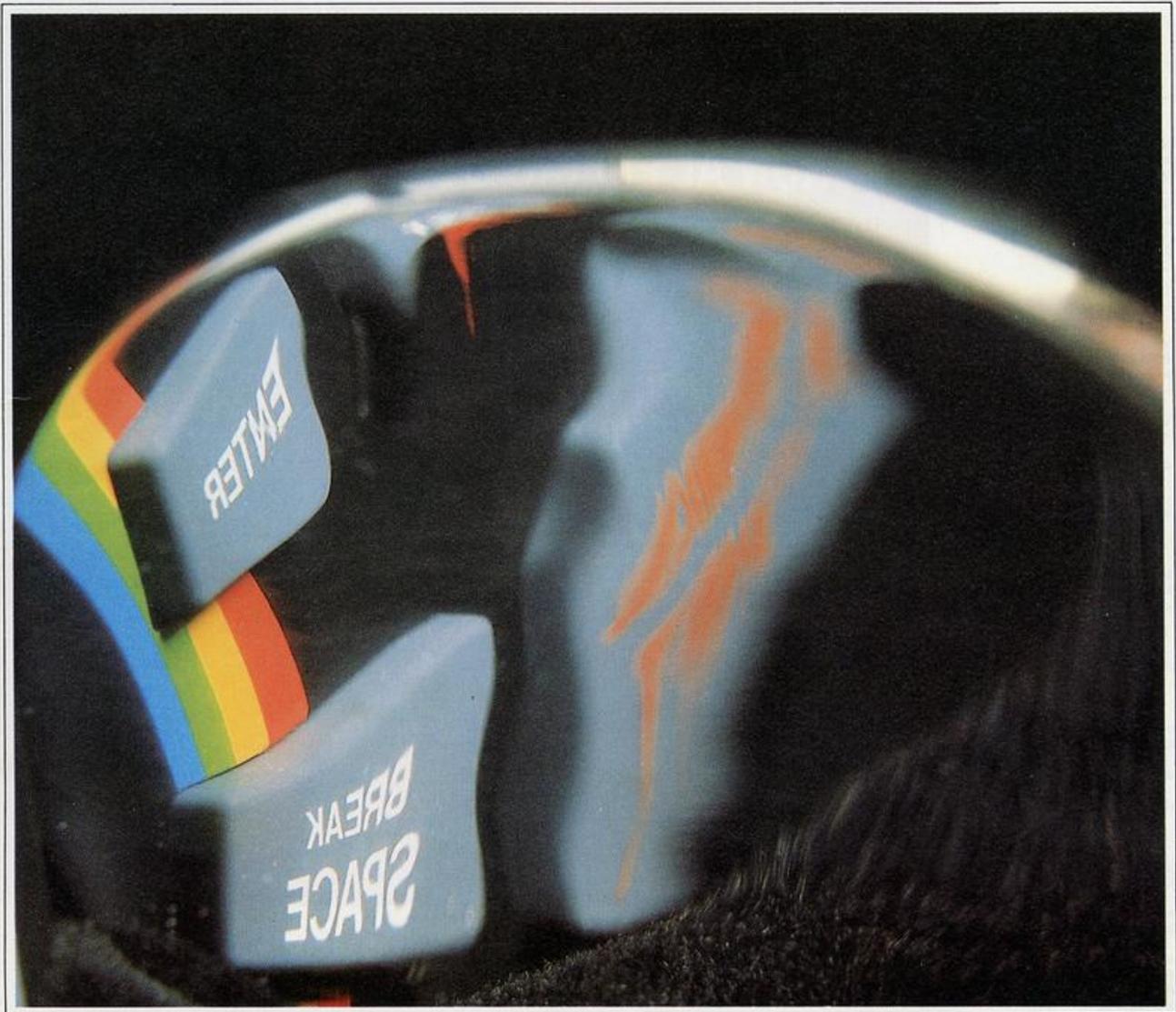


38  
150pts.

