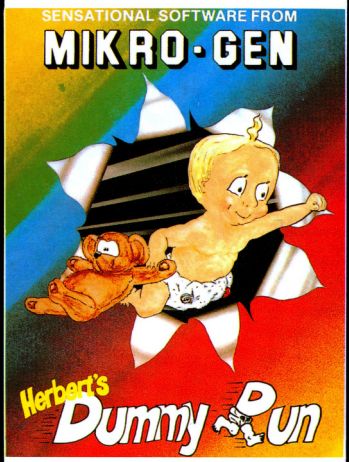
DRAGONTORC

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.

DISTRIBUIDOR EXCLUSIVO DE ESPAÑA: ERBE SOFTWARE

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.°1



DUMMY RUN

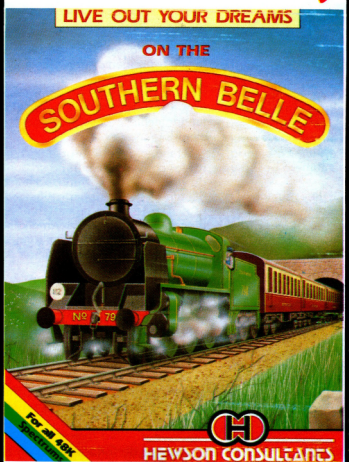
POR FIN EN AMSTRAD LAS AVENTURAS DEL PEQUEÑO DE LA FAMILIA WALLY. UN JUEGO QUE TE SORPRENDERA POR LA GRAN CANTIDAD Y CALIDAD DE SUS GRAFICOS. TODO PUEDE PASAR EN ESTE FANTASTICO PROGRAMA QUE SUPERA LOS LIMITES DE LA IMAGINACION Y QUE SE HA CONVERTIDO EN UN CLASICO DE LOS JUEGOS DE ORDENADOR.

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.

E STA. ENGRACIA, 17 - 28010 MADRID. TEL. 447 34 10

SALTOS RELATIVOS

Javier Igual

Ya sabemos cómo tomar decisiones en código máquina, inspeccionando el estado de los flags.

Consecuencia lógica de esto es la existencia de órdenes que permiten bifurcar a determinados segmentos del programa, en función de dichos flags.

El Z80 pasa dos tipos de órdenes de salto, JP y JR, salto absoluto y relativo. Veamos cómo funcionan.

Vamos a aprender algo más acerca de la representación de números en código máquina.

Si echamos una mirada al programa 1, se puede ver que su comprensión no reviste mucha dificultad: sumamos 6 a &FF (255) y almacenamos el resultado en &2FF8.

Al examinar &2FF8, encontraremos que contiene 5, pues, en binario 255+1+5 equivale a 0+5.

DIRECCION	OPCODE	MNEMONICO
3000	3E 06	LD A, &06
3002	C6 FF	ADD A, &FF
3004	32 F8 2F	LD (&2FF8), A
3007	C9	RET

Programa 1

Restar en binario

Es decir, cada vez que sumemos &FF al acumulador, obtendremos como resultado un número menos que el que había inicialmente; resulta que, según la aritmética binaria, para restar 1 a un número basta sumarle &FF (!).

Y para restar 2? Pues sumar &FE *(254)* y así sucesivamente:

30+254=28+2+254=28+0=28

El programa 2 muestra el código de lo antedicho.

Estas características de los número binarios pueden servir para conseguir un rudimentario sistema de numeración de números, valga la redundancia, negativos.

Hasta ahora todos los números que hemos tratado han sido positivos, pero a veces es útil tener a nuestra disposición sus homónimos negativos *(los encuentro particularmente útiles en lo relativo a mi cuenta bancaria)*.

Ya se ha visto lo que pasa al sumar &FF a un número positivo: su valor desciende en una unidad. Por ello, no es ninguna tontería considerar &FF como —1 desde el punto de vista del código máquina, &FE como —2, etc.

En función del contexto

Observe la palabra **«considerar»**; lo que proponemos es algo completamente arbitrario, y el que &FF sea «negativo» depende del contexto en el que nos estemos moviendo. Afortunadamente, casi siempre podemos elegirlo.

