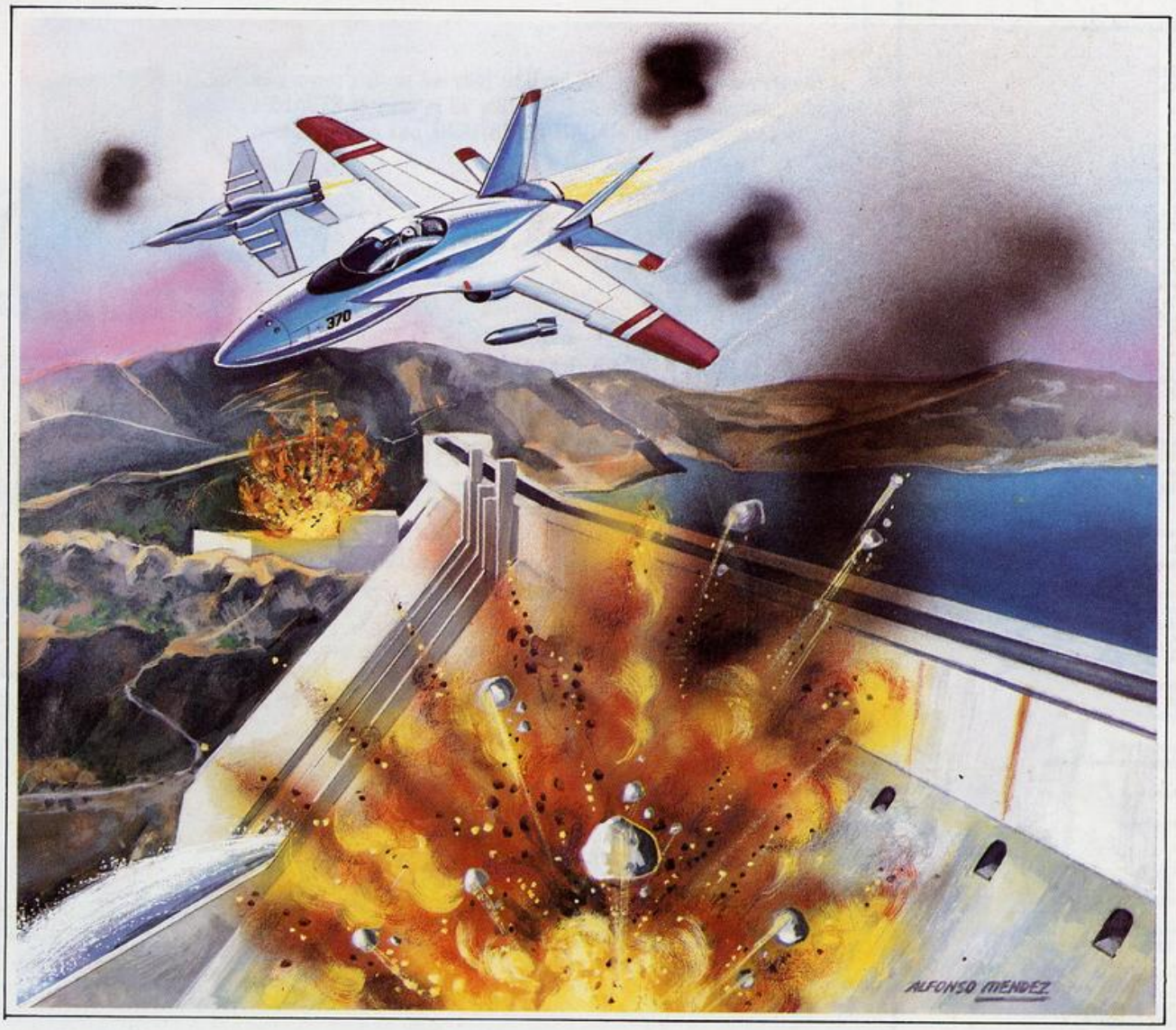
31
150pts.

RUN

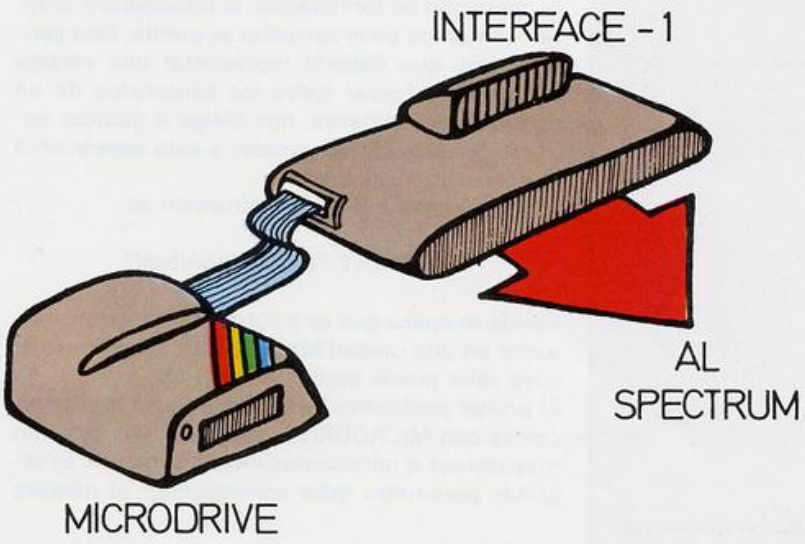
Enciclopedia Práctica del Spectrum



Nueva Lente/Ingelek

EL MICRODRIVE

EL ZX MICRODRIVE es el sistema de almacenamiento de datos de acceso rápido de mayor difusión en el mercado. Ello es debido a dos razones fundamentales: su bajo costo en relación a su capacidad y prestaciones, y su facilidad de manejo.
Para poder conectar a nuestro Spectrum unidades MICRODRIVE es preciso adquirir el correspondiente interface denominado ZX INTERFACE 1. Este interface, uno de los dos comercializados por *Sinclair Research Ltd.*, podemos denominarlo como de «gestión» puesto que, además de permitir acoplar al ordenador hasta 8 unidades MICRODRIVE, soporta la conexión a impresoras y otros periféricos del estándar RS232, así como el establecimiento de una red *(NET)* de comunicación de hasta 64 Spectrums.
El MICRODRIVE permite los mismos tipos de almacenamiento de datos del casete: programas, bloques de memoria, pantallas y matrices; pero incluye además una nueva forma denominada archivo secuencial. Asimismo incorpora la posibilidad de ejecución automática de un programa determinado, la de proteger programas, y la de efectuar copias de ficheros.



Para poder conectar a nuestro Spectrum unidades MICRODRIVE, es necesario adquirir un INTERFACE 1.

El MICRODRIVE permite almacenar: programas, bloques de memoria, pantallas, matrices y además archivos secuenciales.

FORMAT

Para que un cartucho de MICRODRIVE virgen sea utilizable es preciso someterlo a la operación de formateado. Esta operación consiste en la exploración de la cinta sin fin que constituye el cartucho con dos objetivos fundamentales: dividirlo en sectores, los cuales albergarán bloques de datos, y realizar al mismo tiempo la comprobación de que estas porciones son aptas para las operaciones de lectura/escritura, siendo eliminadas en caso contrario.
Es por este motivo que el espacio disponible en el cartucho, al concluir la operación de formateado, no es siempre el mismo. En condiciones normales este espacio debe oscilar entre los 85 y 95 Kbytes. De no ser así, es síntoma de deterioro del

¡!

Para que un cartucho de MICRODRIVE virgen sea utilizable es preciso formatearlo. Esta operación consiste en explorar el cartucho para dividirlo en sectores, y realizar la comprobación de que son aptos para las operaciones de lectura/escritura.

¡!

Un síntoma de deterioro de un cartucho es el aumento considerable en el tiempo de carga y grabación.

*

El INTERFACE 1 permite acoplar hasta 8 unidades MICRODRIVE, impresoras y otros periféricos del estándar RS232, así como el establecimiento de una red *(NET)* de comunicación de hasta 64 Spectrums.

