AÑO I N.º 2

MICROHOBBY AMSTRAD Semanal

REVISTA INDEPENDIENTE PARA USUARIOS DE ORDENADORES AMSTRAD

AÑO I N.º 2

150 Ptas.

Canarias 160 ptas.

AMSWORD, UNA MAQUINA DE ESCRIBIR INTELIGENTE

ADMIRA LOS SPRITES EN EL AMSTRAD CON EGG BLITZ

COMANDOS GRAFICOS EN AMSTRAD ANALISIS

A TRAVES DE LA JUNGLA DEL MICRO

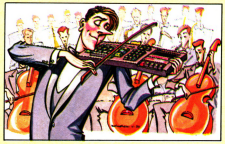
SOFTWARE

GREMLINS, LA FAMOSA PELICULA DE SPIELBERG EN TU ORDENADOR

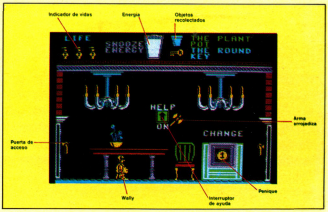
MICROMANIA. Sólo para adictos

«COMO SE PROGRAMA UN JUEGO»

... para aprovechar a tope tus posibilidades como programador.



«AMSTRAD MUSICAL»... *amplios artículos a tu medida.*



«PATAS ARRIBA»... *la sección que destripa los mejores juegos, POKE A POKE, para hacerte invencible.*

Una revista con marcha para los que necesitan saber TODO sobre ordenadores.

... Y además, la posibilidad de ganar una POLAROID si encuentras al travieso BYTE enmascarado.

HOBBY PRESS, S.A. *Editamos para gente inquieta.*

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
Secretaría Redacción
Carmen Santamaría
Fotografía
Carlos Candel
Javier Martínez
Portada
Manuel Barco
Ilustradores
J. Igual, J. Pons, F. L. Frontán, J. Septién, Pejo, J. J. Mora

Edita
HOBBY PRESS S.A.

Presidente
María Andrino
Consejero Delegado
José I. Gómez-Centurión

Jefe de Publicidad
Concha Gutiérrez

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

Dto. Circulación
Carlos Peropadre

Distribución
Coedis, S. A. Valencia, 245. Barcelona

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

MICROHOBBY

AMSTRAD semanal

REVISTA INDEPENDIENTE PARA USUARIOS DE ORDENADORES AMSTRAD — AÑO I N.º 2

Año I • Número 2 • 10 al 16 de Septiembre de 1985
150 ptas. (sobretasa Canarias, 10 ptas.)

Primera plana

EXPLODING FIST

La última creación de MELBOURNE HOUSE, entra de lleno en los juegos de acción.

Ser un maestro del kárate, requiere un largo aprendizaje. Cada combate se realiza contra un karateca más experto, desde iniciado a décimo dan, cada adversario pondrá a prueba nuestra concentración y rapidez de reflejos.

Emplear nuestros mejores golpes en los momentos adecuados, es imprescindible para ganar los combates.

GRAFICOS DE EMPRESA

Un paquete con una serie de rutinas y utilidades que permiten manejar gran cantidad de plotters e impresoras estará pronto disponible para AMSTRAD; se llama GENPLOT.

Permite representar visualmente en pantalla series de datos en forma gráfica, los conocidos gráficos de barras, histogramas, etc., superponiendo unos gráficos a otros si fuera necesario. Se comercializará en versiones de cassette y disco.

LLEGA EL NUEVO CPC128

No contentos con revolucionar por dos veces consecutivas el mercado de ordenadores caseros, lanzando el CPC464 y su hermano mayor, el compatible CPC664, nace ahora el **CPC6128**.

Este nuevo ordenador posee 128 KBYTES DE RAM, disco de 3 pulgadas integrado, un Z80 como microprocesador, el mismo que el de sus antecesores, y funciona bajo el sistema operativo CP/M PLUS *(también conocido como CP/M 3.0)*, manteniendo la compatibilidad con los dos modelos anteriores.

Estos significa poder acceder a un tipo de programas mucho más potentes, sin perder por ello el software que corre bajo CP/M 2.2 *(sistema operativo del CPC664)*, el cual puede ejecutarse también sin ningún problema. Por tanto, el número de programas de gestión y de utilidades para la programación se incrementa significativamente, con el aliciente de que ahora disponemos del doble de memoria.

No se ha dejado de lado el asunto de los juegos: ya existen las versiones de los más famosos en disco o en cassette.

Según nos tiene acostumbrados, **AMSTRAD** ha mejorado el Basic de la nueva máquina, añadiendo comandos nuevos o haciendo aún más potentes algunos de los existentes.

Pronto tendremos a nuestro alcance la última bomba de **AMSTRAD**.

SUBEN LOS PRECIOS DE LOS ORDENADORES

Probablemente todos recordarán el viejo refrán que asegura que «en el amor y en la guerra... todo vale». Pero aquí no se trata de amor, sino de guerra. De una guerra despiadada, encabezada por aquéllos a los que no les baste la libre competencia para colocar sus productos en nuestro mercado, sino que se ven obligados a recurrir a argumentos más imperativos, recuerdo de tiempos en que los mismos tomaban las mismas decisiones, una y otra vez.

Resulta que suben los ordenadores, y además, todos los terminales de pantalla, teclados e impresoras, tanto de fabricación nacional como de importación, deben estar homologados, es decir, cumplir una serie de normas stándart de calidad y seguridad.

Nuestra incrédula sorpresa ante la primera medida, y nuestro aplauso para la segunda.

En efecto, el Consejo de Ministros, es su reunión del día 17 de julio de 1985, a propuesta del Ministerio de Economía y Hacienda, aprobó una elevación en el vigente arancel de aduanas, que afecta a las máquinas automáticas para el tratamiento de la información y sus unidades, lectores magnéticos y ópticos, ...etc., es decir, a los ordenadores personales, entre los cuales, por supuesto, se encuentra el AMSTRAD.

Esta sorprendente medida, publicada en una fecha de lo más oportuna, en pleno julio, representa un duro golpe para el mercado español de microinformática. O al menos, para casi todo el mercado, ya que dicha subida NO AFECTA A LOS ORDENADORES FABRICADOS EN TERRITORIO NACIONAL.

El objeto de esta medida está bastante claro, y para comprender su alcance no es necesario hacer uso de grandes dosis de mala intención.

Existe un ordenador cuyo nivel de venta de unidades sólo se puede describir si al cero se le añaden varios decimales. Sus fabricantes han utilizado grandes dosis de influencia para conseguir un decreto que perjudicara a su competencia, pero que sólo con grandes dosis, también, de un desenfrenado optimismo, se puede suponer que les beneficiara a ellos.

En definitiva: el patético síndrome del perro del hortelano, que ni vende, ni deja vender al amo.

Tampoco hay que olvidar la conocida voracidad fiscal de nuestra Hacienda Pública, decidida a recaudar el río revuelto. En marzo, cuando los acuerdos arancelarios de la CEE entren en vigor, estos nuevos aranceles no podrán ser aplicados, pero la campaña navideña proporcionará muchos millones al fisco... o disminuirá radicalmente la venta de ordenadores domésticos. En fin, en cualquier caso, peor para el usuario, porque es difícil hablar de consumidores dentro de la informática, donde muchos de los ordenadores se han convertido en indispensables herramientas de trabajo y diversión.

La natural reacción de los sectores implicados, ante el castigo impuesto desde las alturas, ha conseguido al menos mitigar en parte el rigor de la ley:

La subida de los precios afectará **«solamente»** a aquellos ordenadores cuya memoria RAM no supere las 64K, es decir, a la inmensa mayoría de ellos.

También se ha conseguido que el aumento de precio se refiera únicamente a la unidad central *(el teclado)*, porque, al menos para el AMSTRAD, existía la incríble pretensión de gravar también la pantalla *(!)*. No deja de ser un orgullo que AMSTRAD reciba un trato de favor a la hora de limitar sus ventas sea como sea, por parte de según quien.

Lo que no se ha conseguido, es arremeter contra la disposición de que cualquier máquina que se venda en nuestro país debe llevar la letra **«ñ»**, lo cual, en un sentido, está muy bien.

En efecto, es lógico que los teclados que usamos nos permitan escribir con todas las letras de nuestro idioma, pero es aún más lógico otorgar un plazo razonable para efectuar los cambios necesarios en la ROM de los ordenadores de forma que los capaciten para incluir la **«ñ»**.

Nos preguntamos, seriamente preocupados, si los instigadores de esta disposición piensan que es sencillo modificar la ROM de un ordenador en CUATRO MESES. Si lo piensan, bueno, pues al menos sería discutible. Si no lo piensan, **¿a qué viene conceder un plazo tan exiguo?**.

La respuesta a esta pregunta es lo sufientemente triste para dejar que cada uno extraiga sus propias conclusiones.

Que quede muy claro nuestro apoyo incondicional a todas las medidas que tiendan, como queremos creer que es el caso de la **«ñ»**, a mejorar la calidad y seguridad de los productos que los españoles usen en sus hogares o fuera de ellos.

En este sentido, sin duda, van encaminadas el resto de las especificaciones que todos los ordenadores deben cumplir para poder ser oficialmente homologados.

Pero, por favor, den un plazo razonable para que las máquinas existentes puedan adaptarse a ellas y, sobre todo, comuniquen a los distribuidores DONDE tienen que ir a homologar sus equipos cuanto antes, porque, por incríble que parezca, esta elemental previsión no ha sido cumplida.

Desde este mismo momento queremos alzarnos contra aquéllos que nos acusen de arrimar el ascua a la sardina de AMSTRAD con la trillada excusa del «bien del usuario», y ello por dos razones: la primera, que honradamente creemos que la subida de precios y la homologación a marchas forzadas tienen escaso sentido, si es que no esconden otro tipo de «razones» que en el mundo de los ordenadores ya son de dominio público, y la segunda, dedicada a los especialmente escépticos, es que el lanzamiento inminente de máquinas con 128 Kbytes de RAM las colocará fuera del alcance de la subida de precios.

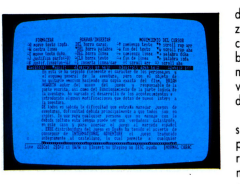
PROCESO DE TEXTOS

Cuando se crearon los ordenadores, y aun tiempo después, más de un usuario potencial se preguntó para qué servían esos aparatos tan raros, sobre todo, para que le servían a él. Duda justificada, si tenemos en cuenta que el aspecto que ofrecían las computadoras al exterior era el de unos artefactos enormes, servidos por un escogido sacerdocio de técnicos vestidos de blanco y que costaban un montón de dinero.

Como se suele decir, desde entonces ha llovido mucho. Los ordenadores actuales son pequeños, potentes y lo suficientemente baratos para poder comenzar a plantearse el adquirir uno para poder usarlo con éxito en... ¿en qué?

Adiós a las máquinas de escribir

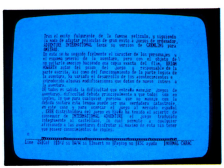
Bien, una de las aplicaciones que ha convertido al ordenador en presencia obligada en oficinas y muchos hogares, sin duda ha sido el PROCESO DE TEXTOS, esto es, emplearlo como si de una máquina de escribir se tratase pero con una enorme ventaja: EL PROCESÓ DE CREACION DE UN DOCUMENTO QUEDA COMPLETAMENTE SEPARADO DEL ACTO DE IMPRIMIRLO. A poco que se considere, esta posibilidad está muy lejos de ser un asunto trivial; en efecto, poder manipular un texto a nuestro antojo, añadiéndolo o suprimiéndolo, cambiándolo de lugar, etc., potencia enormemente el aspecto creativo y funcional de la escritura, y convierte al ordenador en algo más allá de un capricho, elevándolo a la categoría de herramienta de trabajo extremadamente útil y a menudo insustituible.



No es de extrañar que todas estas máquinas rivalicen en ofrecer al usuario programas de proceso de texto potentes y fáciles de manejar, programas **«user-friendly»,** según la terminología inglesa.

En esta línea se encuentra uno de los programas de proceso de textos que AMSTRAD ofrece, en versiones de cinta y disco. Se llaman AMSWORD y es el producto de una colaboración entre TASMAN SOFTWARE y AMSOFT.

AMSWORD consta de dos partes o ficheros llamados DISC.BAS y AMSWORD.BIN en la versión de disco. El primero es un largo y eficiente programa escrito en Basic que se encarga de gestionar cosas como menús, las labores de salvar y cargar textos del disco, etc.



Basic y código máquina, codo con codo

El segundo son nada menos que 14 bytes de código máquina que se ocupan de realizar labores complejas, imposibles de llevar a cabo en Basic o que serían muy lentas. Ambos ficheros armonizan perfectamente, y muestran que no sólo de lenguaje máquina viven los programas. Un buen Basic, como el de AMSTRAD, también puede hacer mucho.

AMSWORD nos deja libres unos 13 bytes, suficiente para textos de tamaño mediano; el programa no contempla el uso de la memoria RAM del ordenardor como buffer, manteniendo el grueso del documento en el disco y accediendo a él cuando sea necesario; es decir, cuando la memoria del AMSTRAD se llena, no se puede escribir una letra más. Hay que salvar en disco el texto y comenzar a escribir el resto como otro fichero diferente. Antes de rasgarnos las vestiduras ante esta particular forma de proceder del programa, y clamar enfurecidos contra el «desperdicio» de la capacidad del disco, tal vez convenga recordar que este procedimiento presenta ventajas e inconvenientes. La obvia *(y única)* desventaja estriba en la imposibilidad de escribir documentos largos *(libros, biblias)* de UNA SOLA VEZ; nos veríamos obligados a fraccionarlos en secciones, con las consiguientes molestias si, por ejemplo, el párrafo 5 de la página 32 decidimos colocarlo en la página 1. Sin embargo, si planeamos con cuidado nuestros «trozos», la continuidad del documento no debe romperse; nada nos impide sacarlos por impresora uno detrás de otro, numerando adecuadamente las páginas.

operaciones que realicemos con él son instantáneas.

Segunda y principal: AMSWORD, a pesar de poder ejecutar la mayoría de las funciones de los procesadores de texto consagrados, aunque en menor escala, es un programa corto, sorprendentemente corto, diría yo. Otros de más categoría incluyen multitud de ficheros, a los cuales el programa principal tiene que acceder en un momento dado para cumplir alguna misión muy concreta. Como todos estos programas auxiliares ocupan espacio, al final resulta que del disco con el programa completo nos quedan muy pocos bytes libres, con lo cual se abren ante nosotros tres caminos:

1. *Poner los textos en un disco de datos independiente. Esto implica, en el mejor de los casos, andar sacando y metiendo discos sin parar, atendiendo a los mensajes del programa pidiendo el disco del sistema. En el peor de los casos, habría que adivinar cuando el programa principal necesita determinado fichero auxiliar del disco, sacar el disco de datos, meter el del programa y otra vez el de datos...*

2. *Emplear dos unidades de disco, con el consiguiente aumento del coste.*

3. *Mezclar programa y datos en el mismo disco, con la subsiguiente limitación de tamaño en el documento.*

Mantener el equilibrio

AMSWORD es un compromiso muy logrado entre los programas que necesitan dos unidades de disco y los que requieren una sola. Obtenemos alto rendimiento en relación con el coste, pero no podemos pedir a este programa las prestaciones de otros cuyo precio supera con mucho las 100.000 ptas. Creemos que las necesidades de un amplio grupo de usuarios quedan cubiertas con AMSWORD, hasta que se decidan a aumentar su capacidad de proceso con una segunda unidad de disco que les permita tratar grandes volúmenes de texto de una sola vez, si es que lo necesitan.

Una vez aclarado *(espero)* el lugar que este programa pretende ocupar dentro del ámbito de proceso de texto, muy bien podríamos pasar a inspeccionar, con un poco de detalle, los comandos y opciones de que dispone, junto con la filosofía del programa, el nudo alrededor del cual el programador configuró su obra.

Banco de pruebas

En honor del usuario

Tres palabras retratan al AMSWORD: FACIL DE MANEJAR. En efecto, pocas veces hemos tenido el placer de utilizar un programa sin tener que mirar los temidos manuales (sólo 45 páginas, gracias a Dios) ni una sola vez. Está claro que el diseñador del programa cargó las tintas en este asunto, y con pleno éxito. Desde el mismo momento que el programa arranca, el atribulado usuario novel se ve asistido por una pantalla de ayuda a la que se ha dedicado un tercio del display, y en la cual puede verse parte de los comandos disponibles a través de una ventana. Desplazándola, accedemos al resto de comandos y opciones disponibles, que podemos dividir por bloques de la siguiente manera.