Esta idea debería usarse para mirar las cosas desde más de un punto de vista; después de todo, ya hemos aceptado que 10 puede significar, eso, 10 ó 16 según que estemos hablando de un número decimal o hexadecimal, esto es, según el **contexto** que arbitrariamente damos por válido.

Todas estas implicaciones al **Amstrad** le tienen sin cuidado; funcionar, funciona, y bien.

El, ni corto ni perezoso, transforma **«3-4»** en **«3+252»**, o sea, 255, o sea —1. ¡Correcto!

Para nosotros el asunto no es tan simple, ya que, siguiendo el razonamiento anterior hasta **«sus últimas consecuencias»**, resulta que &01 equivale a —255 *(piénselo)*.

Después de todo este lío, no hemos dejado numerosos positivos con los que jugar.

Código máquina

Más convenios

Para cortar el nudo gordiano, es necesario hacer un convenio más, arbitrario como de costumbre: los números desde &FF a &80 representarán, a partir de ahora *(según el contexto, insisto)* los números negativos desde —1 a —128, mientras que &0 a &7F serán **«positivos»**, de 0 a 127 *(figura 1, les juro)*.

Positivo		Negativo	
0	&0	—1	&FF
+1	&1	—2	&FE
+2	&2	—3	&FD
"	"	"	"
"	"	"	"
"	"	"	"
+126	&7E	—127	&81
+127	&7F	—128	&80

Figura 1: Representación de los números negativos

JR	&18
JR NZ	&20
JR Z	&28
JR NC	&30
JR C	&38

Tabla 1: Opcodes de JR

En definitiva, seguimos usando un solo byte para representar 256 números, pero, en vez de ser todos positivos, ahora la mitad son negativos.

Una ventaja de este acuerdo es que, a nivel binario, todos los **«negativos»** tienen su byte número 7 puesto a 1, lo cual resultará extremadamente útil en un futuro próximo.

Esto parece complicado, y lo es, al menos hasta que se coge un poco de práctica. Por ahora, olvidémoslo a nivel aritmético y centrémonos en la utilidad de lo que acabamos de ver para los saltos relativos.

Echad una mirada al programa 3. Simplemente saltamos a la posición &3006 y RETornamos. De paso, la instrucción NOP *(del inglés NO Operation)* le dice al micro, literalmente: **«haz nada»**. Simplemente están puestas para tener un sitio al que saltar *(JP)*.

El programa 4 hace exactamente lo mismo, pero usa un salto relativo *(mnemónico JR, opcode 18)* en lugar de un JP.

El contador de programa

Antes de profundizar en esto, vamos a tratar de aclarar lo que es un contador de programa.

DIRECCION	OPCODE	MNEMONICO
3000	3E 1E	LD A,30
3002	C6 FF	ADD A, &FE
3004	32 F8 2F	LD (&2FF8), A
3007	C9	RET

Programa 2

Técnicamente hablando, es un registro interno del Z80 que almacena una dirección indicando dónde está el byte o bytes de código máquina que se está ejecutando en ese momento.

Después de todo, el lenguaje máquina no tiene números de línea como Basic, así que el contador de programa suple esta falta usando el equivalente de un número de línea Basic: una dirección de memoria.

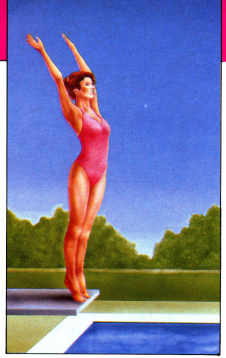
De una forma algo menos rigurosa, pero más intuitiva y útil, podemos decir que el contador de programa

DIRECCION	OPCODE	MNEMONICO
3000	C3 06 30	JP &3006
3003	00	NOP
3004	00	NOP
3005	00	NOP
3006	C9	RET

Programa 3

apunta a la dirección de memoria del siguiente código de operación a ser ejecutado por el microprocesador.