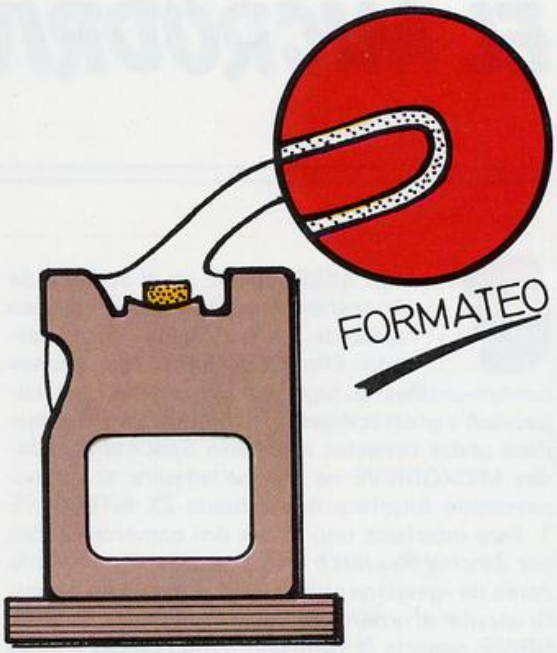
*

El MICRODRIVE permite almacenar en cartucho programas, bloques de memoria, pantallas, matrices de datos y archivos secuenciales.

cartucho, lo cual debe hacernos pensar en reponerlo en un futuro no muy lejano en evitación de males mayores.
El ordenador se toma unos 30 segundos en realizar esta operación. Durante este tiempo, el contorno primero parpadea y más tarde es recorrido verticalmente por unas bandas anchas de color oscuro, presentando finalmente el mensaje: *O. K.*
Debemos tener en cuenta que cuando se realiza la operación de formateado, la información anterior contenida en el cartucho se pierde. Esta contingencia, que debería representar una ventaja por permitir borrar todos los contenidos de un cartucho rápidamente, nos obliga a guardar extremo cuidado en no someter a esta operación a un cartucho inadecuado.
El formato general de la instrucción es:

FORMAT "m";n;"nombre"

Donde **m** indica que se trata de formatear un cartucho en una unidad MICRODRIVE de número n, cuyo valor puede oscilar entre 1 y 8.
El primer parámetro es fijo para todas las operaciones con MICRODRIVE, pudiendo escribirse en mayúsculas o minúsculas indistintamente. El segundo parámetro debe corresponder al número



Para que un cartucho de MICRODRIVE sea utilizable es necesario formatearlo, es decir, explorar la cinta sin fin.

Un síntoma del deterioro de los cartuchos es el aumento considerable en el tiempo de carga y grabación.

de la unidad en la que deseamos efectuar la operación. Caso de existir una única unidad debe especificarse siempre un 1, de existir más, irán conectadas una a la otra hacia la izquierda de la primera, siendo su numeración correlativa desde 2 hasta 8. El tercer parámetro contendrá el nombre (1 a 10 caracteres) que deseamos dar al cartucho.
Los tres parámetros referenciados son de uso obligatorio. En caso de olvidarnos del primero o de especificarlo incorrectamente, obtendremos el mensaje de error *Invalid device expression* (Identificación de periférico incorrecta). Asimismo, si especificamos como segundo parámetro el número 0 o uno superior a 8 obtendremos el mensaje *Invalid drive number* (número de MICRODRIVE incorrecto). Por último, si señalamos un nombre no válido para el cartucho (una cadena vacía o excediendo 10 caracteres), aparecerá el mensaje *Invalid name* (nombre incorrecto).
Las sucesivas operaciones de borrado y escritura de archivos contenidos en los cartuchos va deteriorando éstos de forma progresiva. Un síntoma de que este hecho se está produciendo es el aumento considerable en el tiempo de carga y grabación. Una solución para evitar este problema es copiar el contenido del cartucho deteriorado sobre un segundo, realizando posteriormente un nuevo formateado del primero. Los cartuchos así

regenerados, cobran nueva vida, obteniendo un mayor rendimiento en lo sucesivo hasta que vuelvan a deteriorarse por el uso.

CAT

La sentencia **CAT** permite obtener un listado por pantalla, ordenado alfabéticamente, del contenido de un cartucho. Su formato general es:

CAT n

Donde **n** es obligatorio y debe ser un número de 1 a 8 especificando el número de unidad MICRODRIVE en la cual trabajar. La especificación incorrecta de este número provoca el error *Invalid drive number.*
CAT suele emplearse inmediatamente después de efectuar un **FORMAT,** para cerciorarnos de que la capacidad del cartucho es la adecuada y su nombre el correcto.

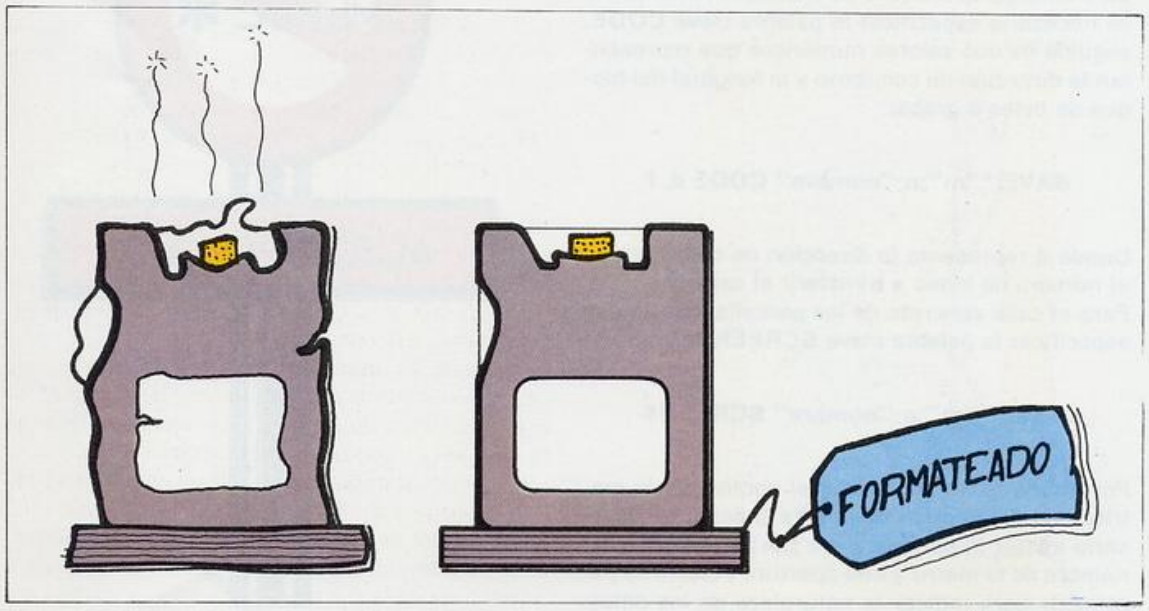
La sentencia CAT permite obtener un listado por pantalla, ordenado alfabéticamente, del contenido de un cartucho.

SAVE *

La sentencia **SAVE** * es la equivalente en MICRODRIVE de la sentencia **SAVE** para el casete, permitiendo almacenar en cartucho programas, bloques de memoria, pantallas y matrices de datos. El formato general de la sentencia es:

SAVE *''m'';n;''nombre''

Si deseamos regenerar un cartucho basta con trasladar la información a otro cartucho, y realizar un formateo al mismo.



¡!

La sentencia **CAT** permite obtener un listado por pantalla, ordenado alfabéticamente, del contenido de un cartucho.

*

El MICRODRIVE es el sistema de almacenamiento de datos de acceso rápido de mayor difusión debido a su bajo costo en relación a su capacidad y prestaciones.

*

Para poder conectar a nuestro Spectrum unidades MICRODRIVE es preciso adquirir el ZX INTERFACE 1.

La sentencia SAVE * *es el equivalente en MICRODRIVE de la sentencia* SAVE *para casete.*

¡!

Al realizar la operación de formateado, la información anterior contenida en el cartucho se pierde.

*

El MICRODRIVE permite la invisibilidad de los nombres de programas y datos en el directorio (**CAT**) del MICRODRIVE añadiendo a éstos el prefijo **CHR$ 0**.

*

En condiciones normales, la capacidad de un cartucho oscila entre los 85 y 95 Kbytes. De no ser así, es síntoma de deterioro del cartucho.