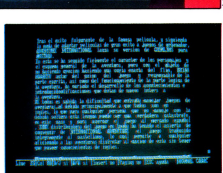
- COMANDOS DE FORMATEO
- BORRAR/INSERTAR TEXTO
- MOVER CURSOR POR LA PANTALLA
- AJUSTAR MARGENES
- AJUSTE DE TOPES DE TABULACION
- COMANDOS DE BLOQUE
- CONMUTADORES
- MANEJO DE LA IMPRESORA
- CARACTERES ESPECIALES
- CONTROLES DE IMPRESORA

Además de estas opciones, visibles en pantalla en todo momento, pulsando CTRL + ENTER accedemos al menú principal del programa, que podría etiquetarse como GESTION DE FICHEROS.

Algunas de las opciones de este menú no requieren mayor comentario, si cometido resulta obvio (grabar texto, cargar texto, mezclar texto, retorno al texto), mientras que otras realizan funciones más complejas y si requieren una descripción en profundidad; son las siguientes:

a) Retorno al Basic
b) Definir caracteres
c) Grabar programa
d) Imprimir texto

M. Barco



Todo en RAM. Dicho y hecho

Las ventajas: no existen tiempos de espera. Al estar todo el texto en memoria, todas las

CONMUTADORES

COMANDO	FUNCION
CTRL+F	Justificar a la derecha sí/no
CTRL+G	No cortar palabras al cambiar de línea sí/no
CTRL+H	Modo inserción *(al pulsar ENTER todo el texto se desplaza una línea hacia abajo a partir de la posición del cursor)* sí/no
CTRL+P	Mostrar una línea punteada indicando cambio de página sí/no
CTRL+1	Mostrar en pantalla un tercio de la pantalla de ayuda a la vez que el texto
CTRL+2	Desactiva la opción anterior, permitiendo ver más texto en pantalla de una vez.

Personalizar AMSWORD es muy fácil.

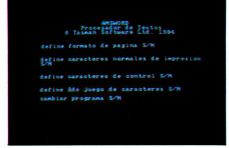
RETORNO AL BASIC: Esta opción permite volver al Basic sin que se pierda el texto del documento que tengamos almacenado en memoria. La utilidad consiste en que así podemos inspeccionar y modificar el programa si lo creemos necesario, con el doble objeto de comprender su funcionamiento (está escrito en un Basic de alto nivel) y poder programar las teclas de función con la frase que deseemos; normalmente serán aquellas que se emplean muy a menudo en nuestro documento, con lo cual aparecerán en pantalla a la pulsación de una única tecla *(más CTRL)*. Una vez realizadas las modificaciones necesarias en el Basic, simplemente tecleando RUN volvemos a nuestro texto en el punto donde lo dejamos.

DEFINIR CARACTERES: Si la opción anterior permitía personalizar el programa en su parte Basic, con esta alcanzamos un control mucho más profundo. Podemos definir el formato de la página de texto con el doble efecto de alterar su apariencia en pantalla y, más tarde, en la impresora, cambiando el número de líneas por página, escogiendo el número de líneas y texto que vamos a asignar a cabeceras y pies de página, etc.

También se nos permite alterar los caracteres de impresión, los caracteres de control de la impresora *(conocidos como* ***«secuencias de escape»*** *porque comienzan por chr$(27), el caracter ESCAPE)* y el segundo juego de caracteres de que dispone el programa *(letras griegas, letras acentuadas, signos matemáticos, ñ, etc.)*.

En una aplicación como el AMSWORD, concebida alrededor del concepto **«comodidad para el usuario»**, no podía faltar la opción de alterar a nuestro gusto el aspecto de la pantalla (colores de tinta, fondo y borde, forma y tipo del cursor, márgenes, etc). Los usuarios del monitor monocromo se verán agradablemente sorprendidos por una completa y suave gama de verdes.

En fin, podemos remodelar el programa hasta tal punto que ni su propio creador lo reconocería.



GRABAR PROGRAMA: Todo lo dicho anteriormente respecto al «rediseño» del AMSWORD está muy bien, pero no sería de gran utilidad si hubiera que teclearlo cada vez que arrancamos el programa. Por ello, esta opción cumple el doble cometido de proporcionar tantas copias de seguridad como deseemos y, de conservar en el disco la versión personalizada del AMSWORD de uso habitual. Además, el CP/M se encarga de que no haya confusión con anteriores versiones si estuvieran en el mismo disco, ya que las añade la extensión **«BAK»** para diferenciarlas.

COMANDOS DE BORRADO/INSERCION

COMANDO	FUNCION
DEL	Borra un único carácter
SHIFT+DEL	Borra una palabra
CTRL+DEL	Borra una línea de texto
CTRL+CLR	Borra todo el texto, pidiendo confirmación antes de proceder
CTRL+I	Inserta un carácter o una línea

MOVIMIENTO DEL CURSOR

COMANDO	FUNCIÓN
CTRL+ CURSOR ARRIBA	Mueve el comienzo del texto
CTRL+ CURSOR ABAJO	Final de texto
CTRL+ CURSOR DERECHA	Comienzo de línea
CTRL+ CURSOR IZQUIERDA	Fin de línea
CTRL+<	Scroll arriba
CTRL+>	Scroll abajo
SHIFT+ CURSOR ARRIBA	Scroll rápido hacia arriba
SHIFT+ CURSOR ABAJO	Scroll rápido abajo
SHIFT+ CURSOR DERECHO	Avance hacia la derecha de palabra en palabra
SHIFT+ CURSOR IZQUIERDO	Hacia la izquierda de palabra en palabra

AJUSTE DE MARGENES Y TABULACIONES

COMANDO	FUNCION
CTRL+A	Margen izquierdo en pantalla
CTRL+D	Margen derecho
CTRL+S	Quita márgenes y restablece los que el programa tuviera por defecto.
TAB	Avanza el cursor hasta el siguiente tope de tabulación
SHIFT+TAB	Pone tabulador en la columna marcada por el cursor
CTRL+TAB	Quita tabulador
CTRL+X	Repone los tabuladores por defecto
CTRL+Z	Borra TODOS los tabuladores

IMPRIMIR TEXTO: Esta posibilidad, la más importante de todas, posee también un submenú que permite dar forma al texto inmediatamente antes de sacarlo por impresora.

COMANDOS DE FORMATEO

COMANDO	FUNCION
CTRL+Q	Desplaza una línea de texto hacia la izquierda si queda espacio para ello.
CTRL+E	Desplaza hacia la derecha con las mismas limitaciones que en el caso anterior.
CTRL+W	Centra una línea de texto.
CTRL+J	Justifica un párrafo por la derecha y por la izquierda de acuerdo con los márgenes vigentes en este momento.
CTRL+K	Como en el caso anterior, pero sólo con una línea de texto.

COMANDOS DE MANEJO DE BLOQUE

COMANDO	FUNCION
CTRL+B	Marca inicio de bloque de texto
CTRL+V	Marca el final del bloque de texto
CTRL+M	Mueve el bloque marcado y lo coloca a partir de donde se encuentre el cursor, eliminándolo de su lugar anterior
CTRL+N	Igual que el comando anterior, pero el bloque marcado no se elimina de su lugar primitivo, sólo realiza una copia
CTRL+C	Borra el bloque marcado

COMANDOS DE IMPRESORA Y OTROS

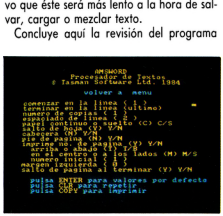
COMANDO	FUNCION
CTRL+6	Convierte la línea 1 del texto en cabecera, almacenándose en una zona interna del AMSWORD
CTRL+T	Extrae la cabecera de su lugar de almacenamiento para inspeccionarla y/o modificarla. CTRL+6 volverá a guardarla
CTRL+7	Lo mismo que CTRL+6, pero el texto se considera pie de página
CTRL+Y	Idem CTRL+T, para pie de página
CTRL+ENTER	Pasa al menú principal
CTRL+SPACE	Muestra los caracteres en vídeo inverso que hay que colocar en el texto para obtener determinados tipos de letras
CTRL+ \	Enseña el segundo juego de caracteres del AMSWORD
CTRL+*	Convierte en minúsculas todas las letras por las que pasamos el cursor
CTRL++	De minúsculas a mayúsculas

Cada ítem de este submenú tiene un valor por defecto; si decidimos conservarlo, simplemente pulsando ENTER pasamos al siguiente, de lo contrario, tecleamos el nuevo valor de entre los que se sugieren en pantalla y asunto concluido.

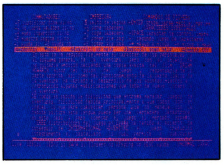
Podemos decidir las líneas de comienzo y final de la impresión, el número de copias, si queremos o no salto de página, el margen izquierdo, etc.

Todo lo dicho para el AMSWORD versión disco es válido para su homónimo de cinta, salvo que éste será más lento a la hora de salvar, cargar o mezclar texto.

Concluye aquí la revisión del programa AMSWORD, que por razones de espacio, no ha podido ser todo lo exhaustiva que quisiéramos. Creemos que los posibles usuarios han podido hacerse una idea de hasta qué punto puede resolver sus necesidades en materia de proceso de textos. Próximamente, analizaremos la **«otra cara de la moneda»** encarnada en un tradicional proceso de textos que funciona bajo el sistema operativo CP/M, de los llamados **«software de gestión profesional»**, y que aborda el problema del tratamiento de textos desde un punto de vista completamente diferente del AMSWORD. Su nombre: MICROSCRIPT.



Banco de pruebas



FICHA AMSWORD

AMSWORD II
CREADO POR:
TASMAN SOFTWARE & AMSOFT
COMERCIALIZADO POR: INDESCOMP
SOPORTE: DISCO Y CINTA DE CASSETTE
P.V.P.:
VERSION CASSETTE: **2.300 Ptas.**
VERSION DISCO: **6.500 Ptas.**

GREMLINS

Tras el éxito fulgurante de la famosa película, y siguiendo la moda de adaptar largometrajes de gran éxito a juegos de ordenador ADVENTURE INTERNATIONAL lanza su versión de GREMLINS para AMSTRAD.

En el juego para ordenador, se ha seguido fielmente el carácter de los personajes y el esquema general de la película, pero con el objeto de no quitarle emoción haciendo una copia exacta del film, BRIAN HOWARTH autor del guión del juego y responsable de la parte escrita, así como del funcionamiento de la aventura, ha variado el desarrollo de los acontecimientos, e introducido algunas modificaciones que dotan de nuevo interés a la aventura. De todos es sabida la dificultad que entraña manejar juegos de aventuras, dificultad debida principalmente a que todos son en inglés, lo que para cualquier persona que no maneje con la debida soltura esta lengua, puede ser una verdadera catástrofe. En este caso y para acercar el juego al mercado español, ERBE, distribuidora del juego en España ha tenido el acierto de conseguir de ADVENTURE INTERNATIONAL el juego traducido íntegramente al castellano, lo cual permite a cualquier aficionado a las aventuras disfrutar al máximo de ésta, sin tener que poseer conocimientos de inglés.

Aparte de la ventaja que supone el hecho de que el texto esté en castellano, el juego está realizado de forma que en la gran mayoría de las ocasiones, éste se ve acompañado por una ventana gráfica situada en la mitad superior de la pantalla, en la cual vemos el entorno que rodea a la posición en la que nos encontramos.

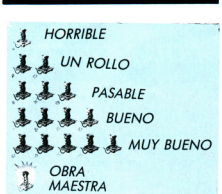
Los gráficos que acompañan al texto son impecables y llenos de colorido. Dejando atrás aquellas aventuras en las que el texto era el soporte fundamental de la aventura y las imágenes un raro complemento.

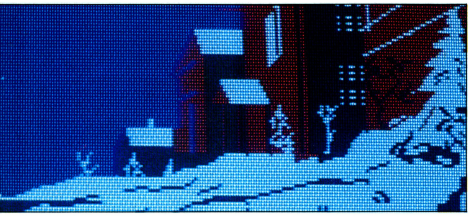
El programa maneja un completo diccionario, que contiene un número indeterminado de verbos los cuales nos permiten realizar un amplio número de actividades; ir, mirar, coger, saltar, leer, soldar, matar, y muchas otras que sólo seremos capaces de descubrir en el transcurso de la aventura.

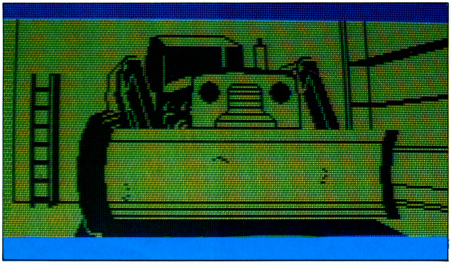
Completando el léxico manejado por el programa, con todas las instrucciones necesarias para el movimiento; arriba, abajo, norte, sur, este y oeste. Y los nombres de los objetos y lugares típicos de KINGSTON FALLS, pequeña ciudad donde tiene lugar la aventura; cine, estación, piscina, encendedor, máquina fotográfica, soplete, puerto, taladro eléctrico, etc.

En los momentos difíciles podemos solicitar ayuda por medio del socorrido HELP, ayuda que vendrá en función de la situación en que nos encontremos, los objetos que llevemos en ese momentos y el inefable factor aleatorio que siempre introduce el ordenador.

Volviendo a la estructura física del juego es importante destacar que los







Mr. Joystick

gráficos creados por ADVENTURE INTERNATIONAL, para ambientar el juego son francamente buenos, realizados con una técnica de dibujo realista, explotando al máximo el sentido de la perspectiva y con un tratamiento del color francamente cuidado.

El volcado de las distintas pantallas que constituyen los lugares donde nos encontramos, se realiza instantáneamente ayudando a dar dinamismo a la acción que se desarrolla en tiempo real. Este es un factor determinante en el transcurso de los acontecimientos, por lo cual decisiones meditadas detenidamente nos llevarán a un desenlace fatal.

La trama de la aventura es la siguiente:

Billy, el protagonista de la película, del cual nosotros tenemos el control absoluto, es un muchacho provinciano que un día encontro a GIZMO, el progenitor de los gremlins. Estos pequeños MOGWAI, explotando la compasión de BILLY, han conseguido que éste les dé de comer después de medianoche, con lo cual se transforman en GREMLINS malos, que bajo el mandato de su líder STRIPE, famoso por su maldad sin límites, quieren apoderarse de la sosegada y tranquila villa de KINGSTON FALLS eliminando a todos sus habitantes.

Nuestra tarea, con la ayuda de GIZMO, es la de impedir que nuestra amada villa natal caiga en manos de esos malvados seres que nos acosan por todos lados. Los tres consejos fundamentales que siempre debemos tener en cuenta son los siguientes:

** Impedir que se expongan a la luz solar, o a cualquier otro tipo de luz.*

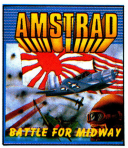
** No dejar que tomen contacto con el agua.*

**. Y sobre todo no darles nunca de comer después de medianoche.*

Siempre que se vulnere una de estas reglas, aumentará desmesuradamente su grado de maldad haciendo aún más difícil nuestra misión.

Sólo tú puedes salvar a KINGSTON FALLS de la plaga de los secuaces del malvado STRIPE.

BATTLE FOR MIDWAY



En mayo de 1942, llevado por el éxito obtenido en la batalla del pacífico, en los meses siguientes al ataque de PEARL HARBOR, el jefe de la primera flota japonesa, almirante YAMAMOTO propone un plan para aniquilar a la flota americana.

Su objetivo, destruir la base aérea de la isla de MIDWAY.

Al amanecer del 4 de junio, cuatro portaaviones de la armada japonesa: AKAGI, HIRYU, SORYU, KAGA, se adentran en la zona marítima de MIDWAY, apoyados por una flota de destructores, acorazados y cruceros de superficie.

Por su parte la flota americana solamente estaba compuesta por los portaaviones; HORNET, YORKTOWN y ENTERPRISE, y una exigua flota de superficie.

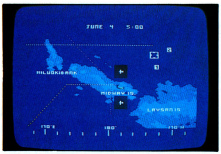
BATTLE FOR MIDWAY ha sido programado por ALAN STEEL, pionero en la creación de WAR-GAMES.