Normalmente, el Z80 va incrementando el contador de programa paso a paso, sumándole los bytes requeridos por cada opcode... hasta que se encuentran con una instrucción JP.



Código máquina

Ordenes JP y JR

En ese momento, los dos bytes siguientes a la instrucción, que especifican la dirección de salto, son cargados por el Z80 en el contador de programa, y la ejecución continúa allí. Por eso las órdenes de salto se llaman así. Literalmente, el micro **«salta»** por encima de un bloque de memoria y se dirige a otro lugar completamente distinto.

En el programa 4, hemos hecho precisamente esto, pero usando una forma más corta, en ocupación de memoria, de decirle al Z80 que salte: el salto relativo.

En lugar de ordenar: **«dirígete inmediatamente a la dirección &3006»,** el micro entiende que debe dirigirse a una posición de memoria que está un cierto número de bytes más allá de la actual posición del contador de programa *(sí, por eso se llaman relativos)* especificando por la cifra o etiqueta que sigue a la orden JR.

En el famoso programa 4, el opcode 18 es seguido por un 3. **¿Quiere esto decir que saltamos de &3000 a &3000+3?**

Por supuesto que... no. Recuérdese que el contador de programa apunta al SIGUIENTE código de operación que va a ser ejecutado, en este caso el primer NOP *(&3002)*, por lo que &3002+3=&3005. Ahí salta el programa, y ahí debe saltar.

El truco de los saltos relativos están en calcular la dirección de destino, es decir, el offset que hay que colocar después del mnemónico JR.

Para ello, una buena técnica es escribir la orden como:

```
JR_ _ _ _ _ _
```

luego, comenzando en el byte siguiente a estos 2, contar de uno en uno hasta llegar al sitio donde queremos saltar.

Una vez averiguado el valor, se pone detrás del mnemónico JR y listo.

Si podemos realizar saltos relativos hacia adelante, también podemos efectuarlos hacia atrás, como en el programa 5, y aquí hacen su entrada triunfal los números negativos.

Después de que el Z80 encuentra la orden JR, el contador de programa apunta a la orden RET *(&3005)*.

Recordando la regla anteriormente expuesta, son necesarios 5 bytes *(—5 bytes)* para saltar a destino, o sea, &FB bytes.

DIRECCION	OPCODE	MNEMONICO
3000	18 03	JR here
3002	00	NOP
3003	00	NOP
3004	00	NOP
3005	C9	here RET

Programa 4

DIRECCION	OPCODE	MNEMONICO
3000	00	there NOP
3001	00	NOP
3002	00	NOP
3003	18 FB	JR there
3005	C9	RET

Programa 5

Este es todo el misterio, o casi, de los saltos relativos.

Al igual que con los desplazamientos absolutos, existen los relativos según se cumpla o no determinada condición, reflejada en los flags.

Saltos relativos condicionales

Por tanto, los saltos relativos condicionales, como JR Z, o JR NC, también funcionan, calculándose los desplazamientos a la dirección destino de la misma forma; la tabla 1 muestra los códigos de operación.

A lo largo de estas series, en todos los programas donde se habían colocado saltos absolutos (JP), pueden ponerse en su lugar los relativos, encima ocupando menos memoria *(un byte menos)*.

Cabría preguntarse entonces para qué están las instrucciones JP. Bien, debido a la necesidad de usar números negativos, los saltos relativos tienen un rango o **«alcance»** máximo: 127 bytes hacia adelante y —128 hacia atrás. Las razones de esto ahora resultarán evidentes. Los JPs, sin embargo, no están sujetos a estas restricciones: puede direccionar desde 0 a 65535, al precio de gastar más memoria.

DIRECCION	OPCODE	MNEMONICO
3000	3E 20	LD A, &20
3002	CD 5A BB	here CALL &BB5A
3005	C6 01	ADD A,1
3007	30 F9	JR NC, here
3009	C9	RET

Programa 6

El que piense que todos estos cálculos que requieren los relativos de bytes hacia adelante y hacia atrás, son un rollo, cuenta con la más vigorosa adhesión por parte de todos los programadores; para eso están los programas ensambladores, que además permiten el uso de etiquetas.