Dicho formato permite almacenar programas. Para conseguir la autoejecución, al igual que sucede con la sentencia **SAVE**, es necesario especificar la palabra clave **LINE**, seguida del número de línea a partir de la cual se desea que comience la ejecución del programa, al cargarlo la próxima vez en memoria:

SAVE *''m'';n;''nombre'' LINE l

Donde **l** representa el número de línea desde el que se desea comience la ejecución.
En la autoejecución, cosa que no ocurre con el casete, no está permitida la operación de **MERGE *** del programa así grabado en cartucho. En cierto modo, esta particularidad nos brinda un método sencillo de efectuar la protección de programas almacenados en cartucho contra la obtención de copias ilegales.
De forma similar a lo que sucede con el casete, para almacenar bloques de memoria en cartucho es necesario especificar la palabra clave **CODE**, seguida de dos valores numéricos que representan la dirección de comienzo y la longitud del bloque de bytes a grabar:

SAVE *''m'';n;''nombre'' CODE d, l

Donde **d** representa la dirección de comienzo y **l** el número de bytes a transferir al cartucho.
Para el caso concreto de las pantallas, basta con especificar la palabra clave **SCREEN$**:

SAVE *''m'';n;''nombre'' SCREEN$

Por último, para almacenar el contenido de matrices de datos numéricas o de cadena, es necesario incluir la palabra clave **DATA**, seguida del nombre de la matriz y una apertura y cierre de paréntesis para indicar la naturaleza de los datos:

SAVE *''m'';n;''nombre'' DATA x( )

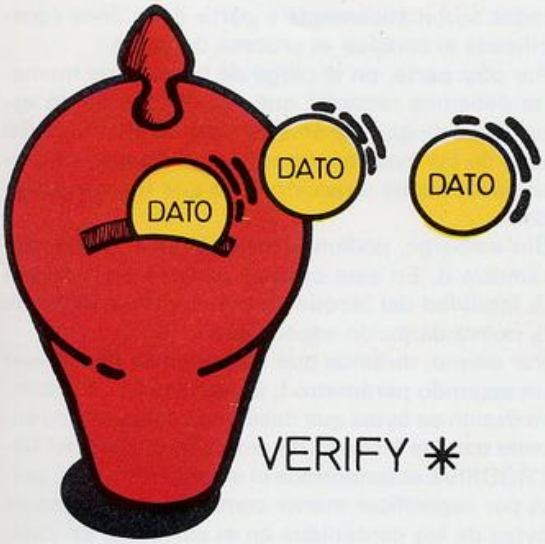
SAVE *''m'';n;''nombre'' DATA x$( )

Donde x y x$ representan los nombres de una matriz cualquiera, numérica o de cadena respectivamente.
Al igual que sucede con la sentencia **FORMAT**, es obligatorio especificar todos los parámetros, produciéndose en caso contrario alguno de los correspondientes mensajes de error.

VERIFY *

La sentencia **VERIFY *** es también equivalente en MICRODRIVE a la sentencia **VERIFY** del ca-

En la autoejecución no está permitida la operación de MERGE * *del programa en cartucho así grabado.*



La sentencia **VERIFY *** *tiene por objeto darnos la seguridad de que el almacenamiento en cartucho se ha realizado sin problemas.*

sete, siendo aplicable a programas, bloques de memoria, pantallas y matrices de datos almacenados previamente en cartucho por medio de **SAVE ***.

La verificación tiene por objeto darnos la seguridad de que el almacenamiento en cartucho se ha realizado sin problemas. Por ello, y dada la alta velocidad a que el MICRODRIVE permite realizar esta operación, es siempre conveniente pasar por este punto inmediatamente después de cada **SAVE ***. Los formatos que esta sentencia puede adoptar son, respectivamente:

VERIFY *''m'';n;''nombre''
VERIFY *''m'';n;''nombre'' CODE d
VERIFY *''m'';n;''nombre'' SCREEN$
VERIFY *''m'';n;''nombre'' DATA x( )
VERIFY *''m'';n;''nombre'' DATA x$( )

Donde los parámetros deben seguir las reglas anteriormente comentadas.

A diferencia de lo que sucede con el casete, la sentencia **VERIFY *** no comprueba byte a byte el contenido de la memoria del ordenador con el del cartucho, sino que se limita a constatar que el nombre solicitado se encuentra grabado y que sus diferentes bloques de datos se hallan enlazados correctamente.

En el **MICRODRIVE**, a diferencia de en el casete, sí es posible la verificación de pantallas.

LOAD *

La sentencia **LOAD *** es también equivalente en MICRODRIVE a la sentencia **LOAD** del casete, permitiendo la lectura de programas, bloques de memoria, pantallas y matrices de datos almacenados con **SAVE ***. Los formatos que esta sentencia puede adoptar son, respectivamente:

LOAD *''m'';n;''nombre''
LOAD *''m'';n;''nombre'' CODE d
LOAD *''m'';n;''nombre'' SCREEN$
LOAD *''m'';n;''nombre'' DATA x( )
LOAD *''m'';n;''nombre'' DATA x$

Donde los parámetros deben seguir las reglas anteriormente comentadas. Dado que el nombre del

No se permiten ni la verificación ni la carga especificando una cadena vacía.

¡!

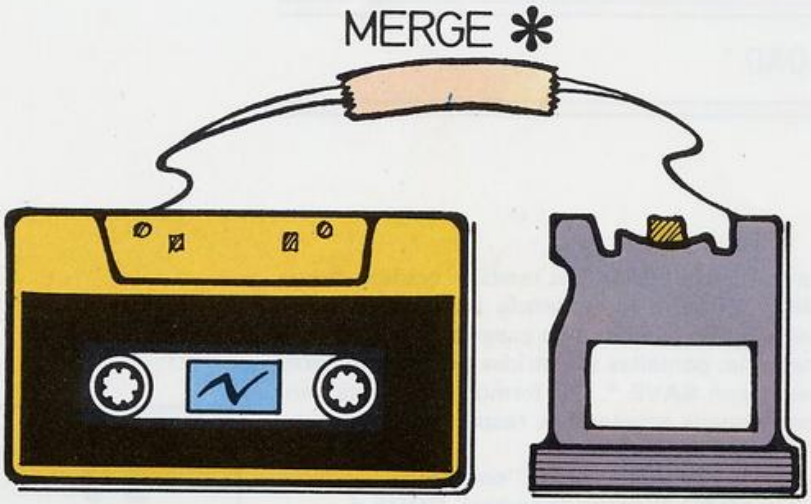
En la autoejecución, no está permitido el **MERGE *** del programa en cartucho así grabado.

*

Las sucesivas operaciones de borrado y escritura de archivos contenidos en los cartuchos van deteriorando éstos de forma progresiva.

*

La sentencia **SAVE *** permite almacenar en cartucho programas, bloques de memoria, pantallas y matrices de datos.



La sentencia **MERGE** * *permite la fusión en memoria del programa contenido en casete, con el procedente de cartucho.*

MERGE *

La sentencia **MERGE** * es equivalente en MICRODRIVE al **MERGE** de casete, permitiendo la fusión en memoria del programa contenido en ésta con uno procedente de cartucho.

En esta operación se tiene en cuenta tanto el área de programa propiamente dicha (líneas que lo archivo es siempre imprescindible, no se permiten ni la verificación ni la carga especificando una cadena vacía (equivalente a **VERIFY** "" y **LOAD** "").

Recordemos que los programas almacenados en cartucho con el parámetro **LINE x**, serán ejecutados automáticamente a partir de la línea especificada al concluir el proceso de carga.

Por otra parte, en la carga de bloques de memoria debemos recordar que, en el caso de no especificar ningún parámetro, ésta se efectuará del total de los bytes contenidos en el bloque y a partir de la misma dirección de la que fueron grabados.

Sin embargo, podemos especificar el primer parámetro **d.** En este caso se cargará en memoria la totalidad del bloque de bytes, pero a partir de la nueva dirección especificada.