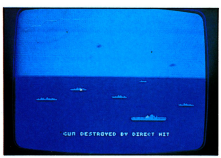
Avalado por la experiencia acumulada durante 25 años, en la realización de juegos bélicos, ALAN STEEL ha tenido en cuenta, en su versión de La Batalla de Midway, todos los factores que intervienen en una batalla aeronaval, consiguiendo una fiel reproducción de la histórica contienda.

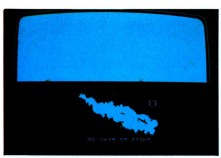
El programa está realizado con una técnica totalmente moderna, siendo controlado por joystick, con la ventaja de que no hay que escribir ni una sola línea de texto.

Todo el juego se realiza gráficamente, sobre el escenario de la batalla; un mapa de la zona marítima de Midway y sus alrededores.

Para obtener cualquier tipo de información, sobre nuestra flota de superficie o la enemiga, basta con situar el cursor sobre ella y pulsar la tecla adecuada.







Un reloj en pantalla tiene la misión de reflejar más exactamente el transcurso de la batalla, con él conocemos la hora de Midway en tiempo compensado.

El juego reproduce exactamente los hechos más notorios que tuvieron lugar en la batalla; por ejemplo, el ataque aéreo japonés al aeropuerto de MIDWAY, se realiza a las 6.00 según el reloj de pantalla.

A las 19.00 horas, cae la noche y

Mr. Joystick

debemos hacer aterrizar a todos los aviones que estén en vuelo. Durante la noche no se permiten ataques aéreos, solamente son posibles combates de superficie.

El juego incluye pantallas de combate de tipo arcade, en las que debemos derribar aviones enemigos, mantener combates de superficie con la flota enemiga y bombardear objetivos.

Detalle que añade a los programas de tipo WAR-GAME, una parte animada, lo que nos saca de la monotonía de las pantallas estáticas.

Para localizar la flota enemiga,

disponemos de dos aviones de reconocimiento, que se mueven en el espacio aéreo de MIDWAY.

Con ellos debemos detectar la posición de la flota enemiga y atacar sus portaaviones con nuestra fuerza aérea.

Todo el mecanismo de guerra que supone un combate aeronaval, está estudiado minuciosamente, de forma que debemos armar nuestros aviones, despegar de los portaaviones y regresar a los mismos, antes de que el combustible se agote.

Un juego con el que los amantes de los WAR-GAMES, disfrutarán a sus anchas, impidiendo que cambie el curso de la historia.

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;* ***POLE POSITION, 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 éstos 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

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, éstos deben superar las defensas costeras y llegar a la fortaleza que es el objetivo final.

POLE POSITION es la última creación de DATSOFT en la que podrás experimentar toda la emoción de un gran premio automovilístico, vuelta de clasificación y carrera cronometrada contra tus adversarios.

DALEY THOMPSO'N DECATHLON con este juego OCEAN enciende la llama olímpica y te reta a superar los récords de los campeones 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.

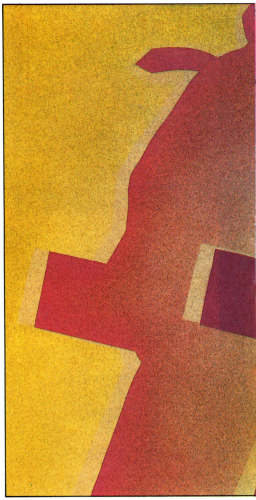
SEGUNDO DIA: 110 m vallas, lanzamiento de disco, salto con pértiga, lanzamiento de jabalina y los 1.500 m.

Primeros pasos

A TRAVES DE LA JUNGLA DEL MICRO

Programar es mucho más sencillo de lo que se piensa. Mediante una serie de conceptos clave y un poco de práctica, nuestras propias aplicaciones comenzarán a funcionar en nuestro AMSTRAD muy pronto.

F. L. Frontán



Cuando uno escribe un artículo, no sabe a quién se dirige ni quién lo va a leer. La persona que lo juzgue puede ser cualquiera: alguien que piensa, tal vez, comprar un AMSTRAD, o alguien que acaba de adquirirlo con la esperanza de que su inversión revierta en una mejor gestión de sus asuntos, o quizá sólo se pretenda pasar un rato agradable jugando **«por ordenador»;** quien sabe...

Pero de lo que no cabe duda es que por el hecho de leer este artículo, usted y yo compartimos un pequeño secreto y un oculto anhelo: queremos aprender a programar nuestro AMSTRAD.

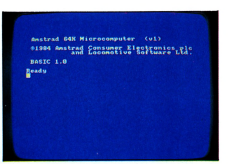
Claro que existen multitud de programas comerciales que resuelven prácticamente cualquier necesidad, pero eso a algunas personas les dirá muy poco. Preferirán el **«hágalo usted mismo»** no sólo por el menor coste en dinero, sino por el prurito de haber creado algo que funciona, y bien, a su propio estilo.

El gusanillo de la programación es una criatura maliciosa, y, cuando muerde, difícilmente suelta su presa. Uno se pasa horas frente a su máquina maravillado de poder combinar de una manera tan sencilla la utilidad con el arte, porque programar es, por encima de todo, crear.

No obstante, en el camino que todo programador recorre, ha habido un punto crucial en el que se ha encontrado con el ordenador encendido, frente al teclado, sin saber muy bien qué hacer a continuación; justo desde este confuso momento, para que deje de serlo, comenzamos nuestra singladura. Programar es más fácil de lo que parece.

Comenzar por el principio

Una vez encendido el ordenador, el primer signo de vida que advertimos será algo parecido a esto *(según se trate del 464 o del 664)*:



Como indica el mensaje de bienvenida, tenemos en nuestras manos un microcomputador de 64K que maneja colores. Las **«64K»** es una medida de la memoria disponible. En términos de uso común en ordenadores, tener más memoria quiere decir, a grosso modo, que podemos escribir más en la máquina antes de que **«se llene».**

Basic, como veremos más tarde, es el nombre del lenguaje en el cual damos órdenes al AMSTRAD. El resto del mensaje aclara quién comercializa el micro *(AMSTRAD)* y quién escribió el Basic *(Locomotive)*. En cuanto al críptico «READY», significa que el ordenador está preparado *(en inglés, READY)* para aceptar información y procesarla. A esta palabra la llamaremos el **«prompt»** o simplemente el **«READY».**

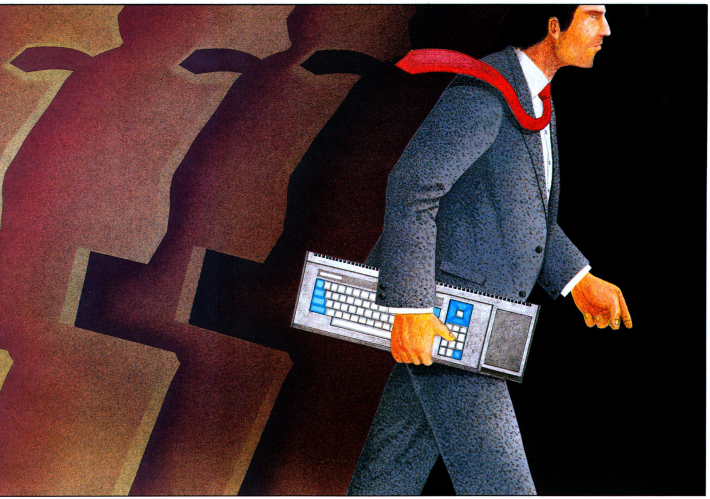
Al final de estos mensajes iniciales, puede verse un bloque cuadrado de color; se le llama CURSOR e indica dónde aparecerá la letra que tecleemos a continuación (podemos comprobarlo pulsando dos o tres teclas).

Teclado + ingenio = programa

Resulta obvio que el teclado es de capital importancia, ya que a través suyo podemos comunicarnos con la máquina. Fundamentalmente es un teclado tipo máquina de escribir standard *(tipo QWERTY)* con algunas teclas de más y un bloque numérico a la derecha del **«teclado principal».**

Obsérvese que existen teclas separadas para la letra «O» y el número «0»; éste siempre aparecerá cruzado por una línea, sea tecleado desde el teclado principal o desde el numérico. Es muy importante recordar esta diferencia entre 0 y O; podemos garantizar que gran cantidad de «inexplicables» errores en tus primeros programas sucederán por confundirlos. Encima del bloque numérico hay un grupo de 5 teclas, una etiquetada como COPY y rodeada por cuatro con flechas. De momento, las ignoraremos.

Volviendo al teclado principal, vereis que además de las letras y números *(teclas «alfanuméricas»,* en jerga de ordenadores), hay otras que destacan por un color diferente al resto. Algunas, como Caps Lock y Shift, nos serán familiares de las máquinas de escribir. Otras, como Ctrl y Esc, deberían ser algo nuevo.



Antes de continuar, vamos a introducir una pequeña convención que nos facilitará las cosas. Si es necesario pulsar la tecla SHIFT, en el texto se indicará

[SHIFT]

Si, por el contrario, aparece SHIFT, sin corchetes, queremos indicar que tecleéis primero la **«S»,** luego la **«H»,** etc.

La tecla [ENTER] es muy importante, y hay dos: una grande en el teclado principal y otra pequeñita en el numérico. Se usa de manera similar a la de una máquina de escribir, para asegurarse de que la escritura continúa en la línea siguiente.

También, y esa es la diferencia, sirve para indicarle al AMSTRAD que hemos terminado de introducirle información y puede comenzar a procesarla.

A estas alturas, la pantalla del ordenador debe tener algo como lo siguiente:

Si no lo has hecho hasta ahora, escribe unas cuantas letras y pulsa ENTER. Se obtendrá este resultado:

No preocuparse por el mensaje de «error de sintaxis» (SYNTAX ERROR en inglés); el AMSTRAD no puede estropearse por escribir cosas en él. Podemos experimentar con toda libertad.

No sabe, no contesta

Todo lo que el **«SYNTAX ERROR»** quiere decir es que el ordenador no comprende las palabras u órdenes que le hemos introducido *(los COMANDOS).* Necesita que se lo digamos en su propio lenguaje, BASIC. El Basic es muy similar al inglés, pero sólo usa unas cuantas palabras de ese idioma para hacer la vida un poco más fácil al AMSTRAD; como hemos comentado antes, esas palabras se denominan COMANDOS *(en inglés, KEYWORDS, palabras-clave).*

Por ejemplo, podemos decirle al ordenador que un programa Basic ha terminado con el comando END *(en español, FIN).* Tecleemos END y su equivalente FIN en nuestra lengua, pulsado [ENTER] después de cada palabra.

La máquina sólo acepta la primera, que ya tiene incorporada en el **«vocabulario»,** COMO TODOS LOS COMANDOS BASIC.

Realmente, esto no parece tener mucho sentido, porque no hay programa alguno al que pueda aplicarse el dichoso END, ¿verdad? De acuerdo, así es; pero por lo menos el irritante SYNTAX ERROR no ha hecho acto de presencia, y hemos comprendido la diferencia entre

COMANDOS y «otros», que puede resumirse así: a la mínima dificultad, el ordenador se lava las manos y nos informa de que no entiende ni media, y que lo resolvamos nosotros, que para eso estamos. Doloroso pero inevitable.

También en mayúsculas

Los impacientes ya habrán descubierto que si se pulsa simultáneamente [SHIFT] y una letra, ésta aparece en mayúsculas. Intentadlo, por favor, si no lo habéis hecho antes. Es como esperabamos, igual que una máquina de escribir. Tampoco produce mucha sorpresa el hecho de que, si pulsamos a la vez [SHIFT]+6, por ejemplo, obtenemos el signo «&». En general, aparecerá en pantalla el signo que se encuentra en la mitad superior de la tecla, si lo tiene; de lo contrario obtendremos una mayúscula. Análogamente, la tecla [CAPS LOCK], al pulsarla, actúa como si presionaramos [SHIFT] continuamente; inténtalo. Si volvemos a pulsarla, volvemos al estado normal, es decir, pulsando **:«d»,** por ejemplo, aparecerá en minúscula *(en el lenguaje informático, estos comandos que permutan entre dos estados diferentes suelen llamarse «conmutadores», traducción libre de la palabra inglesa «toggling»).*

En la duda... probar

¿Qué habría pasado si anteriormente, al teclear «end» lo hubiéramos hecho en mayúsculas? Una de las cosas más agradables en el uso de ordenadores, que uno llega a apreciar con el tiempo, es que la especulación no tiene cabida. Si te sorprendes a ti mismo preguntándote «¿qué pasaría si...?», lo único que hay que hacer es probarlo, directamente. Como ya dije, es imposible estropear el AMSTRAD tecleando cosas, y, creedme, la experimentación es la mejor forma de aprender cómo programar; probablemente la UNICA forma.

Por tanto, teclead por favor:

END [ENTER]

(insistimos por última vez en que [ENTER] quiere decir PULSAR la tecla ENTER, no teclear E, N, T, E, R). El micro actúa exactamente igual que cuando lo hacíamos en minúsculas. De hecho, al AMSTRAD le trae sin cuidado el tamaño de la letra con que se le introduzcan los comandos (no como otro micros, más exigentes, que nos condenan a hablarles siempre de la misma manera).

La autorrepetición, en defensa de nuestros dedos

Otra cosa interesante de la que es probable que el lector ya se haya dado cuenta, es que las teclas poseen AUTORREPETICION, es decir, una vez pulsada una tecla, si se mantiene así, la letra que representa aparece en pantalla una y otra vez.

Pulsa [ENTER] para llevar el cursor a una línea ignorando cualquier inoportuno SYNTAX ERROR, y usa la facultad de autorrepetición con cualquier tecla. Debiéramos obtener una línea de caracteres con el cursor al final de la misma, esperando pacientemente. Ahora, si pulsamos la tecla [DEL], el cursor se desplaza a la izquierda y «devora» al carácter que estaba allí hace un instante. Esta es la manera más simple de corregir errores de tipografía o eliminar caracteres indeseados: pulsar [DEL] *(en inglés DELeting)*. También tiene autorrepetición, así que si la mantenemos pulsada, se comerá nuestra recién nacida línea de caracteres hasta que no sobreviva ni uno solo; en ese momento, el cursor se negará a moverse y un irritado **«bip»** nos recordará que no podemos borrar más en esa línea de la pantalla.

A este tipo de correcciones se le conoce como «EDITAR TEXTO», y existen formas más sofisticadas de llevarlo a cabo, involucrando a las teclas COPY, CTRL y las «flechas» de movimiento del cursor, como más tarde veremos.

Por ahora, con todos estos experimentos, la pantalla original debe estar más o menos llena de texto; tanto si es así como si no, pulsar [ENTER] varias veces para ver el efecto de «SCROLLING» *(la pantalla sube hacia arriba para permitir introducir más texto)*.

Si se ha tenido la paciencia suficiente para pulsar [ENTER] hasta que todo el texto desaparezca, tendremos naturalmente una pantalla en blanco con el cursor al final; como quien no quiere la cosa, acabamos de borrar la pantalla.

¡Tranquilos! Hay formas más fáciles de hacerlo. Si llenamos la pantalla de caracteres y tecleamos:

cls [ENTER]

deberían desaparecer todos y encontrarnos solamente el «READY». Esta palabra, cls, es un COMANDO que cumple la función de borrar la pantalla de texto y deriva de las voces inglesas (una vez más) CLear Screen. Los escépticos pueden comprobar que:

CLS [ENTER]

también borra la pantalla.

Por el momento, esto es todo respecto al teclado. Aunque hay más teclas que cumplen funciones importantes, ESC, CTRL, etc., no es imprescindible ocuparnos de ellas ahora; se supone que estábamos hablando de programación.

Oír es obedecer

Alguien que sabía mucho de informática definió a los ordenadores como **«gigantes estúpidos»** *(efectivamente, era un catedrático)*, lo cual, en palabras sencillas, quiere decir que los computadores, nuestro AMSTRAD, ejecutan LITERALMENTE las órdenes que reciben; no hacen menos, pero tampoco más. Así, supongamos que queremos sumar 2+2, y llenos de alegría por haber encontrado un uso para nuestra máquina, le ordenamos que lo haga. El AMSTRAD lo hará a una velocidad de vértigo, con una precisión de docenas de decimales, pero pasará por alto el pequeño detalle de COMUNICARNOS EL RESULTADO. Recuerden, SOLO le hemos ordenado que sume dos números, ¡no que nos diga cuánto suman!