Sin embargo, como práctica y comprensión íntima de estas intrucciones, es IMPRESCINDIBLE hacerlo a mano un tiempo *(muy corto)*.

¡Ah, sí! La semana pasada dejamos un **«ejercicio»** para resolver por nuestros lectores: el triángulo de asteriscos.

La respuesta, una de las posibles, es el programa 7. Naturalmente, no hemos podido resistir la tentación de acudir a los saltos relativos.

DIRECCION	OPCODE	MNEMONICO
3000	0E 01	LD C,1
3002	41	back LD B,C
3003	3E 2A	over LD A, &2A
3005	CD 5A BB	CALL &BB5A
3008	05	DEC B
3009	C2 03 30	JP NZ, over
300C	3E 0A	LD A, &0A
300E	CD 5A BB	CALL &BB5A
3011	3E 0D	LD A, &0D
3013	CD 5A BB	CALL &BB5A
3016	0C	INC C
3017	79	LD A, C
3018	FE 09	CP &09
301A	20 E6	JR NZ, back
301C	C9	RET

Programa 7

¡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 sólo 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

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 incluído 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)*

¡¡Increíble!!

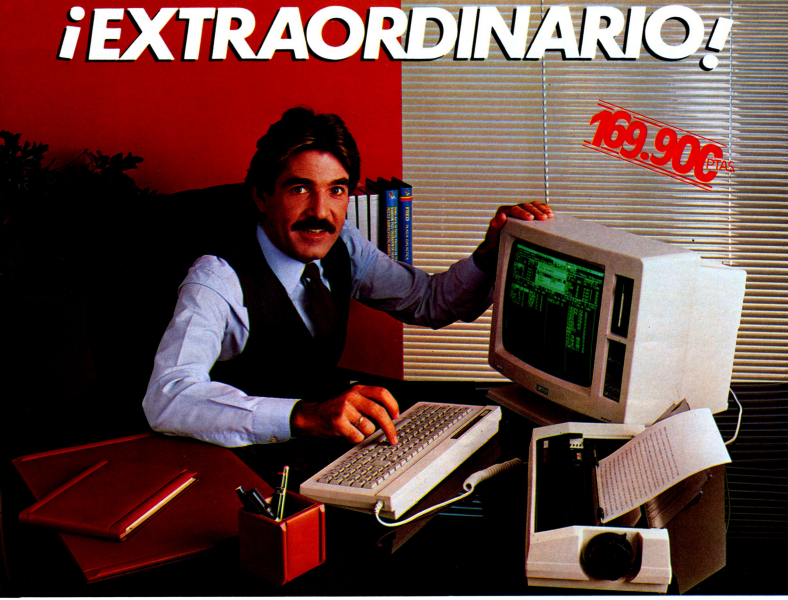
ESPAÑA

AMSTRAD

GARANTIA INDESCOMP

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

¡EXTRAORDINARIO!



AMSTRAD le propone la solución más completa al ordenador personal y al proceso de textos. El sistema PCW 8256 le ofrece, por el precio de una máquina de escribir, un ordenador personal de 256K con ***teclado profesional*** *y caracteres en castellano (ñ, acentos, etc.). Una pantalla de* ***fósforo verde de alta resolución*** *(90 col. x 32 lin.), con una* ***unidad de disco*** *de 3" (180K por cara) integrada en el monitor (con opción de integrar un segundo disco de 1Mb) y una* ***impresora*** *de textos de alta calidad con diversos tipos de letra.*

Imprime tanto hoja a hoja como papel continuo y tiene una alineación automática de papel. Junto con el sistema PCW 8256, se suministra el procesador de textos LOCOSCRIPT (totalmente en castellano) el cual tiene reservadas, teclas específicas de control).