Por último, diremos que no podemos especificar un segundo parámetro **l,** determinando el número exacto de bytes que deseamos cargar como sucede con las lecturas de cinta. En el caso del MICRODRIVE obtendremos el error *CODE error,* tanto por especificar menor como mayor número de bytes de los contenidos en el cartucho. En cualquier caso, en la carga desde cinta con un parámetro **l** menor que el original, se efectúa la operación pero con la obtención del error *Tape loading error.*

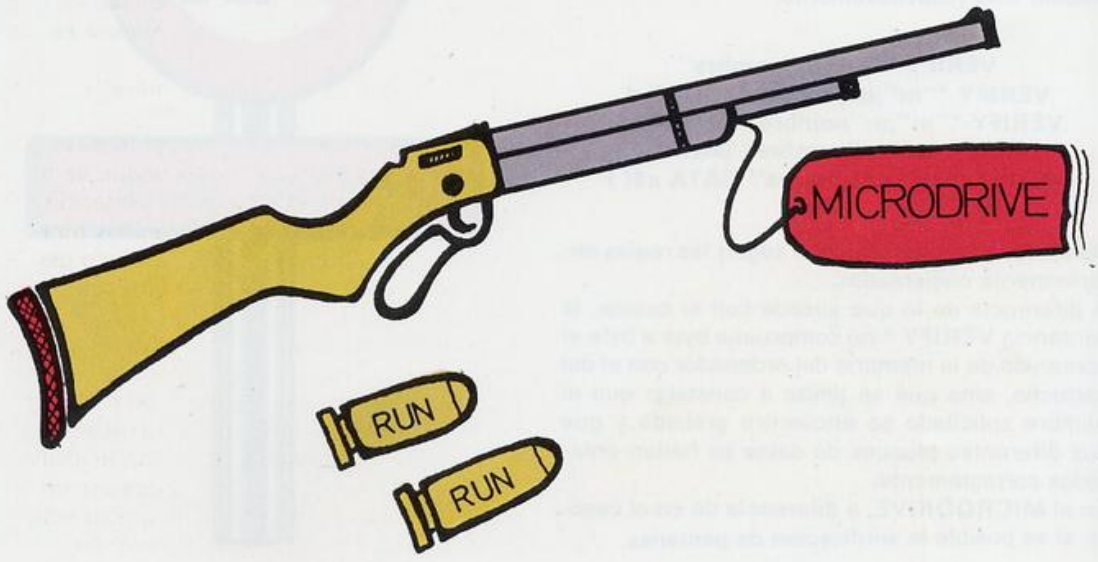
¡!

La verificación tiene por objeto darnos la seguridad ,de que el almacenamiento en cartucho se ha realizado sin problemas.

*

VERIFY * no comprueba byte a byte el contenido de la memoria del ordenador con el del cartucho, sino que se limita a constatar que el nombre solicitado se encuentra grabado y que sus diferentes bloques de datos se hallan enlazados correctamente.

El MICRODRIVE permite la carga automática de un programa contenido en cartucho con el nombre de **run.**



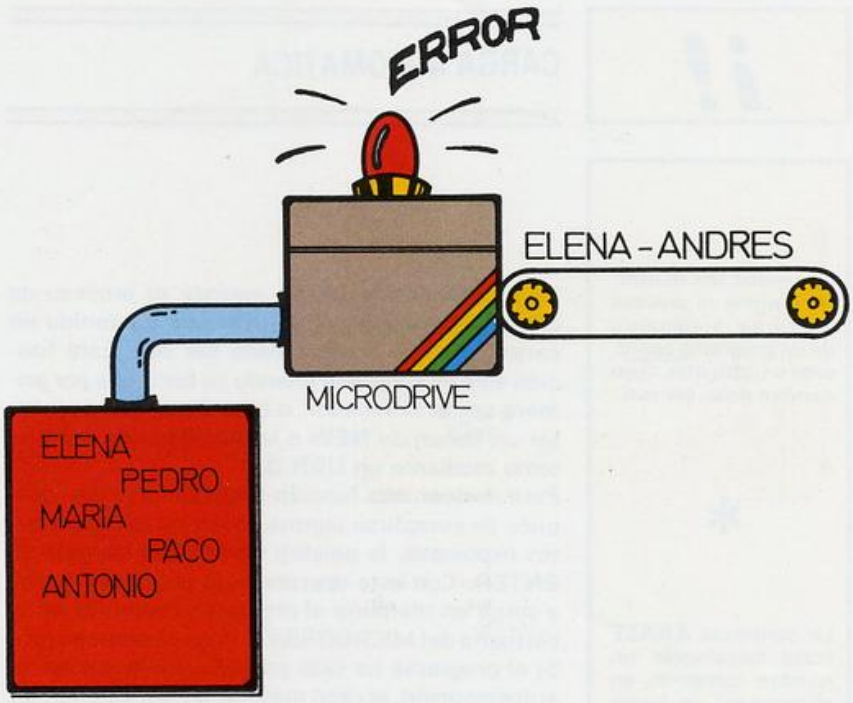
componen), como la correspondiente zona de variables; de forma que los nuevos números de línea reemplazan a los anteriores y son incluidos en el programa si no existían, aplicándose esta misma regla al contenido de las variables. El formato general de la sentencia es:

MERGE *''m'';n;''nombre''

Recordemos aquí que los programas almacenados en cartucho por medio de la sentencia **SAVE * LINE** no admiten la opción de **MERGE**, permitiendo únicamente la de **LOAD *** (carga directa con autoejecución).

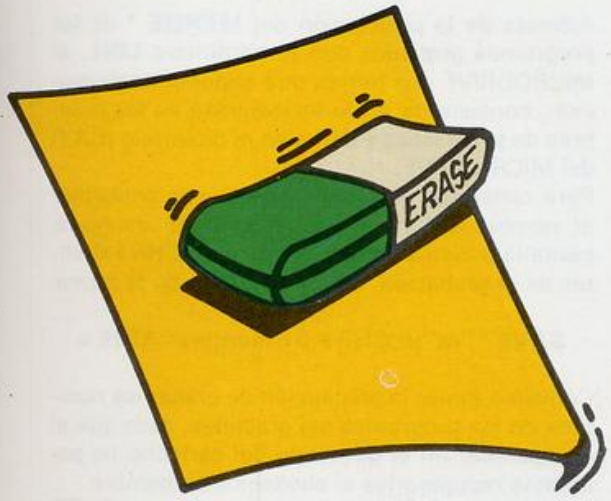
ERASE

Hasta ahora hemos tratado sobre los equivalentes a las funciones de grabación, verificación y lectura de programas, bloques de memoria, pantallas y matrices de datos; siendo característica general el empleo del mismo comando BASIC que para las operaciones en cinta, seguido de un asterisco (*).



El MICRODRIVE no permite nombres duplicados y avisa con un mensaje de error cuando se trata de grabar con un nombre ya existente.

La sentencia **ERASE** *permite borrar físicamente un nombre contenido en cartucho.*



Sin embargo, al igual que **FORMAT** y **CAT** son sentencias específicas de este periférico, **ERASE** también lo es. La sentencia **ERASE** permite borrar físicamente un nombre contenido en el cartucho, de forma que el espacio ocupado por el mismo se suma al total del disponible en la unidad. El formato general de la sentencia es:

ERASE *''m'';n;''nombre''

Donde no es necesario especificar la clase de datos de que se trata: programa, bloque de memoria, pantalla o matriz de datos. La sentencia **ERASE** sirve para eliminar los contenidos innecesarios en el cartucho, pero es también de uso obligado en las sucesivas actualizaciones de los contenidos.
Si, por ejemplo, hemos cargado un programa en memoria para modificarlo, hemos de efectuar un **ERASE** de la copia almacenada en cartucho antes de ejecutar el **SAVE *** de la nueva versión del programa. Esto se debe a que el MICRODRIVE no permite nombres duplicados, y avisa con un mensaje de error cuando se trata de grabar sobre un nombre existente: *Writing to a 'read' file.*

¡!

Los programas almacenados en cartucho con el parámetro **LINE**, son ejecutados automáticamente a partir de la línea especificada al concluir el proceso de carga.

*

La sentencia **VERIFY *** es aplicable a programas, bloques de memoria, pantallas y matrices de datos almacenados previamente en cartucho por medio de **SAVE ***.

¡!

La unidad MICRODRIVE permite el proceso de carga automática de un programa contenido en cartucho, cuyo nombre debe ser **run**.

*

La sentencia **ERASE** borra físicamente un nombre contenido en el cartucho, de forma que el espacio ocupado por el mismo se suma al total del disponible en la unidad.

*

La sentencia **MERGE** * permite la fusión en memoria del programa contenido en ésta con uno procedente de cartucho.

*

La sentencia **LOAD** * permite la lectura de programas, bloques de memoria, pantallas y matrices de datos almacenados con **SAVE** *.