Para decirle al ordenador que escriba cosas en la pantalla, tenemos que usar el comando Basic PRINT *(recuerdo de cuando el resultado de un programa u orden se plasmaba en papel, en lugar de un monitor)*.

Así, para VER el resultado de 2+2, debemos teclear:

print 2+2 [ENTER]

Observad que no se necesita el signo «=» como en el caso de una calculadora. De la misma forma que el micro no permite sustituir el 0 por la O, tampoco se puede usar «x» o «X» como signo de multiplicar; sólo reconoce el signo «*», como en el caso de:

PRINT 4*3 [ENTER]

El signo «—», de restar, permanece inalterado; se encontrará en la misma tecla que «=». Para dividir, sin embargo, se utiliza una barra vertical «/» como en:

PRINT 12/4 [ENTER]

A esta altura de nuestra disgresión, ya queda claro que el AMSTRAD, antes de procesar cualquier orden, debe ser notificado de ello tecleando [ENTER] al final de la misma; por tanto, a partir de ahora, lo omitiremos. ¡Asegurarse de que vosotros no hacéis otro tanto!

Por favor, probad ahora la siguiente secuencia de órdenes:

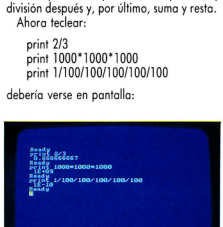
```
PRINT 2+8—3
PRINT 4*8/2
PRINT 4*8+2
PRINT 4*(8+2)
```

Pensando un poco acerca de los resultados, descubriremos algo importante y muy de agradecer: el AMSTRAD interpreta las secuencias de operaciones en el mismo orden en el que lo aprendimos en la escuela, esto es, lo que va entre paréntesis, primero, multiplicación y división después y, por último, suma y resta.

Ahora teclear:

```
print 2/3
print 1000*1000*1000
print 1/100/100/100/100/100
```

debería verse en pantalla:



Primeros pasos

Dentro de un orden

El punto a destacar es que el micro posee un límite de precisión. Por ejemplo, 2/3 no es exactamente 0.666666667. El error está por debajo del millón, pero existe, y habrá algunos programas en los que tengamos que tenerlo en cuenta.

De forma similar, con números muy grandes o muy pequeños, el ordenador ahorra memoria representándolos en NOTACION CIENTIFICA, también llamada FORMATO EXPONENCIAL; así, 1000*1000*1000, en lugar de aparecer como:

1000000000

se muestra en pantalla como:

1 E9

que debe leerse como **«uno multiplicado por diez elevado a la novena potencia»,** esto es, un 1 seguido de 9 ceros.

Análogamente:

1/100/100/100/100/100

producirá 1 E-10, es decir, 1 dividido por un 1 seguido de 10 ceros, 0.0000000001, la respuesta correcta.

Si todo este fárrago numérico no resulta inmediatamente claro, no hay que preocuparse demasiado; cuando sea necesario se explicará con mayor detalle; por suerte o por desgracia, el cálculo es una parte importante de casi todos los programas, y es conveniente dominar sus bases de cara al ordenador aunque pueda parecer un poco árido.

No sólo números

Vamos a intentar ahora algo un poco más alegre: escribir frases en la pantalla. Si tecleamos:

```
print «AMSTRAD»
```

el ordenador escribirá AMSTRAD en la posición del cursor.

Obsérvese que la palabra que queremos escribir va entrecomillada, y, sin embargo, las comillas no aparecen en pantalla. A la frase entrecomillada se la denomina «cadena alfanumérica», nada menos, o «string literal» en inglés. Cadena porque el AMSTRAD lo trata como una serie de caracteres, uno detrás de otro, y literal porque se escribe en pantalla lo que aparece entre comillas exclusivamente. Así:

```
print «AMSTRAD»
print « AMSTRAD»
print «  AMSTRAD»
```

produce una salida diferente porque el número de espacios que preceden a la palabra es distinto.

Los **«strings»** no tienen porque ser palabras solamente; podemos escribir números o cualquier símbolo, con tal que vaya entrecomillado. Probad con:

```
print «4*3»
PRINT 4*3
```

para ver la diferencia. Experimentad con distintos mensajes literales; ¿cuán largos pueden ser?

¿Qué es un programa?

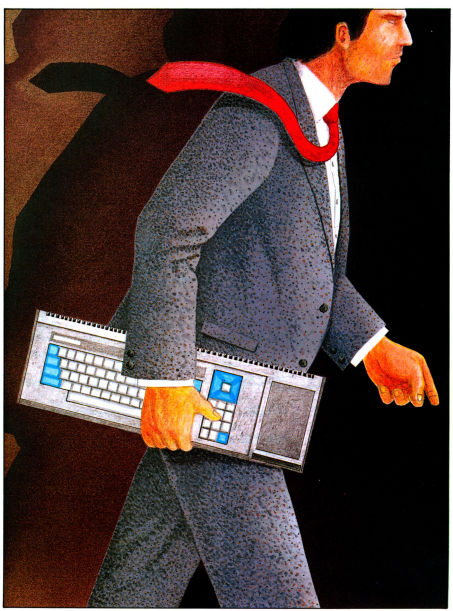
Más de uno habrá pensado que si esto es la programación, ¡pues vaya rollo! Escribir paso a paso la suma de 2 y 2 no reviste dificultad, pero si hay que hacer cientos, o miles de sumas o otros cálculos, ¿qué?

Lo ideal sería dar al ordenador toda una serie de instrucciones, lo más genéricas posible, para que las fuera ejecutando paso a paso y sirvieran para diferentes tareas con mínimos cambios.

Bien, pues el que pensara eso dio en el clavo. Tal secuencia de instrucciones con esas características es lo que llamamos UN PROGRAMA, y comenzaremos a escribirlos y hacerlos funcionar la próxima semana.

Antes de despedirnos, quisiéramos realizar una aclaración importante. Esta serie de artículos van destinados a todas aquellas personas que desean aprender a programar desde el principio. Por ello, pensamos que tratar en profundidad unos temas que para muchos lectores, sin duda, parecerán elementales, cumple el doble objeto de refrescar conceptos básicos para algunos, y evitar la angustiosa situación que nosotros mismos padecimos de tener que partir de cero sin ninguna ayuda en absoluto, lenta y trabajosamente.

De cualquier forma, la vivacidad de un semanario nos permitirá analizar pronto temas más importantes, sin perder de vista la claridad y la minuciosidad.



CODIGO SECRETO

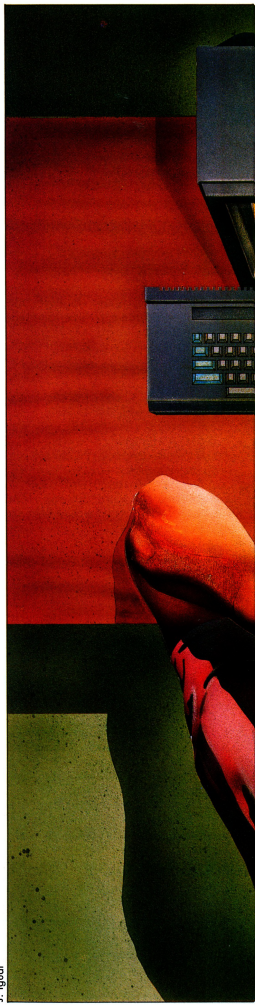
El programa es una versión del conocido juego del Master Mind realizada con bastante ingenio. Para los que no lo sepan, el objetivo del juego consiste en adivinar, en el menor número de intentos posibles, una combinación de colores «propuesta» por el ordenador; se trata de acertar no sólo el color, sino también el lugar, dentro de una secuencia dada, donde dicho color se encuentra. Si el ordenador, al escoger aleatoriamente su secuencia de colores **«decide»** que hay rojo en el lugar número 3, y nosotros decimos que hay rojo en el número 2, pues hemos fallado.

La máquina nos lo hará saber mediante un código también de color: un objeto negro indica color y lugar correctos, mientras que un objeto blanco marca color correcto en lugar equivocado.

El programa se ha reestructurado lo más posible, para evitar las sentencias GOTO que le restarían claridad. Existe un bloque de programa que llama a una serie de subrutinas cuando son necesarias *(líneas 50-180)*. Cada una de ellas está señalada con una sentencia REM, y separada de las demás por dos puntos *(por ejemplo, líneas 570 y 730)*.

CODIGO SECRETO (SUBRUTINAS)

NOMBRE	NUMERO DE LINEA	FUNCION
INICIALIZACION	**230-350**	*Define las envolventes de tono y volumen y el carácter empleado para señalar los aciertos o errores (la «clavija»). Dimensiona las matrices, configura los colores y desactiva la autorrepetición de las teclas.*
DIBUJA TABLERO	**360-570**	*Los caracteres para el tablero se encuentran en las sentencias DATA. Imprime las teclas y colores usados.*
LECTURA E IMPRESION	**580-620**	*Se le llama para leer las DATAS y dibujar el tablero.*
ESTABLECE VARIABLES	**640-720**	*Inicializa las variables «guesses», «correct» y «level».*
GENERA CODIGO	**740-790**	*Crea el código secreto.*
INSERTA TU CLAVE	**810-970**	*«INPUT» nuestra elección. Borra si DEL se pulsa.*
COMPARA CLAVE	**990-1100**	*¿Hemos acertado?*
SEÑAL NEGRA	**1120-1170**	*... si el color es correcto y en el lugar correcto.*
SEÑAL BLANCA	**1190-1230**	*... si el color correcto está en lugar equivocado.*
FIN DE JUEGO	**1250-1450**	*Imprime un mensaje y hace un sonido. Muestra el código secreto y pregunta si se desea jugar otra vez.*
INSTRUCCIONES	**1470-1680**	*Imprime las instrucciones.*

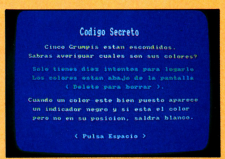


J. Igual

Serie Oro

```
10 REM  ********************
20 REM ** Codigo Secreto **
30 REM **By R.A.Waddilove**
40 REM ********************
50 MODE 1
60 GOSUB 1470
70 MODE 0
80 GOSUB 280
90 WHILE K%=43
100 GOSUB 640:GOSUB 360:GOSUB 740
110 WHILE correct<5 AND guesses<10
120 GOSUB 810:GOSUB 990
130 WEND
140 GOSUB 1250
150 WEND
160 SPEED KEY 25,2
170 MODE 1:INK 1,24:PEN 1
180 END
190 :
200 REM ** inicializacion **
210 SPEED KEY 255,255
220 ENV 1,10,1,1
230 ENT 1,100,-1,1
240 ENT 2,100,-1,1,1,8,100,79,1,1
250 ENT 3,100,-1,1,1,8,150,95,1,1
260 SYMBOL 255,124,124,124,56,56,56,56,1
6
270 DIM code(5),guess(5),markedguess(5),
markedanswer(5)
280 FOR i%=0 TO 15
290 READ j%:INK i%,j%
300 NEXT
310 K%=43
320 RETURN
330 :
340 DATA 1,6,9,24,11,15,17,26,7,16,13,16
,3,10,12,0
350 :
360 REM ** pinta tablero **
370 CLS:PEN 8:RESTORE 540
380 PRINT "   Codigo Secreto":PRINT
390 PEN 10
400 FOR i%=10 TO 1 STEP -1:PRINT i%:PRIN
T:NEXT
410 PEN 13:LOCATE 1,2:GOSUB 590
420 FOR i%=2 TO 20 STEP 2
430 RESTORE 550:GOSUB 590:GOSUB 590
440 NEXT
450 GOSUB 590:GOSUB 590
460 PEN 10:LOCATE 5,23:PRINT "? ? ? ? ?"
470 LOCATE 1,25:PEN 14:PRINT "Color";
480 FOR i%=1 TO level
490 PEN i%:PRINT i%;
500 IF i%<7 THEN PRINT CHR$(8);
510 NEXT
520 RETURN
530 :
540 DATA 158,154,154,158,154,154,154,154
,154,154,154,154,154,158,154,154,154,154
,154,156
550 DATA 149,9,9,149,9,9,9,9,9,9,9,9,9,1
49,9,9,9,9,9,149,151,154,154,159,154,154
,154,154,154,154,154,154,154,159,154,154
,154,154,154,157
560 DATA 149,9,9,149,9,9,9,9,9,9,9,9,9,1
49,9,9,9,9,9,149,147,154,154,155,154,154
,154,154,154,154,154,154,154,155,154,154
,154,154,154,153
570 :
```

```
500 REM ** lectura y pintado **
590 FOR K%=1 TO 20
600 READ ;X:PRINT CHR$(;X);
610 NEXT
620 RETURN
630 :
640 REM ** establece variables **
650 guesses=0:correct=0:level=0
660 CLS:PEN 1:LOCATE 2,5
670 PRINT "Cuantos colores ?"
680 LOCATE 5,10:PRINT "Pulsa 4 - 7"
690 WHILE level<4 OR level>7
700 level=VAL(INKEY$)
710 WEND
720 RETURN
730 :
740 REM ** genera codigo **
750 RANDOMIZE TIME
760 FOR i%=1 TO 5
770 code(i%)=INT(RND*level)+1
780 NEXT
790 RETURN
800 :
810 REM ** inserta tu clave **
820 WHILE INKEY$<>"":WEND
830 guesses=guesses+1
840 LOCATE 5,23-2*guesses
850 i%=0
860 WHILE i%<5
870 i%=i%+1:k$=""
880 WHILE k$("1" OR k%)CHR$(level+48)
890 k$=INKEY$
900 IF i%>1 AND k$=CHR$(127) THEN PRINT
CHR$(8);CHR$(8);" ";CHR$(8);:i%=i%-1
910 WEND
920 SOUND 4,239,5
930 PEN VAL(k$)
940 PRINT CHR$(224);CHR$(9);
950 guess(i%)=VAL(k$)
960 WEND
970 RETURN
980 :
990 REM ** compara la clave **
1000 FOR i%=1 TO 5
1010 IF guess(i%)=code(i%) THEN GOSUB 11
20 ELSE markedanswer(i%)=0:markedguess(i
%)=0
1020 NEXT
1030 IF correct=5 THEN RETURN
1040 correct=0
1050 FOR i%=1 TO 5
1060 FOR ;X=1 TO 5
1070 IF guess(;X)=code(i%) AND markedans
wer(i%)=0 AND markedguess(;X)=0 THEN GOS
UB 1190
1080 NEXT
1090 NEXT
1100 RETURN
1110 :
1120 REM ** senal negra **
1130 PEN 15:PRINT CHR$(255);
1140 correct=correct+1
1150 markedanswer(i%)=-1
1160 markedguess(i%)=-1
1170 RETURN
1180 :
1190 REM ** senal blanca **
1200 PEN 7:PRINT CHR$(255);
1210 markedanswer(i%)=-1
```



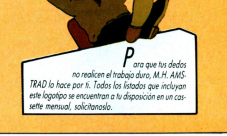

```
1220 markedguess(;X)=-1
1230 RETURN
1240 :
1250 REM ** game over **
1260 PEN 8:PRINT CHR$(30);
1270 IF correct=5 THEN PRINT "     FELIC
IDADES     ":SOUND 4,168,500,3,1,1:SOUND
2,180,500,2,1,2:SOUND 1,195,500,1,1,3 EL
SE SOUND 4,1000,500,15:PRINT "*** MALA
SUERTE ***"
1280 INK 8,7,1
1290 FOR i%=1 TO 10000:NEXT
1300 INK 8,7
1310 PRINT CHR$(30);"   Codigo Secreto
   "
1320 LOCATE 3,23:PEN 12:INK 12,0,1
1330 PRINT CHR$(243);CHR$(9);:PAPER 12
1340 FOR i%=1 TO 5
1350 PEN code(i%)
1360 PRINT CHR$(225);CHR$(9);CHR$(7);
1370 FOR ;X=1 TO 2000:NEXT
1380 NEXT
1390 LOCATE 1,25:PEN 8:PAPER 0
```

```
1400 PRINT "Otro juego (S/N) ? ";:K%=0
1410 WHILE K%=0
1420 K%=(43 AND INKEY(68))-1)+(46 AND IN
KEY(46))-1)
1430 WEND
1440 WHILE INKEY$()"":WEND
1450 RETURN
1460 :
1470 REM ** instrucciones **
1480 INK 0,1:INK 1,1:INK 2,18:INK 3,26
1490 BORDER 1:PAPER 0:PEN 1:CLS
1500 :LOCATE 1,25
1510 PRINT "Codigo Secreto"
1520 FOR i%=0 TO 258 STEP 2
1530 FOR ;X=0 TO 16 STEP 2
1540 IF TEST(i%,;X) THEN PLOT 208+i%,366
+;X*2,3:PLOT 208+i%,368+;X*2,3
1550 NEXT
1560 NEXT
1570 LOCATE 1,25:PRINT SPACE$(20);
1580 INK 1,24:PEN 1:LOCATE 5,5
1590 PRINT "Cinco Grumpis estan escondid
os.":PRINT:PRINT "Sabras averiguar cuale
s son sus colores?"
1600 PEN 2:PRINT
```


```
1610 PRINT " Solo tienes diez intentos p
ara lograr!":PRINT:PRINT " Los colores e
stan abajo de la pantalla":PRINT:PRINT"
        ( Delete para borrar ).'
1620 PEN 1:PRINT:PRINT
1630 PRINT "Cuando un color este bien pu
esto aparece":PRINT " un indicador negro
 y si esta el color ":PRINT:PRINT " pero
 no en su posicion, saldra blanco."
1640 PEN 3:LOCATE 12,25
1650 PRINT "( Pulsa Espacio )";
1660 WHILE INKEY$()"":WEND
1670 WHILE INKEY$()" ":WEND
1680 RETURN
```

CODIGO SECRETO (VARIABLES)

CODE(5)	EL CODIGO SECRETO
GUESS(5)	NUESTRA ELECCION
MARKEDGUESS(5)	EN ESTA MATRIZ, NUESTRA ELECCION QUEDA ALMACENADA
MARKEDANSWER(5)	MEMORIZA NUESTRA RESPUESTA
I%,J%	VARIABLES INDICE EMPLEADAS EN LOS BUCLES
LEVEL	NUMERO DE COLORES
GUESSES	NUMERO DE INTENTOS
CORRECT	NUMERO CORRECTO
K$	VALOR DE LA TECLA PULSADA
K%	VARIABLE DE USO GENERAL



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.