UN EQUIPO EXTRAORDINARIO

Pero esto no es todo. El PCW 8256, al trabajar con el Sistema Operativo CP/M Plus, con un extra de gráficos GSX, tiene la posibilidad de acceder a los mejores programas profesionales del mercado: SuperCalc 2, Multiplan, dBase II, etc., así como a todo tipo de lenguajes: PASCAL, COBOL, FORTRAM, FORTH, etc.

Asimismo, gracias al programa Dr. LOGO, que se suministra con el Sistema PCW 8256, se tiene un inmenso campo de posibilidades en el mundo de la enseñanza.

LO MAS NUEVO EN SU AMSTRAD PCW 8256

Disponer del sistema AMSTRAD 8256, es tener en la mano la última tecnología punta a precio de excepción.

Trabajar con «disco virtual» o, simplemente, operar con «ficheros indexados» era, hasta hace poco, solo imaginable en equipos muy sofisticados y de alto precio.

El Sistema Informático PCW 8256 incluye en su suministro:

- *Teclado profesional.*
- *Unidad de Disco.*
- *Pantalla de alta resolución.*
- *Impresora.*
- *Programas en disco:*
  - *Mallard BASIC con sistema JETSAM para ficheros indexados.*
  - *Sistema Operativo CP/M Plus.*
  - *Procesador de textos LOCOSCRIPT.*
  - *Lenguaje Dr. LOGO.*
  - *Diversas Utilidades.*
- *Completa documentación y manuales en castellano.*
- ***GARANTIA AMSTRAD ESPAÑA UNICA VALIDA PARA ACCEDER AL SERVICIO TECNICO OFICIAL.***

¡¡Increible!!

ESPAÑA

AMSTRAD

GARANTIA INDESCOMP

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

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.

¿Que debo hacer para poder leer el contenido de los 32 K de ROM? Mediante el comando PEEK sólo tengo acceso a la RAM y desconozco la manera de conmutar los francos de memoria.

Marcos Miraflores
(Málaga)

Tienes que hacerlo mediante una rutina en lenguaje máquina que te listamos aquí, junto con un pequeño programa Basic que también incluimos.

El programa en ensamblador se realizó con el ensamblador de Hisoft, pero puede adaptarse sin problemas al ensamblador publicado por nosotras en el número 8 de ***Amstrad*** *Semanal.*

El programa Basic supone que existe el código máquina en la memoria a partir de la dirección 30000, por lo que debes ensamblarlo y guardarlo en cinta o disco, y luego hacer un programita cargador que introduzca el fichero de código máquina en memoria y luego el Basic.

En cuanto al programa en máquina, su funcionamiento es muy simple: las rutinas ei-ROM y di-ROM habilitan y deshabilitan, respectivamente, la ROM inferior del ***Amstrad,*** *permitiéndote que, al hacer una lectura, leas la ROM en lugar de la RAM.*

Para comodidad tuya, la rutina mueve los bytes leídos de la ROM de 10 en 10 a un buffer (etiquetas ***«nbytes»*** y buffer), *de modo que puedas procesarlos como te acomode.*

Si decides que necesitas un buffer mayor, tan sólo hará falta variar el valor de ***«nbytes»*** *en la línea 380 del listado en ensamblador.*

Hemos incluido también una pequeña subrutina (hexa, líneas 230--370 del listado en ensamblador) *que convierte un número binario a formato hexadecimal y lo imprime en pantalla.*

Además, hemos pensado que sería interesante que los bytes hexadecimales aparecieran en la impresora, si se tiene, por lo que la rutina de la línea 360 (i-char) imprime caracter a caracter en el papel. El método es un tanto tosco, como comprobarás, pero por razones de espacio ha habido que conformarse con ello. Modificaciones sencillas, pero largas, te permitirán mejorar el output de la impresora.