CARGA AUTOMATICA

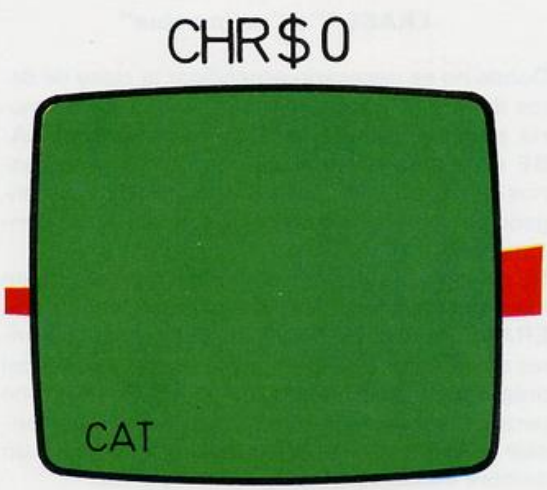
La unidad MICRODRIVE permite el proceso de carga automática de un programa contenido en cartucho, cuyo nombre debe ser **run**. Esta función sólo es accesible cuando se enciende por primera vez el ordenador, o bien después de ejecutar un comando **NEW** o la inicialización del Sistema mediante un **USR 0**.
Para invocar esta función basta con escribir, después de cumplirse alguna de las condiciones antes expuestas, la palabra clave **RUN** seguida de **ENTER**. Con esta operación, el ordenador busca y carga en memoria el programa contenido en el cartucho del MICRODRIVE 1 cuyo nombre es **run**.
Si el programa ha sido grabado con la opción de autoejecución, el caso más frecuente, éste se ejecutará automáticamente a partir de la línea especificada.
Esta opción permite, la creación de un programa que sirva de Menú de acceso a los contenidos del cartucho, con la ventaja de cargarse de forma automática sin requerir conocimientos del BASIC SINCLAIR. Una muestra de su utilidad puede encontrarse en el propio cartucho de demostración suministrado con cada unidad MICRODRIVE.

La carga automática de un programa por el microdrive sirve para la creación de menús de acceso a las contenidas del cartucho.

Para poder crear un programa de este tipo basta con confeccionar un programa BASIC, cumpliendo las condiciones deseadas, y grabarlo en cartucho con el nombre **run** (debe escribirse en minúsculas) y autoejecución a la línea de su comienzo.

PROTECCION DE PROGRAMAS

El MICRODRIVE nos brinda como sistema de protección la invisibilidad de los nombres de programas en el directorio (CAT), *con el prefijo* CHR$ *0.*



Además de la prohibición del **MERGE** * de los programas grabados con el parámetro **LINE**, el MICRODRIVE nos brinda otro sistema de protección, consistente en la invisibilidad de los nombres de programas y datos en el directorio **(CAT)** del MICRODRIVE.
Para conseguir este efecto basta con anteponer al nombre del programa, bloque de memoria, pantalla o matriz de datos, el prefijo **CHR$ 0** antes de la grabación, con o sin **LINE**, de la forma:

SAVE *''m'';n;CHR$ 0+''nombre'' LINE x

Conviene tomar la precaución de anotar los nombres de los programas así grabados, dado que al no aparecer en el directorio del cartucho no podremos recuperarlos si olvidamos su nombre.

PROCESADORES DE IDEAS

La polémica suscitada en torno a la inteligencia artificial en ordenadores, no vislumbra una solución que unifique criterios sobre el tema. Muchos consideran procesos tales como la robótica o la síntesis de voz dentro de este campo, pues de alguna manera en ellos las máquinas son capaces de reproducir, más o menos fielmente, ciertas facetas del comportamiento humano, aunque ejecuten un trabajo meramente mecánico o poco creativo.

Otros no dudan en señalar que las actividades en las cuales la máquina sea capaz de realizar labores típicas de inferencia y relación de ideas, a fin de cuentas que demuestre por sí misma una capacidad para analizar sus «pensamientos», son las únicas válidas para ser consideradas dentro de este campo.

Pero, de una u otra manera, los avances realizados durante los últimos años pueden considerarse como un éxito, y no olvidemos que lejos de la utópica creencia por la cual los más sensacionalistas vaticinan un mundo futuro dominado por máquinas, no por hombres, los progresos realizados dentro de la inteligencia artificial están al servicio de todos. Demos, pues, un repaso al estado actual de esta área de la informática en constante desarrollo.

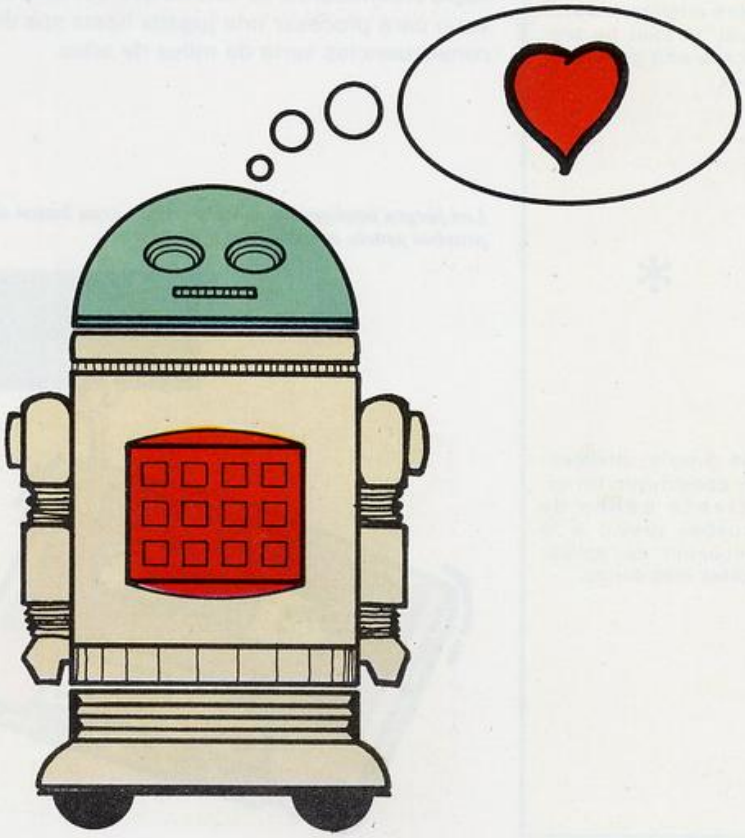
UN BANCO DE PRUEBAS

Resulta increíble como algunos de los grandes precursores de la inteligencia artificial, como Turing o Shannon. dedicaron parte de su trabajo a desarrollar juegos inteligentes o que al menos parezca que lo son. Es más, en los orígenes de la inteligencia artificial, fueron muchos los que aseguraron, tan rotunda como equivocadamente, que pronto una máquina sería capaz de vencer al campeón del mundo de ajedrez. Por supuesto, incluso en nuestros días, el mejor de estos programas no tiene ninguna posibilidad, no ya contra un gran maestro, sino contra un jugador de tipo medio.

Entonces, si los intentos en este sentido han constituido un sonoro fracaso, ¿por qué se sigue investigando? Un programa para jugar al ajedrez, por ejemplo, con toda perfección y garantía de éxito, debería implementar desde el primer movimiento toda la gama posible de jugadas y sus variantes conforme avanza la partida: debería tenerlo previsto todo.

De hecho, el problema teóricamente no estriba en la construcción del programa, difícil de por sí, sino en el elevado tiempo de proceso necesario para concluir una partida. Supongo que nadie es-

Unas de las numerosas definiciones de inteligencia artificial, sostiene que ha de tener capacidad de análisis de pensamiento por sí misma.



¡!

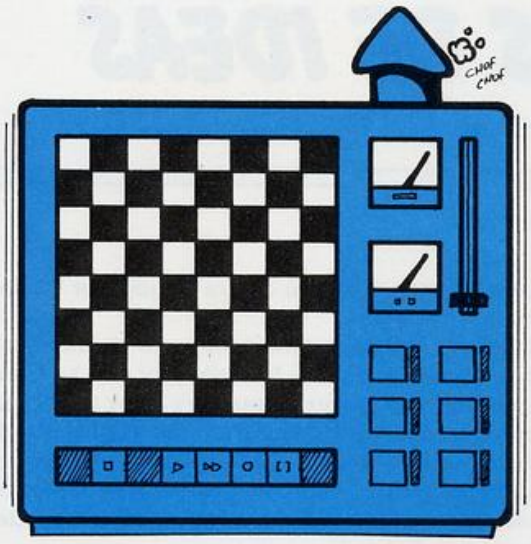
Los sistemas expertos se encargan de almacenar los conocimientos de las personas especializadas en temas específicos.

*

No existe un criterio que unifique las distintas concepciones sobre inteligencia artificial, lo cual ha provocado una gran polémica.

*

Los juegos inteligentes constituyen un excelente banco de pruebas previo a la ejecución de aplicaciones más serias.



En los orígenes de la inteligencia artificial se aseguró que una máquina sería capaz de vencer al campeón del mundo de ajedrez.

taría dispuesto a pasar docenas de años frente a una máquina esperando su genial movimiento. A pesar de la velocidad de proceso en los grandes superordenadores, se estima que el tiempo necesario para procesar una jugada hasta sus últimas consecuencias sería de miles de años.