EXPLORANDO LOS GRAFICOS

Análisis

Esta semana AMSTRAD ANALISIS os trae un programa corto pero eficaz para averiguar como trabajan los comandos ORIGIN, MOVE y DRAW.

El desglose de las líneas es el siguiente:

10 Título del programa. Recordad que todo lo que sigue a una sentencia REM es ignorado por el micro.

20 Borra la pantalla y coloca al AMSTRAD en modo 1.

30 El comando ORIGIN se usa para fijar el origen de gráficos *(el punto de la pantalla que los comandos MOVE y DRAW tratan como 0,0)*. Normalmente es la esquina inferior izquierda de la pantalla, pero aquí escogemos el centro como origen de coordenadas.

40-70 Bucle FOR...NEXT cuya variable de control es **«unit»**. El STEP 10 quiere decir que **«unit»** varía desde 0 a 190 incrementados de 10 a 10 a cada iteración.

50 Llama a la subrutina que dibuja los cuadros. A cada vuelta, el valor de **«unit»** es distinto y, puesto que la subrutina lo usa para calcular donde debe dibujar las líneas, se producirán cuadros de diferentes tamaños cada vez que dicha subrutina se ejecuta.

60 Otra llamada a una subrutina, responsable del dibujo de los rombos. De nuevo, cada uno es diferente por la misma razón que en el caso anterior.

80 Cuando el bucle ha finalizado, se activa la subrutina que pinta las líneas que completan el dibujo.

90 Esto es una estructura WHILE...WEND sin fin, puesto que la condición 1 = 1 es siempre cierta. Se ha puesto para evitar la aparición del mensaje READY *(por supuesto, pulsando ESCAPE dos veces se detendrá el programa)*.

100-160 Estas líneas forman la rutina que dibuja cada cuadrado. La figura 1 muestra como las coordenadas de los cuadrados se relacionan unas con otras en términos de la variable **«unit»** y del origen 0,0. A medida que **«unit»** aumenta su valor, los lados de los cuadrados aumentan su longitud.

170-230 La segunda subrutina. Su misión es dibujar rombos. La figura 2 muestra las coordenadas básicas de cada rombo. Su tamaño también es variable con **«unit»**.

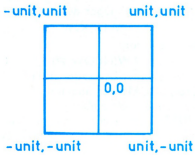


Figura 1

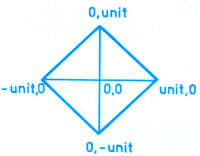


Figura 2

240-330 Por fin, la última rutina. Se la llama cuando el bucle FOR...NEXT ha terminado de dibujar todos los cuadrados y rombos. Ella pinta los ejes y diagonales finales.

VENTANAS (y II)

La semana pasada tuvimos ocasión de aclarar con detalle que era una ventana de texto, y cómo utilizar en ella los comandos relativos a la escritura en pantalla. En este artículo, nos introduciremos más profundamente en el tema, para concluir manejando a la vez las 8 ventanas de las que dispone el AMSTRAD y algún otro «malabarismo más».

El *programa 1* muestra cómo restringir el texto a la parte izquierda de la pantalla. La línea número 30 define la ventana que queremos. Recordad que el comando WINDOW usa, como mínimo, 4 parámetros, de los cuales los dos primeros dan la posición de las esquinas izquierda y derecha de la ventana, respectivamente; en este ejemplo hemos restringido la posición derecha a la mitad de la pantalla (estamos en MODE 1). Los dos parámetros restantes especifican las filas superior e inferior de la ventana; normalmente son 1 y 25, pero aquí se han reducido a 1 y 24. La *figura 1* explica lo que sucede.

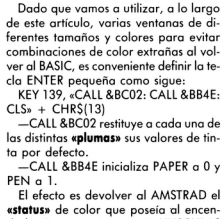


Figura 1: WINDOW 1,20,10,24 en MODE 1

Dado que vamos a utilizar, a lo largo de este artículo, varias ventanas de diferentes tamaños y colores para evitar combinaciones de color extrañas al volver al BASIC, es conveniente definir la tecla ENTER pequeña como sigue:

KEY 139, «CALL &BC02: CALL &BB4E: CLS» + CHR$(13)

—CALL &BC02 restituye a cada una de las distintas **«plumas»** sus valores de tinta por defecto.

—CALL &BB4E inicializa PAPER a 0 y PEN a 1.

El efecto es devolver al AMSTRAD el **«status»** de color que poseía al encenderlo. Es necesario usar MAYUSCULAS al definir dicha tecla, porque el BASIC no convertirá minúsculas a mayúsculas al tratarse de un texto entrecomillado.

Tras esta medida precautoria, concentremos la atención en el *programa 2*. En la línea 30, en lugar de:

WINDOW

se ha escrito:

WINDOW #0

También los comandos PAPER, PEN, CLS y LOCATE han adquirido su correspondiente 0. Sin embargo, si se ejecuta (RUN) el programa 2, su resultado es idéntico en todo al programa 1. **¿Qué ocurre aquí?**.

AMSTRAD: hasta 8 ventanas en pantalla

Recuérdese la posibilidad de mantener SIMULTANEAMENTE en pantalla hasta 8 ventanas; para distinguir entre ellas, se le asigna a cada una un **«numeral de cauce»**, o sea, un número del 0 al 7 precedido por el signo #. Esta misma convención se emplea para los comandos mencionados anteriormente cuando queremos dirigir su salida a alguna ventana de texto en particular *(líneas 40-70 del programa 2)*. Ahora bien, la #0 es la **«VENTANA POR DEFECTO»** del sistema, y queda sobreentendida si empleamos el comando WINDOW sin numeral de cauce; de aquí que la salida de los programas 1 y 2 sea idéntica *(obsérvese la línea 80, donde #0 se ha omitido)*.

El sistema posee su propia ventana por defecto

El *programa 3* demuestra que pueden existir otras ventanas en pantalla además de WINDOW #0. Nos hemos referido, en este caso, a la número 1. Si ejecutamos el programa, no notaremos nada nuevo hasta que pulsemos una tecla para dar fin al programa y que aparezca el mensaje READY *(línea 90)*. En ese momento, dicho mensaje se ve en la esquina superior izquierda de la pantalla, no dentro de la ventana recién definida, como antes.

El motivo es que los mensajes del sistema, tales como READY o PRESS PLAY siempre se envían a la pantalla a través de la ventana #0, al igual que los comandos BASIC. Como el programa 3 no la altera, el mensaje READY se muestra en la pantalla **«normal»**.

Hasta ahora, sólo se ha empleado una ventana a la vez; el *programa 4* **«abre»** dos simultáneamente, la #0 y la #1. Antes de ejecutarlo, puede ser interesante tratar de predecir su resultado; se haya hecho o no, deben verse 2 mensajes cada uno en su respectiva ventana.

El comando LIST *(o LIST #0)* mostrará el listado del programa en la ventana de la izquierda; el intento LIST #1 lo hará aparecer en la otra; podemos listar por diferentes ventanas, según nos interese.

Observando un poco más detenidamente el programa 4, vemos que las dos ventanas se definen en las líneas 30 y 40; el área de pantalla dedicada a cada una se muestra en la figura 2.

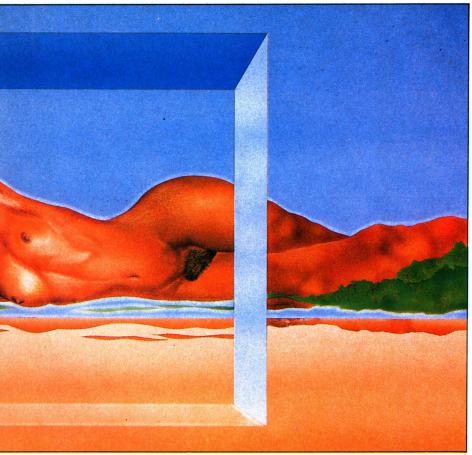
La línea 50 especifica PEN 2 para la ventana #0 con:

PEN #0,2

Esta sentencia nos da texto cyan, mientras que la línea 60 asigna a la otra, la número 1, PEN 3 *(texto rojo)*, mediante:

PEN #1,3

Es necesario recordar que el comando LOCATE dirige su salida a una ventana en concreto, esto es, el sistema de coordenadas para LOCATE #n comienza en la esquina superior izquierda de la ventana etiquetada WINDOW #n.



J. Igual

Así:

LOCATE #0,3,8 (línea 70)

colocará el cursor de texto en la TERCERA COLUMNA y LA OCTAVA FILA de la ventana #0. Análogamente, la línea 90 realizará la misma función en la #1.

Cada ventana tiene asignado su propio cursor

El sistema operativo del ordenador **«recuerda»** la posición de cada cursor de texto asignado a determinada ventana; para convencerse de que ambos cursores están realmente separados, añadir las líneas siguientes al programa 4:

```
105 PRINT #0, TAB(9) «CERO»;
106 PRINT #1, «UNO»
```

Como se puede ver después de ejecutar el programa, «manipular» la posición del cursor de texto en una ventana (la 0), no afecta para nada a su posición en la otra. La extensión de la ventana no queda reflejada por la escritura de los mensajes. Añade las siguientes líneas y verás lo que queremos decir:

```
45 PAPER #0,3: CLS #0
46 PAPER #1,1: CLS #1
```

Canales y corrientes

Antes de continuar, una puntualización acerca de terminología: hemos dicho que la instrucción PRINT #n escribe en la ventana #n.

Desde ahora diremos que lo hace en la CORRIENTE #n *(de la palabra inglesa STREAM)*, lo que sucede es que el flujo de datos de la corriente **«apunta»** a la ventana n. Las corrientes son diferentes canales o vías que el ordenador emplea para separar sus flujos de INPUT (entrada de datos) y OUTPUT *(salida de datos)*. Hay 9 canales de salida:

0-7	Canales de salida de texto hacia pantalla; las ventanas.
8	Canal asignado a la impresora.
9	Canal de cassette/disco.

y 9 canales de entrada:

0-7	Teclado, cada uno relacionado con una ventana.
8	La lectura de datos también del teclado, pero relacionado con la impresora.
9	Entrada de datos para el cassette/disco.

Nuestras ventanas emplean los números 0-7 como canales de salida de datos. Ejecutad, por favor, el programa 5. Mantiene las 8 ventanas a la vez en la pantalla. Una vez visto, incluyamos las siguientes líneas para ver la diferencia que causan:

```
140 FOR BUCLE%=0 TO 7
150 PAPER #BUCLE%, 10
160 CLS #BUCLE%
170 WHILE INKEY$=" ": WEND
180 NEXT BUCLE%
```

Si dos o más ventanas solapan, borrar con CLS una de ellas: borrará el área común a ambas.

Program Acción

El programa 6 ilustra el caso con tres; la última en ser borrada, es la única que sobrevive intacta, como cabría esperar. Otro tipo de solapamiento es cuando unas están dentro de otras; el programa 7 lo muestra.

Aplicaciones prácticas

Desde el punto de vista técnico, ya sabemos bastante del tema ventanas. Vamos a tratar de ver ahora unos cuantos casos de aplicación práctica. Una de las mayores ventajas de esta técnica de programación consiste en separar la entrada y salida de datos —se usa una ventana para mostrar los mensajes del ordenador, y otra nuestras propias réplicas—. Los juegos de aventuras usan a menudo esta aproximación.

El programa 8 muestra una versión del conocido juego «adivina el número» utilizando 3 ventanas. La primera es para nuestra «entrada de datos» y para los mensajeros del ordenador. En las otras dos se escriben nuestras elecciones, las que sobrepasan al número que debemos adivinar en la de la izquierda, y las menores que él en la de la derecha. El programa se encuentra en estado embrionario; puede mojorarse mucho. Sin embargo, deja claro cómo una adecuada elección de las ventanas puede incrementar la estética y facilidad de uso de cualquier aplicación.

Intercambio de ventanas: WINDOW SWAP

Supongamos que hemos definido dos ventanas, #0 y #1. Como ya sabemos, subsecuentes PRINT #0 se dirigirán a la #0, mientras que todo PRINT #1 afectará a la ventana #1.

En AMSTRAD BASIC existe un comando, WINDOW SWAP, que permite el intercambio completo de dos ventanas.

Por ejemplo, si tecleamos:

WINDOW SWAP 0,1

el área de pantalla etiquetada como

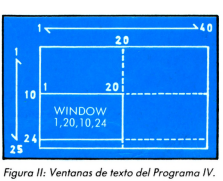


Figura II: Ventanas de texto del Programa IV.