La rutina sólo copia las primeras 16 K de ROM, o sea, el firmware completo. Para inspeccionar el intérprete de Basic, la ROM superior, el programa sería el mismo, pero las rutinas di-ROM y ei-ROM deben corresponder a las direcciones firmware:

ei-ROM equ &B900
di-ROM equ &B903

Por último, el programa Basic te pregunta las direcciones de comienzo y final de la ROM que quieres ver, y luego pasa la dirección inicial a la rutina en máquina como parámetro (línea 60 del listado Basic).

```
   10          org  30000
   20          ent  30000
   30
   40          call ei_rom
   50          ld   l,(ix+0)
   60          ld   h,(ix+1)
   70          ld   de,buffer
   80          ld   bc,nbytes
   90          push bc.
  100          ldir
  110          call di_rom
  120          pop  bc
  130
  140          ld   hl,buffer-1
  150 p_bytes: inc  hl
  160          ld   a,(hl)
  170          call hexa
  180          dec  bc
  190          ld   a,b
  200          or   c
  210          ret  z
  220          ld   a," "
  230          call p_char
  240          jr   p_byte
  250
  260 hexa:    push af
  270          rrca
  280          rrca
  290          rrca
  300          rrca
  310          call nibble
  320          pop  af
  330 nibble:  and  #0f
  340          add  a,#90
  350          daa
  360          adc  a,#40
  370          daa
  380          call p_char
  390          call i_char
  400          ret
  410
  420 nbytes: equ  10
  430 p_char: equ  #bb5a
  440 ei_rom: equ  #b906
  450 di_rom: equ  #b909
  460 i_char: equ  #bd2b
  470 buffer: defs nbytes
  480
  490 ;para la rom alta
  500 ;sustituir di_rom
  510 ;y ei_rom por
  520 ;ei_rom:equ #b900
  530 ;di_rom:equ #b903
  540         end
```

```
10 MODE 2:romtope=&4000:dircarga=30
000
20 INPUT "Direccion de comienzo (me
nor que cero para acabar): ";dir
30 IF dir<0 THEN END
40 INPUT "Direccion final (menor de
 &4000) ";dirfin
50 IF ABS(dirfin)>romtope THEN 40 E
LSE GOSUB 60: GOTO 20
60 IF dir>dirfin THEN RETURN ELSE P
RINT "dir=";dir;TAB(20);CALL dircar
ga,dir:dir=dir+10:PRINT:PRINT:GOTO
60
```

AMSTRAD Semanal comunica a todos sus lectores la apertura de una nueva sección dedicada a recoger las mejores ideas que exploten al máximo las posibilidades del ordenador, materializadas en programas claros y cortos *(máximo 25 líneas).* Los mejores de entre todos ellos serán publicados con el nombre de su autor en la revista, recibiendo como premio, gratuitamente en su domicilio los cuatro primeros números de nuestra cinta mensual. Los programas enviados deberán incluir:

Amstrad Ideas

— Cinta de cassette con el programa o programas grabados.
— Explicación detallada del funcionamiento y propósito del programa, mecanografiado a 2 espacios o con letra clara.

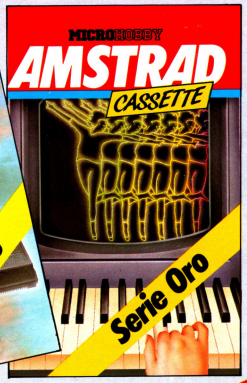
Es imprescindible indicar en el sobre claramente: **AMSTRAD IDEAS.**
La dirección es:
Editorial Hobby Press, S. A.
La Granja, s/n.
Polígono Industrial de Alcobendas.
Madrid.

MICROHOBBY
AMSTRAD Semanal
REVISTA INDEPENDIENTE PARA USUARIOS DE ORDENADORES AMSTRAD

LE OFRECE AHORA SUS PROGRAMAS YA GRABADOS PARA QUE VD. NO TENGA QUE TECLEARLOS. TOTALMENTE DESPROTEGIDOS CON EL OBJETO DE FACILITAR SU COPIA EN DISCO.

Todos los programadores y aficionados a la microinformática, sabemos lo tedioso y propenso a errores que resulta el teclear un listado de un programa. Para facilitar tu labor al máximo y evitar que malgastes largas horas sobre el teclado de tu ordenador tratando de descifrar incomprensibles mensajes de error.