Los juegos inteligentes constituyen un gran banco de pruebas previo a aplicaciones concretas.

Para evitarlo, se establecen ciertas simplificaciones, y es aquí donde radica la importancia de los juegos inteligentes: constituyen un excelente banco de pruebas previo a la ejecución de aplicaciones más serias. Por ejemplo, se desarrollan funciones de evaluación, donde se incluyen el valor teórico de cada pieza o la importancia relativa del dominio sobre ciertas casillas del tablero, es decir, el ordenador es capaz de identificar como movimiento suicida el cambiar un alfil enemigo por su dama. Por tanto, tiene una cierta «consciencia» y en función de ella es capaz de tomar decisiones.

Pero claro, esto no basta. Además se incorporan al programa principios heurísticos, los cuales facultan a la máquina para conseguir su objetivo: ejecutar un movimiento, utilizando técnicas basadas más en la experiencia que en el cálculo. Desde luego, de esta manera no se garantiza que la decisión adoptada por el ordenador sea la mejor, pero sí se asemeja la forma de jugar de la máquina a como lo haría un humano, imitando su capacidad de improvisación.

SISTEMAS EXPERTOS

El estudio de las funciones de evaluación y la inclusión de principios heurísticos, ha conducido a otro logro dentro de la inteligencia artificial: los sistemas expertos. Estos se encargan de almacenar los conocimientos de las personas especializadas en determinados temas específicos, con objeto de facilitar soluciones en caso de ausencia del especialista.

Resuelven difíciles problemas, tan bien o mejor que los expertos humanos, razonan las decisiones que toman, consideran simultáneamente, distintas hipótesis interactivas entre sí, son capaces de dialogar en lenguaje natural, admiten descripciones simbólicas, y desde luego, justifican detalladamente sus conclusiones.

Pero lo más importante es la capacidad para, al igual que ocurre con los conocimientos adquiridos por la experiencia de una persona, manejar datos ambíguos e incluso erróneos. Es decir, simulan la forma de pensar y razonar de un experto humano, ya que estos, muchas veces, son incapaces de explicar el porqué las cosas son así. Pero están seguros de que lo son.

Sus aplicaciones son obvias. Por ejemplo, un sistema de este tipo aplicado a la medicina, solicita del usuario los síntomas que presenta el paciente, y a partir de ellos, facilita el diagnóstico y con

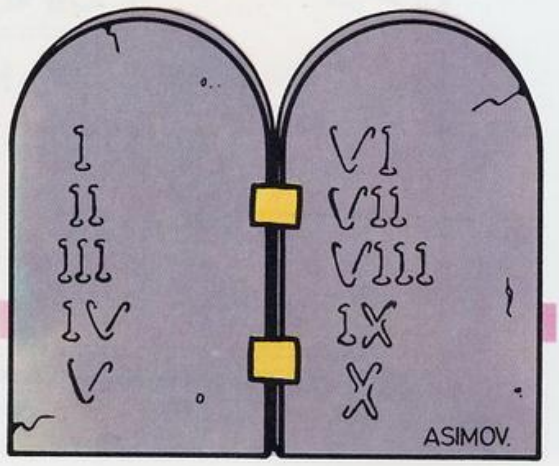
él, la terapia posible a seguir, útil, sin duda, sobre todo si no se encuentra un médico cerca.
O en el control de la seguridad de las grandes plantas de producción de energía nuclear. En un determinado momento, el operario encargado de verificar el correcto funcionamiento de la central, puede verse saturado ante la avalancha de posibles inferencias causales, las cuales conducen inevitablemente al accidente y posteriormente, a la catástrofe. Mantener la «cabeza fría» puede ser de vital importancia, y un sistema experto indicará rápidamente, a la par de conservar los «nervios» en calma, las acciones a tomar para solucionar el problema, evitando que éste vaya a mayores.
No obstante, sus aplicaciones siguen siendo utilizadas de forma puramente consultiva, dejando la decisión final a tomar en manos de los humanos. Hemos de tener en cuenta que son incapaces de llegar a conclusiones intuitivamente, y sus razonamientos no se basan, como en el caso humano, en el sentido común.

Los sistemas expertos se encargan de almacenar los conocimientos de los especialistas.

LA QUINTA GENERACION

Bajo esta denominación se conoce un ambicioso proyecto, emprendido en Japón, encaminado a convertir en realidad la tecnología de la inteligencia en ordenador. El programa cuenta con un presupuesto de 500 millones de dólares y una duración de diez años.
Para ello, el gobierno japonés ha reunido a un grupo de jóvenes «cerebros» dentro del campo de la informática, y en particular, de la inteligencia artificial, ninguno de los cuales en el momento de iniciarse el proyecto contaba con más de treinta y cinco años.
La principal pretensión es la de construir ordenadores de arquitectura distinta de la actual (denominada de Von Neumann), capaces de revolucionar la filosofía existente sobre estas máquinas hasta el momento. Es decir, se pasaría a la tecnología del «qué» debe hacer un ordenador (bastará con decírselo), contrariamente a la de nuestros días, consistente en indicarle, paso a paso, «cómo» debe llevar a cabo su trabajo.
Durante la reciente exposición internacional TSUKUBA'85, celebrada en Japón, brillaron por su ausencia los esperados avances dentro de este campo. En este sentido, las opiniones son para todos los gustos: unos opinan que es todavía pronto para desvelar los logros conseguidos, y por ello se mantienen en secreto, mientras otros indican que la realidad es muy distinta, no habiéndose alcanzado por el momento ningún resultado especialmente significativo.
Pero sí es cierto, no obstante, la fuerte inversión

Isaac Asimov en su ficticio manual de robótica, enumeró una serie de leyes básicas que siempre habría que tener en cuenta.



¡!

Parte del trabajo desarrollado en el campo de la inteligencia artificial se ha dedicado a los juegos inteligentes.

*

Para la escritura de programas inteligentes se han diseñado específicamente lenguajes como: ALGOL, LISP y PROLOG.

*

La síntesis y reconocimiento de voz por parte de un ordenador, es uno de los campos más estudiados en estos últimos años.

El MICRO COMMAND *conectado al Spectrum reconoce el sonido de las palabras que ha aprendido con anterioridad.*

efectuada, tanto por los Estados Unidos, como por Japón en el proyecto y sea cual sea el estado actual de las cosas, el beneficio es palpable, pues se está contribuyendo de forma espectacular al rápido desarrollo de la inteligencia artificial.

El programa japonés sobre inteligencia artificial «La Quinta Generación» cuenta con un presupuesto de 500 millones de dólares.

LENGUAJES

La escritura de programas inteligentes puede efectuarse empleando muchos de los lenguajes existentes en la actualidad, pero al igual que ocurre con otras aplicaciones, algunos han sido específicamente diseñados con este propósito.
Estre los más populares cabe señalar el ALGOL (ALGOritmic Language, lenguaje algorítmico), de estructura similar al FORTRAN (FORmula TRANslator, traductor de fórmulas) pero mucho más flexible que éste. Su origen data de principios de los sesenta, aunque posteriormente se han implementado versiones mejoradas.
Al objeto de facilitar el manejo de información simbólica y grandes listas de datos, surgió en los setenta un lenguaje denominado LISP (LISt Processor, procesador de listas), el cual pronto se popularizó por sus aplicaciones dentro de la investigación para la inteligencia artificial. Un programa escrito en LISP es una serie de listas de funciones matemáticas, las cuales operan sobre otras listas de datos, que cambian, frecuentemente, durante el proceso.
Finalmente, comentar un lenguaje el cual, según todos los indicios constituirá la piedra filosofal en los ordenadores de la quinta generación: el PROLOG (PROgramming in LOGic, programación lógica). Como en los lenguaje anteriores, el manejo de grandes fórmulas configuran su base, pero

desde una óptica diferente, la cual parte de la lógica matemática.

MAQUINAS PARLANTES

La síntesis y reconocimiento de voz por parte de un ordenador, ha sido uno de los campos más estudiados en los últimos años con objeto de conseguir establecer una relación más estrecha entre el hombre y la máquina. Hoy en día, escuchar la voz, más o menos metálica de un ordenador, es una realidad. Nuestro Spectrum no es una excepción y mediante un sencillo interface, el **CURRAH μSPEECH**, repite a través del televisor las frases que nosotros introducimos por el teclado, fuerte y claro.