PROGRAMAS

```
10 REM PROGRAMA 1
20 MODE 1
30 WINDOW 1,20,10,24
40 PAPER 2
50 PEN 3
60 CLS
70 LOCATE 3,8
80 PRINT"La ventana de texto"
90 WHILE INKEY$="":WEND

10 REM PROGRAMA 2
20 MODE 1
30 WINDOW #0,1,20,10,24
40 PAPER #0,2
50 PEN #0,3
60 CLS #0
70 LOCATE #0,3,8
80 PRINT#0,"La ventana de texto"
90 WHILE INKEY$="":WEND

10 REM PROGRAMA 3
20 MODE 1
30 WINDOW #1,1,20,10,24
40 PAPER #1,2
50 PEN #1,3
60 CLS #1
70 LOCATE #1,3,8
80 PRINT #1,"La ventana de texto"
90 WHILE INKEY$="":WEND

10 REM PROGRAMA 4
20 MODE 1
30 WINDOW #0,1,20,10,24
40 WINDOW #1,21,40,10,24
50 PEN #0,2
60 PEN #1,3
70 LOCATE #0,3,8
80 PRINT #0,"La ventana de texto"
90 LOCATE #1,3,8
100 PRINT #1,"Otra ventana"
110 WHILE INKEY$="":WEND

10 REM PROGRAMA 5
20 MODE 0
30 PAPER 18:CLS
40 FOR bucle%=3 TO 17 STEP 2
50 numero%=(bucle%-3)/2
60 WINDOW #numero%,bucle%,bucle%+1,1,25
70 PAPER #numero%,numero%+1
80 CLS #numero%
90 PEN #numero%,numero%
100 LOCATE #numero%,1,12
110 PRINT#numero%,numero%
120 NEXT bucle%
130 WHILE INKEY$="":WEND

10 REM PROGRAMA 6
20 MODE 1
30 hor%=5:long%=20
40 ver%=2:inf%=12
50 FOR bucle%=0 TO 2
60 WINDOW #bucle%,hor%,hor%+long%,ver%,v
er%+inf%
70 PAPER #bucle%,bucle%+1
80 CLS #bucle%
90 hor%=hor%+5
100 ver%=ver%+4
110 NEXT bucle%
120 CALL &BCA2
130 CALL &BBAE

10 REM PROGRAMA 7
20 MODE 1
30 hor%=4:long%=32
40 ver%=4:inf%=14
50 FOR bucle%=2 TO 8 STEP -1
60 WINDOW #bucle%,hor%,hor%+long%,ver%,v
er%+inf%
70 PAPER #bucle%,bucle%+1
80 CLS #bucle%
90 hor%=hor%+bucle%/4:long%=long%/2
100 ver%=ver%+inf%/4:inf%=inf%/2
110 NEXT bucle%
120 PEN 3

10 REM PROGRAMA 8
20 WHILE -1
30 GOSUB 120:REM INICIALIZACION
40 WHILE resp%=numero%
50 GOSUB 260:REM DECIR NUMERO
60 IF resp%=numero% THEN GOSUB 350:REM R
ESPUESTA CORRECTA
70 IF resp%<numero% THEN GOSUB 430:REM D
ENSIDAD ALTO
80 IF resp%>numero% THEN GOSUB 500:REM D
ENSIDAD BAJO
90 WEND
100 WEND
110 END
120 REM *** INICIALIZACION ***
130 REM *** ELECCION DE MODO ***
140 REM *** Y DE VENTANAS ***
150 MODE 1
160 PAPER 2:CLS
170 WINDOW #1,9,16,3,14
180 PAPER #1,1:PEN #1,3:CLS #1
190 WINDOW #2,25,31,3,14
200 PAPER #2,3:PEN #2,1:CLS #2
210 WINDOW #3,4,36,19,24
220 PAPER #3,6:PEN #3,1:CLS #3
230 LOCATE #1,2,12:LOCATE #2,2,12
240 resp$=INKEY$:IF resp$="" THEN 240
250 RETURN
260 REM *** PIDE NUMERO ***
270 REM *** RANGO 1-100 ***
280 CLS #3
290 LOCATE #3,6,3
300 numero%=RND(1)*100+1
310 WHILE numero%<1 OR numero%>100
320 INPUT #3,"Numero: ";numero%
330 WEND
340 RETURN
350 REM *** ELECCION CORRECTA ***
360 CLS #3
370 LOCATE #3,5,2
380 PRINT #3, "Bien hecho" numero% "es c
orrecto"
390 LOCATE #3,3,5
400 PRINT #3, "Pulsa espacio para contin
uar"
410 WHILE INKEY(47)<10:WEND
420 RETURN
430 REM *** DENSIDAD ALTO ***
440 CLS #3
450 LOCATE #3,5,2
460 PRINT #3, "Erroneo. " numero% "es de
masiado alto"
470 PRINT #1,TAB(2) numero%
480 GOSUB 570:REM ESPERAR
490 RETURN
500 REM *** DENSIDAD BAJO ***
510 CLS #3
520 LOCATE #3,5,2
530 PRINT #3, "Erroneo. " numero% "es de
masiado bajo"
540 PRINT #2,TAB(2) numero%
550 GOSUB 570
560 RETURN
570 REM *** ESPERAR TECLA ***
580 LOCATE #3,3,5
590 PRINT #3,"Pulsa espacio para adivina
r otro numero"
600 WHILE INKEY(47)<10:CALL &BB1B:WEND
610 RETURN

10 REM PROGRAMA 9
20 MODE 1
30 WINDOW #0,1,20,10,24
40 WINDOW #1,21,40,10,24
50 WHILE -1
60 GOSUB 80
70 WEND
80 REM RUTINA DE MENSAJE
90 CLS #0:CLS #1
100 PRINT #0:PRINT #0:PRINT #0
110 PRINT #0, "Aqui hay dos"
120 PRINT #0, "ventanas de texto"
130 PRINT #0, "Esta es la 0"
140 PRINT #1,"Esta es la 1"
150 PRINT #1, "Ahora hacemos"
160 PRINT #1, "un canje con ellas"
170 PRINT #1
180 PRINT #1, "WINDOW SWAP 0,1"
190 PRINT #1
200 PRINT #1, "Pulsa espacio"
210 PRINT #1, "para continuar"
220 WHILE INKEY(47)=-1:WEND
230 WINDOW SWAP 0,1
240 RETURN

10 REM PROGRAMA 10
20 MODE 1
30 WINDOW #0,1,20,10,24
40 WINDOW #1,21,40,10,24
45 PAPER #0,3:PAPER #1,9:PEN #0,2: PEN
#1,3
50 WHILE -1
60 GOSUB 80
70 WEND
80 REM RUTINA DE MENSAJE
90 CLS #0:CLS #1
100 PRINT #0:PRINT #0:PRINT #0
110 PRINT #0, "Aqui hay dos"
120 PRINT #0, "ventanas de texto"
130 PRINT #0, "Esta es la 0"
140 PRINT #1,"Esta es la 1"
150 PRINT #1, "Ahora hacemos"
160 PRINT #1, "un canje con ellas"
170 PRINT #1
180 PRINT #1, "WINDOW SWAP 0,1"
190 PRINT #1
200 PRINT #1, "Pulsa espacio"
210 PRINT #1, "para continuar"
220 WHILE INKEY(47)=-1:WEND
230 WINDOW SWAP 0,1
240 RETURN

10 REM PROGRAMA 11
20 MODE 1
30 ancho%=20
40 WINDOW #0,1,20,10,24
50 WINDOW #1,21,40,10,24
60 PAPER #0,3:PAPER #1,9:PEN #0,2: PEN #
1,3:CLS #0:CLS #1
70 texto$="Centrado":GOSUB 160
80 texto$="en":GOSUB 160
90 texto$="primer lugar":GOSUB 160
100 WINDOW SWAP 0,1
110 texto$="Centrado":GOSUB 160
120 texto$="en":GOSUB 160
130 texto$="segundo lugar":GOSUB 160
140 WHILE INKEY$="":WEND
150 END
160 REM Rutina de centrado de texto
170 posicion%=(ancho%-LEN(texto$))/2
180 PRINT TAB(posicion%) texto$
190 RETURN
```

«ventana 1» pasa a ser la #0 y viceversa. Así, el comando:

PRINT #0, "AMSTRAD"

causará la aparición de esta palabra en lo que antes era la ventana #1.

En realidad, todo lo que ha ocurrido es un cambio de etiquetas. El efecto puede ser impresionante si se considera que el cambio afectará a todos los siguientes comandos PRINT, PEN y PAPER, en que dichas etiquetas se vean invulocradas.

El programa 9 ejemplifica el resultado del cambio de ventanas; cada vez que se presione la barra espaciadora, las ventanas #0 y #1 se intercambian.

Es muy importante comprender que sólo se intercambian los números que etiquetan a cada ventana. Los atributos como PEN, PAPER, etc., asociados a cada una, permanecen inalterados bajo el nuevo número #n. El programa 10 aclara esta idea: el texto en la ventana izquierda aparece en cyan, mientras que en la derecha es rojo. El cambio de ventanas de la línea 230 no afecta esto; sólo ha cambiado el número, no las características físicas de cada ventana. Tamaño, colores pen/paper, posición del cursor, etc. no se alteran.

Un importante uso del comando WINDOW SWAP estriba en la posibilidad de que más de una ventana pueda compartir la misma subrutina. La que empieza en la línea 160 del programa 11 contiene un método de centrado de texto que opera por defecto en la ventana #0. Habiendo definido dos *(líneas 40-50)*, podemos usar esta rutina para centrar texto en ambas. Para la #0, se hace directamente en las líneas 70-90. Para la #1, la línea 100 efectúa el canje y en las siguientes *(110-130)* centramos nuestro texto.

Una buena diversión para espíritus aventureros es realizar WINDOW SWAP cuando una de las ventanas no está definida. En un alarde de crueldad, nos negamos a delantar el resultado de tal manipulación.

Nada más por esta semana. Creemos que tenéis material más que suficiente para divertiros y aprender experimentando con él.

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

SOY SOCIO DEL
CIRCULO DE SOFT
Y CONSIGO
LOS MEJORES PROGRAMAS
A LOS MEJORES
PRECIOS

¡Y ANTES
QUE
NADIE!

circulo de soft
MICROAMIGO S.A.


¡¡¡SOMOS SOCIOS DE
Y CONSEGUIMOS LOS
A LOS MEJO
¡Y ANTES QUE NADIE!


circulo de soft
L CIRCULO DE SOFT
S MEJORES PROGRAMAS
ORES PRECIOS!!!
Tú también puedes ser socio de Circulo de Soft. ¡No te cuesta nada!
Envía tu nombre, dirección y teléfono, indicando
la marca y modelo de tu ordenador.
¡TE SORPRENDEREMOS!
NOMBRE
DOMICILIO
LOCALIDAD
PROVINCIA
TEL.
C. P.
ORDENADOR
MICROAMIGO, S.A.
P.º DE LA CASTELLANA, 268
28046-MADRID
TEL.: (91) 733 25 00

TODO SON NUMEROS

J. Molina

Aunque los números binarios y el lenguaje máquina suelen producir cierto pánico entre los usuarios sin mucha experiencia, lo cierto es que con un poco de práctica se pueden llegar a dominar a la perfección. Cuestiones tales como descifrar lo que es un nibble, manejar el sistema hexadecimal, o llamar a una subrutina desde código máquina quedan convertidos en un juego de niños si se aplica correctamente la metodología que a continuación exponemos.

Como recordaréis, los números binarios consisten en una serie de unos y ceros conocidos como «bits». Normalmente, suelen ir unidos en grupos de 8, constituyendo un «byte».

%10101100 es un número binario típico; el signo «%» se coloca delante para evitar confusiones con cifras decimales.

¿Cómo podemos interpretarlo? Los valores en decimal dependen de la columna en la que se encuentren, así:

centenas	decenas	unidades
1	0	0

tiene el valor 100, y es muy diferente a escribir:

centenas	decenas	unidades
0	1	0

equivalente a 10 en decimal.

En el sistema binario de numeración, la situación es idéntica; cada columna posee su valor. La *figura 1* los muestra para un solo byte.

Para obtener el equivalente decimal de una cifra escrita en binario, hay que sumar los valores de las columnas que contienen unos. La *figura 2* muestra la conversión de %10101100 en su homólogo decimal 172.

Una vez cogido el truco a las conversiones decimal-binario, y bastará para lograrlo un poco de práctica, resulta que los temidos números binarios son extremadamente sencillos de manejar.

No obstante, un problema inherente a este sistema de numeración es que, desde el punto de vista humano, es muy común cometer errores «tontos» que pueden estropear completamente un programa que costó horas materializar. Por ejemplo, fácilmente podíamos escribir:

%10110101

cuando nuestra intención era anotar:

%10101101

Una manera astuta de obviar tanto uno y cero junto, es dividir los 8 bits en dos grupos de 4, justo por la mitad. En efecto, **¿qué es más fácil de leer?:**

%10110101

o bien:

%1011 0101

Salta a la vista con mucha más sencillez la configuración del número en el segundo caso que en el primero.

¿Un byte o dos nibbles?

Los dos mitades de un byte se denominan **«nibbles»,** palabra inglesa que nos vamos a ver obligados a utilizar, ya que se encuentra por doquier en todos los libros, castellanos o no, que abordan el código máquina. Así pues, al nibble de la izquierda, que contiene los valores numéricos *(de columna)* más altos, se le conoce como «nibble más significativo» (MSN), y al restante, como **«nibble menos significativo»** o LSN *(en efecto, no es un gran derroche de ingenio, pero...)*. La *figura 3* muestra los valores en decimal de las columnas LSN, de la cual puede deducirse al primer vistazo que la cifra más alta que podemos representar, cuando todos los bits son 1 es 15. Naturalmente, la más baja es 0, como siempre.

Columna	3	2	1	0
Valor	8	4	2	1

Figura 3: Valores LSN.

Más de un lector se preguntará, con toda la razón del mundo: **«¿Y a mí qué? Para qué demonios voy a escribir %1111 cuando puedo escribir 15?»**

Vamos a ver qué sucede si a cada una de las distintas configuraciones de nibbles le asignamos un código consistente en un solo carácter. Observando la *figura 4*, resalta el hecho de que el código asignado a cada nibble coincide con su valor numérico (!).

Antes de quitarnos reverentemente el sombrero ante los diseñadores de este sistema, muy bien se podría inquirir acerca del valor %1010 y siguientes, ya que su valor decimal requiere para representarlo más de un dígito. No hay que preocuparse; todo está previsto. Lo que se decidió es asignar al valor %1010 la letra A, al siguiente la B y así sucesivamente hasta la F *(Figura 5)*. Es decir, escogemos el acuerdo de que, para contar del 1 al 15, en lugar de decir 1, 2, 3, ... 10, 11, 12, 13, 14, 15, escribiremos 1, 2, 3, ..., A, B, C, D, E, F (de momento, ignorar lo que sucede cuando llegamos al 16).

Acabamos de asistir al nacimiento del SISTEMA DE NUMERACION HEXADECIMAL, de inmensa trascendencia en el mundo del código máquina.

Aunque el ordenador emplea bytes para «manejarse», se puede usar nuestro nuevo código con ellos, adjudicando a

Número de columna	7	6	5	4	3	2	1	0
Valor	128	64	32	16	8	4	2	1

Figura 1: Valores decimales de cada columna binaria.

Valor	128	64	32	16	8	4	2	1
Bit	1	0	1	0	1	1	0	0
=	128	+	32	+	8	+ 4		
=	172							

Figura 2: Conversión binaria.

Configuración del Nibble	Valor
0000	0
0001	1
0010	2
0011	3
0100	4
0101	5
0110	6
0111	7
1000	8
1001	9
1010	10
1011	11
1100	12
1101	13
1110	14
1111	15

Figura 4: Configuración de cada nibble y sus valores.

cada nibble el número o carácter que corresponda, de la siguiente manera:

binario: %10101100
nibble MSN: 1010
equivalente decimal: 10
equivalente hexadecimal: A
nibble LSN
equivalente decimal: 12
equivalente hexadecimal: C

Retrocediendo a la figura 2, nuestro número binario equivale en decimal a 172; por tanto, AC es otra manera de condificar 172, sólo que en hexadecimal. Para diferenciarla de las otras dos, la prefijamos con el signo «&».

Otro ejemplo:

binario: %10000011
nibble MSN: 1000
decimal: 8
hexadecimal: 8
nibble LSN: 00114
decimal: 3
hexadecimal: 3

En este caso, los códigos decimal y hexadecimal coinciden *(siempre sucederá así para nibble cuyo valor decimal sea menor o igual a 9)*.

Nuestro número es el &83. Puede verse que el prefijo «&», es absolutamente esencial; si no estuviera, no podríamos decir a simple vista si hablamos de 83 ó 131 (en decimal).

Configuración del Nibble	Código	Valor
0000	0	0
0001	1	1
0010	2	2
0011	3	3
0100	4	4
0101	5	5
0110	6	6
0111	7	7
1000	8	8
1001	9	9
1010	A	10
1011	B	11
1100	C	12
1101	D	13
1110	E	14
1111	F	15

Figura 5: Configuración de cada nibble y su código.

Sistema hexadecimal: un byte, 2 caracteres

Con este método podemos codificar cualquier byte en dos caracteres; así, 255 sería &FF *(trata de descubrir por qué)*.

De la misma forma que en un byte cada columna tenía un valor en su codificación binaria, así sucede también en hexadecimal:

%00001111 en hexa es &0F *(decimal 15)*
%00010000 en hexa es &10 *(decimal 16)*

Atención al último ejemplo, pues es un caso análogo al que sucede en decimal al pasar de 9 a 10. De hecho, en ambos casos se suma 1 al número anterior, último de las unidades, y se pasa al primero de las decenas, sólo que, en hexa, las «decenas» TIENEN 15 DIGITOS.

Así:

&10 es 1×16+0 unidades=16 en decimal
&17 es 1×16+7 unidades=23 en decimal
&1B es 1×16+11 unidades=27 en decimal
&20 es 2×16+0 unidades=32 en decimal
&3C es 3×16+12 unidades=60 en decimal
&A3 es 10×16+3 unidades=163 en decimal
etc., etc...