AMSTRAD SEMANAL te ofrece cada mes los programas publicados en los cuatro números correspondientes, en una cinta de cassette desprotegida, que te permitirá copiar los programas en disco y tener acceso a los listados para su estudio y posterior edición de rutinas.



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

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

Por sólo 675 pts.
(incluidos gastos de envío)

Recíbelos en tu casa cómodamente enviándonos con la menor demora posible, el cupón que se encuentra en la última página de la revista.

COMPATIBLE CON LOS MODELOS CPC-464, CPC-664 y CPC-6128

COLABORADORES:

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

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

Se valorará el conocimiento del inglés.

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

Jorge Juan, 116. 28028 Madrid
Tels. (91) 233 07 35-274 53 80

MICROLID: Gregorio Fdez., 6
Tel. (983) 35 26 27. Valladolid

Hemos creado para ti el nuevo Club del Software MICRO-1. En él vas a encontrar los últimos títulos a unos precios increíbles. Para hacerte socio sólo es necesario que nos pidas uno de estos programas, teniendo como regalo de bienvenida un magnífico bolígrafo con reloj de cuarzo incorporado.

¡¡Primicia AMSTRAD: Ajedrez en tres dimensiones con sintetizador de voz y completamente en castellano: 2.395!!

	Ptas.		Ptas.		Ptas.
Match day	**1.975**	Dragontorc	**1.875**	Raid o moscow	**1.975**
Southern belle	**1.975**	Dummy run	**1.975**	Jump jet	**2.595**
Death pit	**1.975**	Combat lynx	**1.925**	Beach head	**2.395**
Decathlon	**1.975**	Exploding fist	**2.095**	Basketball	**1.950**
Gremlins	**1.975**	Rocky	**1.925**	Highway encout.	**1.975**
Popeye	**1.875**	Basketball	**1.925**	Hightsade	**1.925**

Joystick Quick Shot I
1.795

Joystick Quick Shot II
2.495

Joystick Quick Shot V
¡¡El mejor!!
2.995

Lápiz óptico DK'Tronics
5.850

¡¡Increíbles precios para tu AMSTRAD 464 y 6128!!

Cassette especial para 664 y 6128: 5.295

Tapa de metacrilato transparente para tu AMSTRAD
2.450

Impresoras
20% Dto. sobre P.V.P.

Cinta C-15 **Diskette 3**
85 **995**

LIBROS

	Ptas.		Ptas.
Curso Autodidáctico Basic (I)	**2.725**	Curso Autodidáctico Basic (II)	**2.750**
Programando con AMSTRAD	**2.250**	Hacia la inteligencia artific.	**1.375**
Juegos sensacionales AMSTRAD	**1.790**	Código máquina para princip.	**1.975**
40 juegos educativos AMSTRAD	**1.790**	Programando Basic con AMSTRAD	**1.975**
Guía de referencia de Basic	**1.900**	Música y sonido con AMSTRAD	**1.125**

El pedido te lo enviamos URGENTEMENTE contra-reembolso SIN NINGUN GASTO DE ENVIO, LLAMANDO a los teléfonos: (91) 233 07 35-274 53 80 o escribiendo a MICRO-1. Jorge Juan, 116. 28028 Madrid.

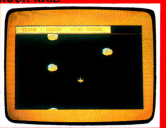
MICRO BYTE PRESENTA... AMSTRAD

NUEVOS PROGRAMAS EN CASSETTE Y DISCO

ARGO NAVIS

El comandante de nave AMSTRAD 1 se encuentra atrapado en las profundidades de una central nuclear y debe salir con vida Excelentes gráficos y sonido. **P.V.P.: CASSETTE 2.200 pts. DISCO 2.900 pts.**

ROCK RAID



Debes pilotar con acierto la nave que a lo largo de su viaje galáctico sufrirá encuentros con meteoritos, residuos planetarios, etc. Gran movilidad y excelentes efectos. **P.V.P.: CASSETTE 1.900 pts. DISCO 2.600 pts.**

WIZARD'S LAIR

Te encuentras atrapado en las profundidades de una caverna, llena de obstáculos, adversidades, etc. ¿Serás capaz de salir con vida? **P.V.P.: CASSETTE 1.900 pts. DISCO 2.600 pts.**