El reconocimiento de voz es otro tema, debido a la distinta modulación y entonación que cada persona imprime en las palabras que pronuncia. Además, existe también el problema de la ambigüedad en el lenguaje, es decir, una misma palabra puede significar cosas diferentes según su contexto. Es un campo todavía en desarrollo, aunque algunos equipos ya realizan aplicaciones prácticas.

El **MICRO COMMAND** es un convertidor analógico-digital, el cual se conecta en el *slot* de expansión del Spectrum. Mediante un proceso de aprendizaje en el cual repetimos ante un micrófono las palabras que el ordenador deseamos reconozca, éste acaba por hacernos caso, aunque su vocabulario está limitado a unas pocas palabras. Su manejo queda reservado exclusivamente al usuario que le enseñó, siendo necesario volver a reprogramarlo cada vez que el micrófono cambie de manos, pues de otra manera no reconocerá la nueva voz.

VISION ARTIFICIAL Y ROBOTICA

Otra área todavía en desarrollo, es la dedicada a dotar a los ordenadores del sentido de la vista. Hasta ahora, el problema del reconocimiento de objetos está resuelto, siempre y cuando éstos sean planos. Se utilizan cámaras de televisión enfocadas, por ejemplo, a las cintas transportadoras, las cuales en colaboración con robots, indican a éstos el momento en que deben recogerlos, ajustarlos en una máquina para su mecanización y posteriormente, tomarlos de allí y depositarlos en la cinta para su transporte a otra sección de la cadena.

La ficción, antesala de una realidad más o menos próxima, vaticina un futuro de fábricas controladas por enormes monstruos mecánicos, exentas de personal humano: «El poder de la máquina sobre el hombre». No obstante, la situación quizá no sea tan desalentadora, siempre y cuando se respeten leyes básicas como las enunciadas por Isaac Asimov en su ficticio manual de robótica:

I. Un robot no debe dañar al ser humano, o por inactividad, permitir que sea dañado.

II. Obedecerán disciplinadamente las órdenes dictadas por el hombre, siempre y cuando no vayan en contra del primer principio.

III. Un robot debe proteger su propia existencia, hasta el punto, en el cual, no se entre en conflicto con los principios anteriores.

El futuro forma ya parte del presente. INVESTRONICA, presentó durante el S.I.M.O. 84 celebrado en Madrid, un pequeño robot, el cual aceptaba las órdenes que recibía sin titubeos. Su utilidad es discutible, y no pasa de ser un mero entretenimiento, pero es una buena muestra de cómo los robots pueden invadir en breve nuestro entorno doméstico.

Lenguajes como el ALGOL, LISP y PROLOG, se han diseñado para la escritura de programas inteligentes.

¡!

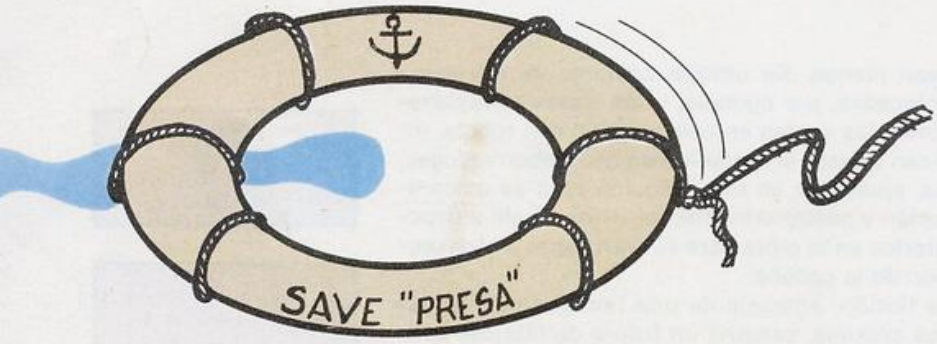
Los sistemas expertos han sido aplicados en el campo de la medicina, y en el de la seguridad de centrales nucleares.

*

La Quinta Generación es un proyecto japonés sobre inteligencia artificial, que cuenta con un presupuesto de 500 millones de dólares y una duracion de diez años.

LA PRESA

LA fértil región de Runlandia está amenazada por unos extraños artefactos voladores que colisionan contra los muros de sus grandes presas, ocasionando terribles destrozos en cada impacto. Sus habitantes han de actuar con rapidez y precisión si quieren evitar el desastre que se les avecina: la inundación de Runlandia. Para ello, se ha situado al pie de la presa un potente cañón que deberá ser disparado con auténtica sangre fría y exactitud por algún voluntario.
¿Quieres ser tú ese avezado voluntario? ¡Sí! Pues saca pecho y adelante. La defensa de Runlandia es tuya.



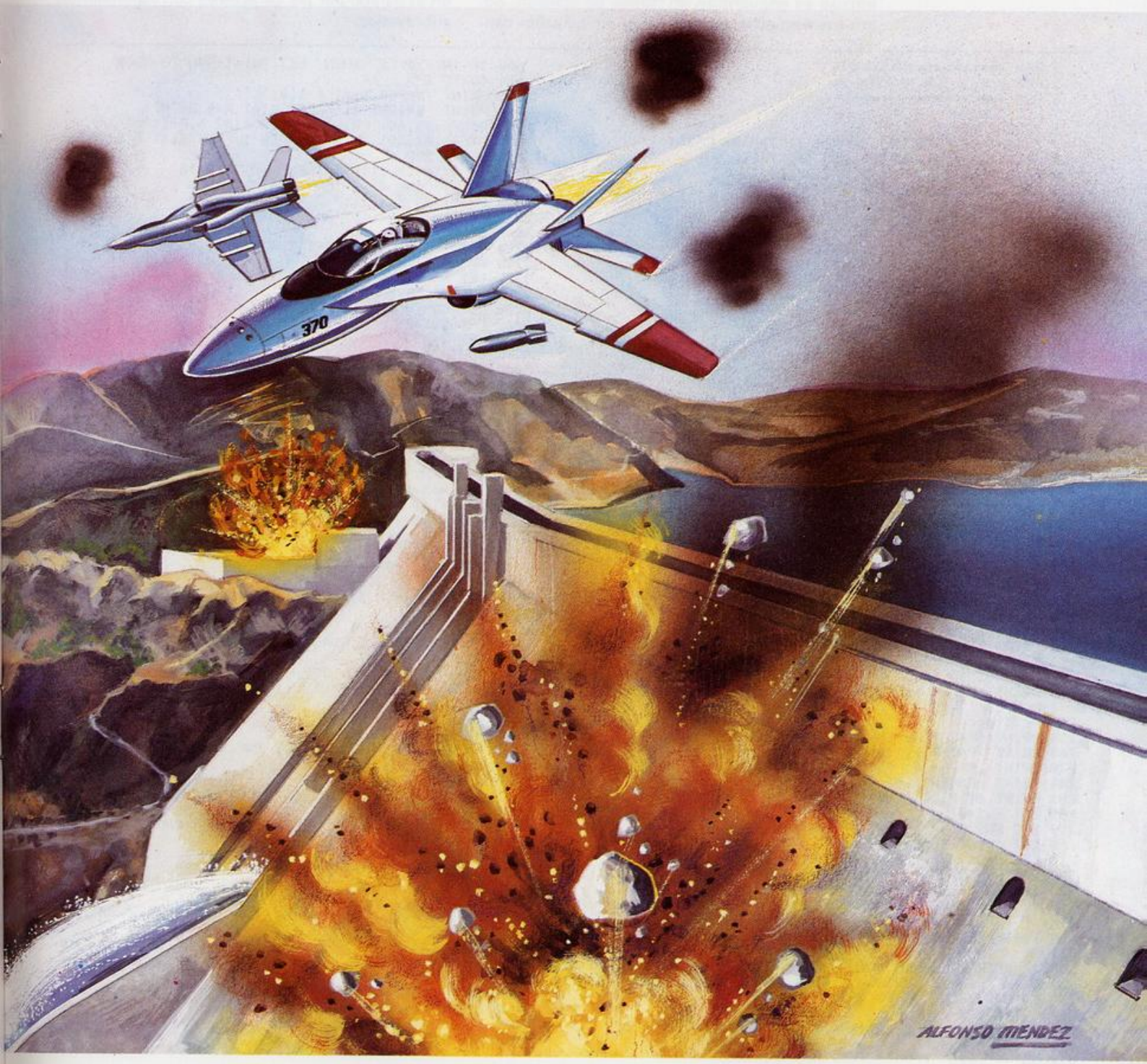
Para «salvar» el programa utilizaremos SAVE "PRESA".