La *figura 6* será de utilidad para realizar conversiones de este tipo.

Dos bytes para manejar valores superiores a 255

La semana pasada adelantamos que el Z80 es capaz de direccionar 65536 posiciones de memoria y que, para representar plenamente este rango de cifras, se necesitaban dos bytes (byte alto y byte bajo) o 16 bits. En hexa, hacen falta solamente 4 caracteres, 2 por cada byte *(figura 7)*.

Como sucedía en el caso de los números binarios, a medida que nos desplazamos hacia la izquierda, el valor decimal de cada columna cambia; en hexa, la cifra se va multiplicando por 16:

columna 1 1= 1
columna 2 1×16= 16
columna 3 16×16= 256
columna 4 256×16= 4096

Si en un número hexadecimal el byte bajo (figura 7) fuera &FF, y quisiéramos añadir 1, es decir, convertirlo en 256, todo lo que habría que hacer es colocar un 1 en la primera columna del byte alto y poner a cero el byte bajo, con lo que se obtendría &100.

Código máquina

Dígito hexa	Valor de cada columna (byte alto)	Valor de cada columna (byte bajo)
0	0	0
1	16	1
2	32	2
3	48	3
4	64	4
5	80	5
6	96	6
7	112	7
8	128	8
9	144	9
A	160	10
B	176	11
C	192	12
D	208	13
E	224	14
F	240	15

Figura 6: Equivalencia hexadecimal.

Todas estas sumas y cálculos pueden parecer complicadas en un principio, sin embargo, la conversión hexadecimal-decimal es muy simple y directa; por ejemplo:

&0202=2×256+2=514
&1A10=1×4096+10×256+1×16=6672
&FFFF=15×4096+15×256+15×16+15×1=65535

Hexadecimal y binario con BASIC

El AMSTRAD puede realizar las conversiones entre los tres sistemas de numeración por nosotros, ahorrándonos todo este fárrago de cálculos, que, sin embargo, conviene realizar a mano al principio para comprender los fundamentos del lenguaje máquina. Para obtener el equivalente hexa de 6672, se puede escribir en BASIC:

```
PRINT HEX&(6672)
```

y veremos en pantalla 1A10, sin signo «&».

La sentencia:

```
PRINT BIN&(25,8)
```

nos permitirá apreciar el equivalente bi-

Columna	Byte alto		Byte bajo	
Número	3	2	1	0
Valor	4096	256	16	1

Figura 7: Valores hexa de los 4 dígitos de un número.

nario de 25 en un campo de 8 caracteres, esto es, 00011001. Si se usa un número mayor que 255, sustituir el 8 por 16.

Con un suspiro de alivio por parte de todo el mundo, espero, digamos que ya hemos tenido bastante de **«teoría básica de números»** para mucho tiempo. Vamos a hacer código máquina.

La semana pasada descubrimos que el lenguaje máquina no es más que una serie de bytes almacenados secuencialmente en la memoria, los cuales le dicen al ordenador que ejecute determinadas acciones. Esta definición del código máquina plantea tres problemas principales:

— Decidir en qué PARTE DE LA MEMORIA lo colocamos.

— Ubicar CORRECTAMENTE dichos bytes, es decir, en un cierto orden.

— Decirle al micro que ejecute la instrucción máquina que corresponda.

Código máquina desde BASIC: POKE y PEEK

El primer punto queda resuelto eligiendo como dirección de almacenamiento la &3000.

En segundo lugar, emplearemos el comando Basic POKE para introducir nuestro código máquina en memoria; esta instrucción tiene dos parámetros separados por una coma: la dirección de memoria y el número (byte) que se va a colocar en ella.

Por tanto:

POKE &3000, &C9

cambiará el contenido de la posición &3000 a &C9 (usamos hexa).

Tecléenlo, por favor; en apariencia nada ha sucedido.

Afortunadamente, existe un comando especial para escépticos conocido como PEEK, capaz de examinar el contenido de una posición de memoria dada como argumento. Si escribimos:

PRINT PEEK (&3000)

obtendremos como resultado 201 *(en decimal)*, es decir, &C9 en hexa. Si todavía se mantiene en la duda, teclee:

PRINT HEX& (PEEK &3000)

y debiera obtener C9. Después de todo, el comando POKE funciona. Ahora, se crea o no se crea, hemos introducido un byte de código máquina en la memoria. &C9 es una instrucción, u opcode *(abreviatura inglesa de código de operación)*, que le dice al ordenador que regrese a la tarea anterior que estuviera realizando.

Nos lo encontramos la semana pasada bajo su etiqueta decimal de 201 y hablamos de su mnemónico, RET.

Para conseguir que el Z80 se digne ejecutar nuestro programa máquina, recurrimos a la instrucción Basic CALL:

CALL &3000

algo así como «GOSUB desde BASIC hasta la rutina en máquina que comienza en la dirección &3000». Por esta razón necesitamos la instrucción RET (&C9) allí. Es el RETURN que **«encaja»** con el **«GOSUB»** del comando CALL.

Así, al teclear CALL &3000 el micro **«salta»** allí y hace lo que le indica el primer byte que encuentra en esa posición; en este caso particular, la acción es **«vuelve a casa»** con lo que regresamos al BASIC y al mensaje READY de la pantalla.

Sí, de acuerdo; no ha ocurrido nada espectacular, pero acabamos de ejecutar con éxito nuestro primer programa en código máquina.

FIRMWARE: Una multitud de rutinas útiles para el programador

Para probarlo, recordaréis que dijimos que el **«firmware»** del AMSTRAD está materialmente repleto de rutinas en lenguaje máquina que ejecutan tareas muy concretas. Conocerlas es fundamental, ¡sobre todo porque ya están escritas!

Vamos a usar una que comienza en la dirección &BBDB, y que borra la pantalla de gráficos. Por favor, teclear:

CALL &BBDB

¿Funciona, verdad?

Todo lo que hay que hacer es llamar a esta rutina desde alguna parte de nuestro propio programa y la pantalla de gráficos será borrada. Como podéis ver, la utilidad del Firmware para el programador en código máquina es enorme.

El Z80 tiene una útil instrucción que podríamos llamar **«gosub a ...»**, cuyo código de operación es &CD (mnemónico CALL, como la instrucción BASIC), y que le dice que la dirección a la que queremos que bifurque para luego regresar, está especificada por los dos bytes inmediatamente siguientes en la memoria.

Por tanto, la serie de bytes:

CD DB BB

borrará la pantalla gráfica. Obsérvese que el Z80 espera encontrar los dos bytes que definen la dirección en el orden contrario al que a primera vista parece más lógico; debe encontrarse primero el byte bajo y luego el alto. También está claro que falta una instrucción esencial en nuestro programa, RET. La secuencia completa de bytes que deben ubicarse empezando en la posición &3000 es:

CD DB BB C9

La *figura 8* lo ilustra en su totalidad.

Para introducirlo en la memoria:

POKE &3000, CD
POKE &3001, DB
POKE &3002, BB
POKE &3003, C9

Si se considera necesario comprobar que todo ha ido bien, usar PEEK. Para ejecutar el programa desde BASIC, bastará con teclear como antes CALL &3000.

Bueno, pues éste es nada menos que el primer programa en código máquina con un efecto visible y tangible que hemos realizado satisfactoriamente.

Dirección	Códigos de operación	Mnemónicos
3000 3003	CD DB BB C9	CALL &BBDB RET

Figura 9: Listado Assembler.

Dirección de memoria	& 3000	& 3001	&3002	&3003
Contenido de la posición de memoria	& CD	& DB	& BB	& C9
Significado de los bytes	CALL (llamada) a la subrutina cuya dirección especifican los 2 bytes siguientes	Byte bajo Dirección de la subrutina requerida por el byte anterior (byte bajo primero)	Byte alto	Retorno al programa principal

Figura 8: Programa para borrar pantalla.

Precaución, cualidad imprescindible para programar

A la hora de programar en lenguaje máquina hay que proceder con gran cuidado; es muy fácil, demasiado, bloquear el AMSTRAD y perder el fruto de horas de esfuerzos. Teclear CALL &0 y se hará evidente lo que quiero decir. Un intento en este momento con nuestra vieja conocida CALL &3000 producirá resultados sorprendentes. En efecto, el programa tan cuidadosamente introducido ya no está ahí; murió. Nos vamos a permitir el lujo de sentar cátedra y decir, con la voz de la experiencia: **«los pequeños errores causan los mayores desastres.»** Solución: SALVE EL CODIGO MAQUINA EN CINTA/DISCO ANTES DE EJECUTARLO.

Como se mencionó la semana pasada, existen programas ENSAMBLADORES que permiten teclear, más que la lista de bytes en la memoria, los mnemónicos asociados a ellos, de una forma bastante más comprensible para las personas. *La figura 9* muestra un listado típico de ensamblador dividido en tres columnas: direcciones, códigos de operación *(opcodes)* y mnemónicos.

Esto es todo por esta semana. Continuaremos aprendiendo nuevas y mejores formas de introducir código máquina en el ordenador, junto con más instrucciones del Z80. Hasta pronto.

EGG BLITZ

El desbordamiento de una presa cercana, ha causado el aislamiento de un grupo de exploradores, que se encontraban realizando una acampada en el valle cercano.

Su situación después de tres días incomunicados se ha vuelto desesperada y el hambre se apodera de todos los miembros de la expedición.

Siendo posible rescatar a los scouts, ya que los tres helicópteros de CIVIL PROTECTION se encuentran averiados, y los servicios técnicos todavía tardarán un par de meses en repararlos, **RODOLF** el intrépido piloto del equipo acrobático DOS ALAS MEJOR QUE UNA, decide acudir en su ayuda y arrojarles los alimentos necesarios para asegurar su supervivencia.

Puesto en contacto con el departamento de SINIESTROS Y CATASTROFES VARIAS, y debido a la escasez de alimentos disponibles, carga su avión con huevos de avestruz, restantes del último viaje al Sahara de la sección suministros, los cuales lanzará sobre los hambrientos scouts. Estos al darse cuenta de su llegada *(hay que ver lo que hace el hambre)*, se suben unos encima de otros formando una montaña humana, a la cual en cada pasada de su modelo acrobático, **RODOLF**, lanzará sus famosos huevos.

Cuando un explorador recibe su huevo, se retira de la pirámide y deja que los de abajo reciban el suyo.

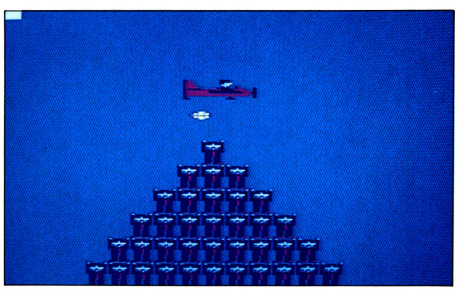
Conseguir abastecer a toda la pirámide es una misión difícil y arriesgada, que en caso de ser cumplida nos reportará una importante bonificación.

El juego está realizado en código máquina con lo que se consigue un movimiento suave del avión, los efectos sonoros están bien conseguidos y la caída de los huevos se realiza punto a punto produciendo un efecto bastante real.

Para los amantes del código máquina indicamos a continuación las direcciones donde comienzan las rutinas principales del juego.

Entre ellas podemos encontrar rutinas de movimiento de sprites, muy interesantes para utilizarlas en nuestros propios programas.

&831D	*Marcador*
&831E	*Dirección de aeroplano*
&8320	*Descenso del avión*
&8321	*Dirección del huevo*
&8323	*Antigua dirección utilizada por la rutina de sprites*
&8325	*Nueva dirección utilizada por la rutina de sprites*
&8327	*Datos utilizados por la rutina de sprites*
&8329	*Número de filas utilizadas por la rutina de sprites*
&832A	*Número de columnas utilizadas por la rutina de sprites*
&832B-&834E	*Data de sonido*
&834F	*Principio del código máquina*
&8364	*Bucle principal de control*
&84A3	*Rutina de sprites*
&83A6	*Movimiento del huevo*
&8463	*Movimiento del avión*