MACADAM FLIPPER

Atractivo programa que nos traslada al manejo de la máquina-flipper del mejor casino de Las Vegas. Posibilidad de creación del tablero, puntuaciones, etc. **P.V.P.: CASSETTE 2.200 pts. DISCO 2.900 pts.**

JUMP JET

Te encuentras a los mandos de la nave "Aircraft". En una perfecta maniobra debes despegar del portaviones. Excelente versión simulador vuelo-combate). **P.V.P.: CASSETTE 2.200 pts. DISCO 2.900 pts.**

MUSIC MAESTRO

El más completo programa de música creado para el AMSTRAD. Permite crear sonidos, melodías y convertir tu ordenador en la mejor "caja de música". **P.V.P.: CASSETTE 2.200 pts. DISCO 2.900 pts.**

PAZAZZ

Programa que permite de una manera sencilla la creación de pantallas con gráficos, dotarles de movimiento, acompañados de música. **P.V.P.: DISCO 2.900 pts.**

SYCLONE 2

Programa de utilidad que permite realizar copias de seguridad (back-ups) a distintas velocidades (baudios). **P.V.P.: CASSETTE 1.800 pts. DISCO 2.500 pts.**

ZEDIS II

Editor-desensamblador del Z-80, para el programador más avanzado. **P.V.P.: CASSETTE 1.900 pts. DISCO 2.600 pts.**

SYSTEM X

Ampliación del lenguaje Basic. Conjunto de 30 nuevas instrucciones (fill, circle, protect) para ayudar en la programación. **P.V.P.: CASSETTE 2.200 pts. DISCO 2.900 pts.**

ODDJOB

La mejor utilidad para el mejor conocimiento del disco.
(Copias de disco, Disk map, Disk track, sector, etc.). **P.V.P.: DISCO 2.600 pts.**

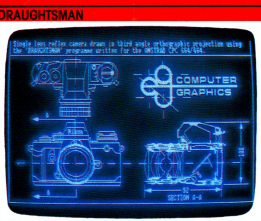
TRANSMAT

Pasar los mejores programas de cinta a disco ya no es problema. Con Transmat este proceso será fácil y sencillo. **P.V.P.: DISCO 2.600 pts.**

OTROS PROGRAMAS EN STOCK

MIÑI OFFICE	P.V.P. CASS. 3.200 pts. P.V.P. DIS. 3.900 pts.
WORLD CUP FOOTBALL	P.V.P. CASS. 1.800 pts.
BATLE FOR MIDWAY	P.V.P. CASS. 1.800 pts.
FIGHTER PILOT	P.V.P. CASS. 2.200 pts.
SURVIVOR	P.V.P. CASS. 1.800 pts.
MOON BUGGY	P.V.P. CASS. 1.800 pts.
TECHNICIAN TED	P.V.P. CASS. 1.800 pts.
FRUITY FRANK	P.V.P. CASS. 1.800 pts.
DATABASE	P.V.P. CASS. 2.100 pts.
LOGO TURTLE GRAPHICS	P.V.P. CASS. 2.400 pts.
TASCOPY Y TASPRINT	P.V.P. CASS. 2.600 pts.
FONT EDITOR	P.V.P. CASS. 1.900 pts.

DRAUGHTSMAN



Sofisticado programa de dibujo que permite tratar la pantalla del AMSTRAD como un sencillo tablero de dibujo. sus resultados son espectaculares **P.V.P.: CASSETTE 4.500 pts. DISCO 5.200 pts.**

ENVIENOS A **MICROBYTE**
P.º Castellana, 179, 1.º - 28046 Madrid

Nombre
Apellidos
Dirección
Población
D.P. Teléfono

ENVIOS GRATIS

JUEGO	C	D	Precio	TOTAL

PRECIO TOTAL PESETAS

Incluyo talón nominativo ☐
Contra-Reembolso ☐

Pedidos por teléfono **91 - 442 54 33 / 44**