EL PROGRAMA

El juego consiste en impedir que los artefactos voladores que aparecen por la derecha de la pantalla, abran una vía de agua en el muro contenedor de nuestra presa.
Nuestros enemigos aparecerán a una altura determinada por el margen derecho de la pantalla. Esta altura es en cada uno de los intrusos aleatoria, por lo cual nunca tendremos una noción de la misma.
Por cada enemigo destruido, nuestro Spectrum nos dará una puntuación diferente, dependiendo de la altura a la que se encuentre en ese momento nuestro blanco. Estos tanteos se irán sumando sucesivamente en un marcador que el jugador tendrá a la vista en todo momento de la partida.
Por cada impacto que reciba la presa por parte de nuestros enemigos, surgirá en la fachada el impacto lógico del bestial embite, dando impresión de que la presa se desmorona.

Los tanteos se irán sumando en un marcador que el jugador tendrá a la vista en todo momento.

Cada vez que deseemos efectuar un disparo deberemos pulsar la tecla 1.


370
ALFONSO MENDEZ

La partida finalizará cuando el agua que contiene la presa quede libre e inunde una zona de pantalla predeterminada por el ordenador.
Debido a la gran cantidad de gráficos definidos que utiliza el programa, debemos tener muy en cuenta los caracteres subrayados que surjan en el listado.
El manejo del cañón de defensa de la ciudad es bien fácil: sólo tenemos que apretar la tecla **1** cuando consideremos que nuestro proyectil y el intruso se encuentran en rumbo de colisión. Sobre los desperfectos que estos «kamikazes» causan en la presa, es interesante hacer notar que no van abriendo un boquete en la misma, sino que los impactos producen un desmoronamiento parcial del muro, lo cual en ocasiones puede ser una ventaja, pues los residuos de un impacto pueden pasar a tapar el desperfecto ocasionado por una colisión anterior.
La estética del programa se ve amparada por la profusión de gráficos definidos (18 en total), que como es habitual se encuentran representados en el listado por los caracteres correspondientes subrayados.

```
  1 REM ********************
  2 REM * (C) J.M.MAYORAL *
  3 REM ********************
  7 LET CONT=0
  8 POKE 23658,8
  9 BRIGHT 0: PAPER 1: BORDER 1: INK 9: CLS
 10 DATA 255,136,136,136,255,162,162,162
 20 DATA 255,136,136,136,255,34,34,34
 30 DATA 255,139,139,139,255,35,35,35
 40 DATA 255,139,139,139,255,35,255,255
 50 DATA 255,136,136,136,255,34,255,255
 55 DATA 255,136,136,136,255,34,35,35
 60 DATA 255,139,139,139,255,35,35,35
 61 DATA 255,90,44,44,44,44,223,223
 62 DATA 6,11,23,46,92,184,248,252
 63 DATA 240,192,128,0,0,128,192,224
 64 DATA 254,139,136,139,249,38,35,17
 65 DATA 255,136,152,184,205,38,72,207
 66 DATA 191,196,68,34,248,47,36,34
 67 DATA 128,64,48,24,46,16,25,1
 68 DATA 1,12,22,44,220,184,136,224
 69 DATA 15,1,12,30,54,44,88,224
 70 DATA 120,0,184,180,58,28,6,1
 90 DATA 0,16,51,127,243,127,51,16
 99 LET G$="ABCDEFGHIJKLMNOPQR"
100 FOR N=1 TO LEN G$
110 FOR M=0 TO 7
115 READ A
120 POKE USR G$(N)+M,A
130 NEXT M
140 NEXT N
150 LET A$="ABBBBBBC"
160 LET B$="ABBBBFED"
170 LET C$="ABBBBG"
180 LET D$="ABBBBBBG"
185 LET E$="ABBBBBBBBBBBBBBBBBBBBBBBBBBBBG"
190 LET X=17
200 PRINT AT 2,3; PAPER 6; INK 9;A$
210 PRINT  PAPER 5; BRIGHT 1;"   "; BRIGHT 0; PAPER
6; INK 9;B$
220 FOR N=1 TO 15
230 PRINT  PAPER 5; BRIGHT 1;"   "; PAPER 6; BRIGHT
0; INK 9;C$
240 NEXT N
250 PRINT  PAPER 6; INK 9;D$
260 PRINT  PAPER 6; INK 9;D$
270 PRINT  PAPER 4; BRIGHT 1; INK 2;E$
280 PRINT  PAPER 1;AT 20,11;"HI"
281 PRINT AT 0,10;"PUNTOS= ";CONT
285 LET SW1=0
290 LET SW2=0
300 LET A=INT (RND*15)+3
310 LET B=31
320 LET B=B-1
325 IF B=2 THEN  GO TO 5000
330 IF ATTR (A,B)=76 THEN  PRINT  INK 2; BRIGHT 1;AT
A,B;"R": PRINT AT A,B+1;" ": LET XP=A: LET YP=B:: GO
SUB 7000: LET CONT=CONT+10: LET X=18: LET SW1=0: GO
TO 300
340 IF ATTR (A,B)=48 THEN  GO TO 2000
350 PRINT  INK 2; BRIGHT 1;AT A,B;"R"; BRIGHT 0;" "
360 IF SW1 THEN  GO TO 400
380 GO TO 600
400 IF X=1 THEN  LET SW1=0: PRINT AT X+1,11;" ": LET
X=18: GO TO 320
410 GO TO 3000
600 IF INKEY$="1" THEN  LET SW1=1: GO TO 3000
620 GO TO 320
2005 PRINT  PAPER 6;AT A-1,B;"L"
2007 PRINT  BRIGHT 1; PAPER 6;AT A+1,B;"M"
2010 BEEP .01,40: PRINT  PAPER 6;AT A,B-1;"K"
2020 BEEP .01,45: PRINT  PAPER 1; INK 6;AT A,B;"J "
2040 GO TO 300
3000 IF ATTR (X,11)=74 THEN  GO TO 3200
3010 PRINT  INK 4; BRIGHT 1;AT X,11;"H"
3020 PRINT AT X+1,11;" "
3030 LET X=X-1
3040 GO TO 320
3200 PRINT  INK 4; BRIGHT 1;AT X,11;"H"
3210 PRINT AT X+1,11;" "
3220 LET XP=X: LET YP=10
3230 GO SUB 7000
3240 LET CONT=CONT+10
3250 LET SW1=0
3260 LET X=18
3270 GO TO 300
5000 REM PRESA ROTA
5010 REM FIN DE JUEGO
5030 PRINT  PAPER 5;; BRIGHT 1;AT A,0;"   "
5035 LET F=A
5040 PAPER 5: BRIGHT 1
5050 PRINT AT A,B+1;" "
5060 PRINT AT A+1,B;"R"
5065 PRINT AT A+1,B;" "
5070 FOR N=A+1 TO 17
5080 PRINT AT N+1,B-1;"R"
5090 PRINT AT N,B-1;" "
5095 BEEP .06,40-N/2
5100 NEXT N
5110 REM  SALIDA DE AGUA
5520 FOR N=3 TO 31
5530 PRINT  PAPER 5; BRIGHT 1; OVER 1;AT F,N;" "
5550 NEXT N
5560 FOR N=F+1 TO 21
5570 PRINT  PAPER 5; BRIGHT 1; OVER 1;AT N,31;" "
5580 NEXT N
5590 FOR N=21 TO A-2 STEP -1
5600 PRINT  PAPER 5; OVER 1; BRIGHT 1;AT N,0;"
5601 NEXT N
5605 FOR N=3 TO A-3
5607 PRINT  PAPER 1; OVER 1; INK 6;;AT N,0;"   "
5610 NEXT N
5614 BRIGHT 1: BORDER 5
5615 PRINT  OVER 1;AT A,0;" PULSA UNA TECLA PARA COME
NZAR "
5620 IF INKEY$="" THEN  BEEP .01,INT (RND*23)+20: GO
TO 5615
5630 RUN
6999 REM   EXPLOSIONES
7000 PRINT AT XP,YP;"  "
7005 FOR Q=1 TO 6
7010 PRINT  OVER 1; BRIGHT 1; INK INT (RND*6)+2;AT XP
,YP;"NO";AT XP+1,YP;"PQ"
7015 BEEP .01,40
7020 NEXT Q
7030 PRINT AT XP,YP;"  "
7040 PRINT AT XP+1,YP;"  "
7045 LET CONT=CONT+20-A
7050 PRINT AT 0,18;CONT
7060 RETURN
```