Serie Oro

```
10 REM  ******************
20 REM *                  *
30 REM *   Egg  Blitz     *
40 REM * By R.A.Waddilove *
50 REM *                  *
60 REM  ******************
70 MODE 1:MEMORY &8223
80 GOSUB 1140:REM incializacion
90 GOSUB 500:REM instrucciones
100 MODE 0
110 WHILE 1
120 GOSUB 720:REM presentacion
130 CALL &8364:REM bucle principal
140 IF (PEEK(&8320) AND 128) THEN GOSUB
940:GOSUB 170 ELSE GOSUB 390
150 WEND
160 END
170 REM ===== fin del juego =====
180 FOR i%=0 TO 7000:NEXT
190 MODE 1
200 CALL &BC02:CALL &BB4E:REM reset vdu
210 INK 0,0:BORDER 0
220 PAPER 3:PEN 1
230 LOCATE 18,1
240 PRINT " E G G   B L I T Z "
250 PAPER 0
260 score=10*(PEEK(start) AND &F0)/&10+(
PEEK(start) AND &F)
270 IF best(score THEN best=score
280 LOCATE 5,5
290 PRINT "Los Scouts han recogido";scor
e;"huevos"
300 PEN 2:LOCATE 10,10
310 IF score=best THEN PRINT "Esta es la
 mejor jugada!" ELSE PRINT "El record es
ta en";best
320 PEN 3:LOCATE 6,15
330 PRINT " Pulsa una tecla para empezar
"
340 WHILE INKEY$()"":WEND
350 WHILE INKEY$=""":WEND
360 POKE start,0:level=9
370 MODE 0
380 RETURN
390 REM ===== aterrizaje =====
400 PEN 10:LOCATE 3,10
410 PRINT "Mas dificil Rex !"
420 SOUND 129,239,160,4
430 FOR i%=0 TO 1000:NEXT
440 SOUND 130,95,105,4
450 FOR i%=0 TO 1000:NEXT
460 SOUND 132,40,50,4
470 FOR i%=0 TO 2000:NEXT
480 CLS:level=level+2
490 RETURN
500 REM === instrucciones ===
510 CLS
520 INK 0,6:INK 1,6:INK 3,25
530 BORDER 3
540 PEN 1:LOCATE 1,25
550 PRINT "Egg Blitz"
```

```
560 FOR i%=0 TO 144 STEP 2
570 FOR j%=0 TO 16 STEP 2
580 IF TEST(i%,j%) THEN PLOT 155+i%*2,35
5+j%*2,2:PLOT 155+i%*2,357+j%*2:PLOT 157
+i%*2,355+j%*2:PLOT 157+i%*2,357+j%*2
590 NEXT
600 NEXT
610 LOCATE 1,25:PRINT SPACE$(15);
620 INK 1,21
630 PEN 3:LOCATE 1,6:PRINT "  Tu el gran
 REX, piloto acrobatico,      debes inten
tar soltar los alimentos      sobre un gr
upo de scout."
640 PRINT:PRINT "  Los huevos que tu dej
aras caer deben    impactar encima de los
 scout que estan    formando una piramide
 humana."
650 PRINT:PRINT "  Procura no soltar los
 huevos en medio   de los scout ni en el
suelo pues hasta   que un huevo sea recog
ido o roto no      podras soltar otro."
660 PEN 1:PRINT:PRINT TAB(3)"Pulsa espac
io para soltar los huevos"
670 PEN 2:LOCATE 7,22:PRINT CHR$(24);"Pu
lsa una tecla para empezar"
680 MOVE 0,0:DRAW 0,398,1:DRAW 638,398:D
RAW 638,0:DRAW 0,0:MOVE 0,345:DRAW 640,3
45:MOVE 0,18:DRAW 640,18
690 WHILE INKEY$()"":WEND
700 WHILE INKEY$=""":WEND
710 RETURN
720 REM ==== Presentacion ====
730 BORDER 1
740 DATA 2,6,1,3,16,3,24,15,12,26,25,0,0
,0,0,0
750 RESTORE 740
760 FOR i%=0 TO 15
770 READ j%:INK i%,j%
780 NEXT
790 PAPER 0:LOCATE 1,25
800 PRINT SPACE$(20);
810 PAPER 0
820 PEN 9:LOCATE 5,1
830 PRINT "Puntos: "
840 CALL &834F,&C850,pdata,&D8C:REM pint
a el avion
850 address=&C6F8
860 FOR y%=level TO 1 STEP -2
870 FOR x%=9-level TO 8-level+y%
880 CALL &834F,address+x%*4,mdata,&1004:
REM pinta scout
890 NEXT
900 address=address-&A8+4
910 NEXT
920 FOR i%=0 TO 1000:NEXT
930 RETURN
940 REM ===== choque =====
950 SOUND 130,0,0,0
960 FOR i%=1 TO 10
970 LOCATE 1,1:PRINT CHR$(11)
980 INK 1,INT(RND*27)
990 SOUND 4,RND*1000,5,6,0,0,1
1000 FOR j%=1 TO 100:NEXT
1010 LOCATE 1,25:PRINT CHR$(10)
1020 SOUND 4,RND*1000,5,6,0,0,1
1030 FOR j%=1 TO 100:NEXT
1040 NEXT
1050 INK 1,6
1060 PAPER 8:LOCATE 1,25
1070 PRINT SPACE$(20);
1080 PAPER 0
1090 SOUND 1,50,300,5,0,1
1100 FOR i%=0 TO 3000:NEXT
1110 PEN 10:LOCATE 8,3
1120 PRINT "Oh no!":PRINT:PRINT TAB(5)"H
as chocado":PRINT:PRINT TAB(5)"con un sc
out!"
1130 RETURN
1140 REM === inicializacion ===
1150 PRINT "Cargando codigo maquina..."
1160 pdata=HIMEM+1:mdata=pdata+156
1170 edata=HIMEM+221:start=HIMEM+250
1180 ENT 1,150,1,2
1190 ENT -2,10,5,1,10,-5,1
1200 ENT 3,200,1,5
1210 ENT 4,50,-4,1
1220 RESTORE 1400
1230 checksum=0
1240 FOR i%=1 TO 236
1250 READ j%:POKE (HIMEM+i%),j%
1260 checksum=checksum+j%
1270 NEXT
1280 IF checksum()21768 THEN PRINT "Erro
r en datas de los caracteres":END
1290 RESTORE 1400
1300 checksum=0
1310 FOR i=&6310 TO &6508
1320 READ a$:POKE i,VAL("&"+a$)
1330 checksum=checksum+VAL("&"+a$)
1340 NEXT
1350 IF checksum()42197 THEN PRINT "Erro
r codigo maquina":PRINT "Por favor, revi
sa las datas del codigo maquina":END
1360 POKE start,0
1370 level=9:best=8
1380 RETURN
1390 REM ===== avion =====
1400 DATA 192,0,0,0,0,0,240,168,0,0,0,0,
192,128,0,0,0,0,240,0,0,0,0,0,192,128,0,
0,0,0,176,48,0,0,0,0,192,192,0,0,0,0,48,
32,0,0,0,0,192
1410 DATA 192,0,0,0,0,16,0,64,0,0,0,0,192,
192,128,0,0,64,204,136,64,128,0,4,132,12
,72,192,192,192,204,204,192,192,128,4,19
2,192,192,12,12,12,12
1420 DATA 12,192,192,4,192,192,192,192,1
2,12,12,12,12,72,192,12,192,192,192,192,
192,192,192,192,192,192,4,192,192,19
2,192,192,192,192,192,192,128,4,8,8,
0,0,0,0,0,4,0,0,0,4,12,0,0,0,0,0,0,12,8,
0,0,0
1430 REM ===== hombre =====
1440 DATA 32,0,0,32,136,240,160,136,216,
24,88,136,152,48,48,136,136,36,32,136,13
6,16,8,136,204,196,196,136,204,200,204,1
36,68,200,204,0,68,200,204,0,68,200,204,
0,68,136,204,0,68,136,204,0,68,136,204,0
,68,136,204,0,192,128,192,128
1450 REM ===== huevo =====
1460 DATA 0,188,168,20,188,188,252,252,2
52,20,188,188,0,188,168,136
1470 REM === codigo maquina ===
1480 DATA 00,00,00,00,00,00,00,00
1490 DATA 00,00,00,00,00,00,01,00
1500 DATA 02,E8,03,00,04,32,00,02
1510 DATA 00,03,0A,00,00,04,E8,03
1520 DATA 02,00,04,2C,01,00,05,32
1530 DATA 00,02,00,00,D0,07,00,05
1540 DATA 0A,00,DD,4E,00,DD,46,01
1550 DATA DD,5E,02,DD,56,03,DD,6E
1560 DATA 04,DD,66,05,C3,E6,04,CD
1570 DATA 06,89,F5,21,50,C0,22,1E
1580 DATA 83,21,00,00,22,21,83,3E
1590 DATA 01,32,28,83,CD,63,04,11
1600 DATA 00,00,21,00,00,CD,10,BD
1610 DATA CD,A6,83,CD,0D,BD,01,1B
1620 DATA 00,AF,ED,42,38,F5,3A,20
1630 DATA 83,E6,C0,20,07,3E,42,CD
1640 DATA 1E,BB,28,D8,F1,CD,0C,B9
1650 DATA C9,2A,21,83,7C,B5,20,28
1660 DATA 3A,20,83,E6,01,32,20,83
1670 DATA C8,3E,2F,CD,1E,BB,C8,2A
1680 DATA 1E,83,01,54,38,89,22,21
1690 DATA 83,11,00,83,01,03,05,CD
1700 DATA E6,84,21,34,83,C3,AA,BC
1710 DATA 22,23,83,01,03,38,89,30
1720 DATA 04,01,58,C8,09,3A,C8,82
1730 DATA BE,20,06,2B,2B,2B,BE,28
1740 DATA 2A,3E,83,BE,28,1D,2A,23
1750 DATA 83,01,08,18,09,30,04,01
1760 DATA 58,C8,09,22,25,83,22,21
1770 DATA 83,11,00,83,01,05,03,CD
1780 DATA A3,84,C9,21,46,83,CD,AA
1790 DATA BC,18,38,E5,21,3D,83,CD
1800 DATA AA,BC,21,01,0C,CD,75,BB
1810 DATA 3A,1D,83,C6,01,27,32,1D
1820 DATA 83,F5,E6,F8,CB,3F,CB,3F
1830 DATA CB,3F,CB,3F,C6,30,CD,5D
1840 DATA BB,F1,E6,0F,C6,30,CD,5D
1850 DATA BB,E1,11,C8,02,01,04,10
1860 DATA CD,E6,84,11,00,83,01,03
1870 DATA 05,2A,21,83,CD,E6,84,21
1880 DATA 00,00,22,21,83,C9,2A,1E
1890 DATA 83,01,F8,C6,AF,ED,42,38
1900 DATA 08,3A,20,83,F6,40,32,20
1910 DATA 83,21,2B,83,CD,AA,BC,2A
1920 DATA 1E,83,22,23,83,23,22,25
1930 DATA 83,22,1E,83,11,24,82,01
1940 DATA 0D,8C,CD,A3,84,2A,1E,83
1950 DATA 01,5B,20,09,3E,08,BE,C8
1960 DATA 21,20,83,CB,FE,C9,ED,43
1970 DATA 29,83,ED,53,27,83,0E,02
1980 DATA CD,19,BD,F3,ED,5B,27,83
1990 DATA 3A,29,83,47,C5,3A,2A,83
2000 DATA 47,2A,23,83,1A,AE,77,23
2010 DATA 13,10,F9,2A,23,83,01,80
2020 DATA 08,09,30,04,01,50,C0,09
2030 DATA 22,23,83,C1,10,DE,2A,25
2040 DATA 83,22,23,83,0D,20,CD,FB
2050 DATA C9,22,23,83,C5,41,2A,23
2060 DATA 83,1A,AE,77,23,13,10,F9
2070 DATA 2A,23,83,01,00,08,09,30
2080 DATA 04,01,50,C8,09,22,23,83
2090 DATA C1,10,E1,C9,00,00,00,00
```

GANA UN AMSTRAD CPC664 PARTICIPANDO EN NUESTRA ENCUESTA

M. H. AMSTRAD, *para acercar más y más la revista a los gustos y preferencias de nuestros lectores, plantea la siguiente encuesta que estamos seguros ayudará a hacer una revista abierta a todo tipo de tendencias dentro del mundo de la informática.*
Entre todas las cartas recibidas, sorteamos un ***AMSTRAD CPC664*** *y 4 unidades de disco.*

Rellenad la encuesta que a continuación os adjuntamos colocando una X en la casilla apropiada y enviadla a:

HOBBY PRESS, S.A.
AMSTRAD SEMANAL

Apartado de Correos 54.062
28080 Madrid

Nombre y apellidos ..... **Edad** .....
Domicilio .....
Localidad ..... **C. Postal** .....
Provincia ..... **Teléfono** .....

Modelo de AMSTRAD ..... CPC464 ☐ CPC664 ☐
¿Para qué lo usas? ..... Juegos ☐ Gestión ☐ Otros ☐
¿Te interesa la programación? ..... Sí ☐ No ☐
Lenguajes que utilizas ..... Basic ☐ Cód. M. ☐ Pascal ☐ Logo ☐
¿Te gustaría aprender nuevos lenguajes? ..... Sí ☐ No ☐
¿Cuáles? ..... Basic ☐ Cód. M. ☐ Pascal ☐ Logo ☐
Forth ☐ C ☐

Programas

¿Sueles teclear los programas de las revistas? ..... Casi todos ☐ 1 de 2 ☐ 1 de 4 ☐ 1 de 8 ☐ Ninguno ☐
¿Qué tipo de programas te interesan? ..... Juegos ☐ Utilidades ☐
Juegos ..... Marcianos ☐ Deportivos ☐ Aventuras animadas ☐
Estrategia ☐ Inteligencia ☐ Aventuras de texto ☐
Utilidades ..... Procesar textos ☐ Bases de datos ☐ Hojal calc. ☐
Lenguajes ☐ Gestión comercial ☐ Gráficos ☐

¿Te gusta que se comenten juegos en tu revista?
¿Cuántas páginas/semana? ..... Ninguna ☐ 1 ☐ 3 ☐ 5 ☐ Más ☐
¿Qué opinas de los artículos de Cód. M.? ¿Te interesan?
¿Cuántas páginas/semana? ..... Ninguna ☐ 1 ☐ 3 ☐ 5 ☐ Más ☐
¿Te gustaría que se hablase del ordenador y sus periféricos? ¿Cuántas páginas/semana? ..... Ninguna ☐ 1 ☐ 3 ☐ 5 ☐ Más ☐
¿Te interesaría una sección dedicada a principiantes en Basic Amstrad? ¿Cuántas páginas/semana? ..... Ninguna ☐ 1 ☐ 3 ☐ 5 ☐ Más ☐
¿Te interesaría una sección dedicada a gráficos y sonido en el Amstrad? ¿Cuántas páginas/semana? ..... Ninguna ☐ 1 ☐ 3 ☐ 5 ☐ Más ☐
¿Te gustarían artículos acerca de programas de aplicación comerciales? ¿Cuántas páginas/semana? ... Ninguna ☐ 1 ☐ 3 ☐ 5 ☐ Más ☐
¿Qué secciones añadirías a la revista? .....
¿Qué secciones quitarías? .....

Sin duda alguna

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

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

SIN DUDA ALGUNA está abierta a todos.

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

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

¡Os esperamos!

GANA 100.000 PESETAS CON MICROHOBBY

AMSTRAD SEMANAL

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

Las condiciones son sencillas:

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

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

— *Descripción general del programa.*

— *Tabla de subrutinas y variables utilizadas, explicando claramente la función de cada una de ellas.*

— *Instrucciones de manejo.*

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

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

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

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

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

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

— Los programas sospechosos de plagio serán eliminados inmediatamente.

¡ENVIANOS TU PROGRAMA!

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

Mercado común

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

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

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

MICROHOBBY AMSTRAD SEMANAL

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

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

Envíanos con la menor demora posible, el cupón correspondiente.

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.

Una Gran Noticia para los Usuarios de AMSTRAD

A partir del próximo septiembre estará en vuestra tienda de informática, en los quioscos de prensa o —si preferís suscribiros— en vuestro domicilio, la revista AMSTRAD USER. Una publicación mensual, repleta de información, con abundantes listados, trucos de programación, crítica de software y periféricos, noticias y novedades, concursos, etcétera. Para estar al día. Para sacarle aún más partido a tu AMSTRAD.

Para más información:

AMSTRAD

ESPAÑA

Dpto. Publicaciones Avd. del Mediterráneo, 9 28007 Madrid

CPC-464 COLOR

NUEVO CPC-664 COLOR

AMSTRAD 464/664

El increíble ORDENADOR PERSONAL

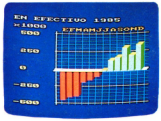
Si en la primavera de 1984 AMSTRAD conmocionó al mundo informático con el modelo CPC 464, la aparición ahora de CPC 664 -en el que el magnetófono ha sido sustituido por una unidad de disco de 3" (180 K) incorporada- vuelve a despertar el entusiasmo de especialistas y público. El éxito arrollador de ambos modelos encuentra su explicación en la filosofía de diseño de AMSTRAD. Una filosofía que ofrece:

Un sistema completo que incluye la unidad central, el monitor y el magnetófono o la unidad de disco. Un equipo compacto, listo para funcionar sin cableados engorrosos ni necesiad de adquirir más periféricos. Sólo requiere desembalarlo y enchufar un cable -un solo cable- a la red.

Con un paquete de **programas de obsequio** y, además, el Sistema Operativo CP/M y el lenguaje LOGO incluídos en el suministro del CPC 664.

Unas prestaciones del más alto nivel, con 64 K de memoria RAM, 32 K de memoria ROM, con resolución de 640 x 200 puntos, 27 colores, 20, 40 u 80 columnas de texto en pantalla, 8 "ventanas" de trabajo, teclado profesional con 32 teclas programables, sonido estéreo con 3 canales y 8 octavas por canal. Y un BASIC super-ampliado y dotado incluso de comando de control del microprocesador (Every, After...).

Una tecnología contrastada y fiable basada en el popular microprocesador Z80A y en una electrónica depurada y con un riguroso control de calidad.



Una extensa biblioteca de programas que se incrementa literalmente día a día y que ya dispone de centenares de títulos para todos los gustos y necesidades: gestión profesional (Contabilidad, Control de Stocks, Bases de Datos, Hojas de Cálculo, Procesadores de Texto,...), educación, lenguajes, y ayuda a la programación (Ensamblador, Desensamblador, Pascal, Forth, Logo, Diseñador de Gráficos, Diseñador de Sprites...), de toma de decisiones (Proyect Planner, Desición Maker,...) juegos de habilidad (La Pulga, Manic Miner, Decathtion, Android,...) juegos de inteligencia (Ajedréz, Backgamon,...), juegos de estrategia (Batalla de Midway, II Guerra Mundial,...), juegos de aventuras (Hobbit, Sherlock Holmes,...) juegos de simulación (sumulador de Vuelo, Tenis, Billar, Mundial de Fútbol,...).

Una asistencia técnica rápida y eficaz que **AMSTRAD ESPAÑA** garantiza **exclusivamente** a los equipos adquiridos a través de su Red Oficial de Distribuidores y acompañados de la **Tarjeta de Garantía de AMSTRAD ESPAÑA.**

Unos precios increíbles que no admiten comparación con los de cualquier otro ordenador personal de sus características.

★Ordenador CPC 464, con monitor y magnetófono incorporado. Manual del Usuario y obsequio del Libro "Guía de Referencia del Programador" y de 8 programas:

Con Monitor de fósforo verde(12"). . . . **64.900 pts.**
Con Monitor color(14"). . . . . **93.900 pts.**

★Ordenador CPC 664, con Unidad de Disco incorporada, Manual del Usuario, incluyendo Sistema Operativo CP/M, Lenguaje Logo y **obsequio de cinco programas (Base de Datos, Proceso de Textos, Diseñador de Gráficos, Random Files, Puzzle y Animal, Vegetal, Mineral.**

Con Monitor de fósforo verde(12"). . . . **109.500 pts.**
Con Monitor color(14"). . . . . **134.500 pts.**

AMSTRAD

ESPAÑA

Avd. de Mediterráneo, 9, 28007 MADRID. Tels. 433 45 48 - 433 48 76

Delegación Cataluña: C/. Tarragona, 100, 08015 BARCELONA - Tel. 325 10 58

NOTA: Es muy importante verificar la garantía del aparato ya que sólo **AMSTRAD ESPAÑA** puede garantizarle la adecuada reparación y sobre todo materiales de repuesto oficiales (Monitor, ordenador, cassette o unidad de discos).