# AVUN

Enciclopedia Práctica del Spectrum



Nueva Lente/Ingelek



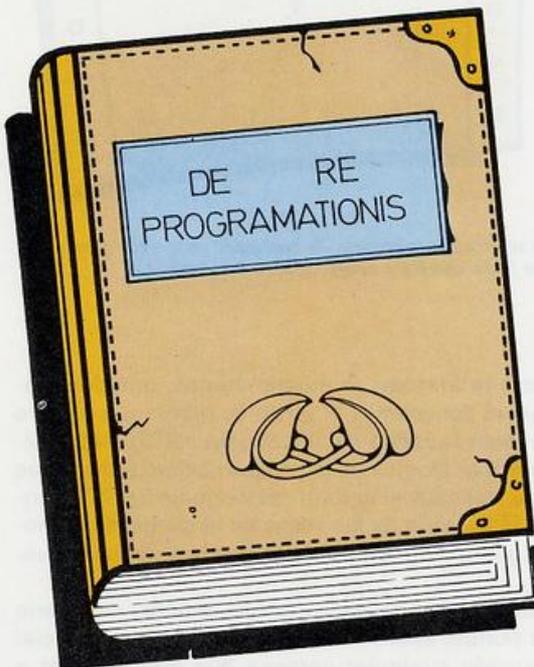


# DE COMO HACER BUENOS PROGRAMAS

**S**eguramente, el título de este capítulo resulte engañoso, y nos pueda hacer pensar que nos encontramos ante un obsoleto tratado de ciencia medieval; nada más lejos de la realidad. Lo cierto es que a los medios informáticos profesionales o pseudoprofesionales (firmas de software, editoriales de prensa técnica informática, academias, escuelas, e incluso universidades), acuden cada vez con más frecuencia, personas que dicen contar entre sus conocimientos con el de la programación BASIC. Lamentablemente, un gran número de ellos se encuentran muy equivocados, y no por propia culpa, sino animados por la creencia erróneamente extendida, de que cualquiera que conozca la misión de un puñado de instrucciones puede considerarse un programador.

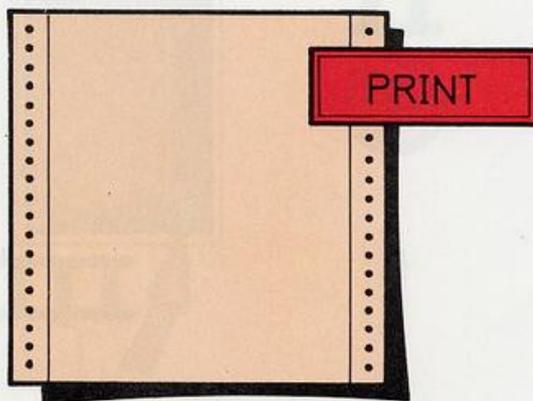
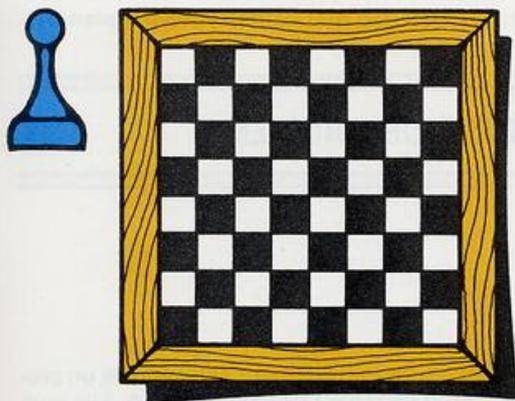
## EL «ARTE» DE PROGRAMAR

El conocer los movimientos de las piezas de ajedrez no nos permite asegurar que sabemos jugar al difícil ajedrez; de forma similar, el conocimiento de un sencillo conjunto de palabras BASIC, máxime si entendemos el idioma inglés, no nos



*La buena programación no forma parte de un obsoleto tratado de ciencia medieval.*

*Al igual que conocer los movimientos de las piezas de ajedrez no implica saber jugar al ajedrez, el conocer el uso de unas cuantas instrucciones no supone saber programar.*



Una vez confeccionado el algoritmo correspondiente, es conveniente que sea plasmado en forma escrita, de manera clara, y añadido a la carpeta de documentación.



Un organigrama es un conjunto de símbolos estandarizados, que utilizamos en informática, para facilitar el paso desde el algoritmo hasta la confección del programa en lenguaje.



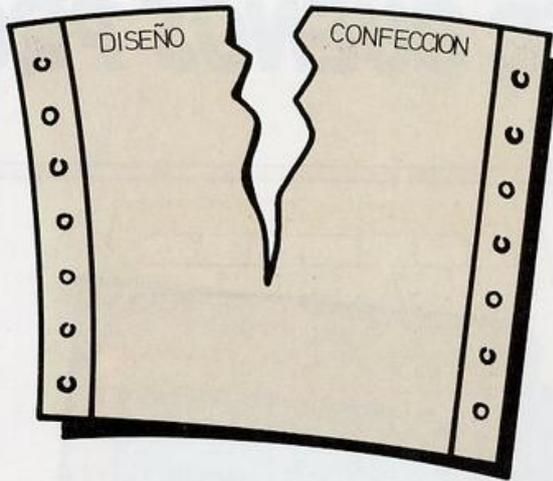
Todos los consejos dados para la fase de diseño de un programa, son de carácter general, aunque lógicamente la simplicidad de algunos programas, hace de determinados punto un gasto inútil de tiempo y esfuerzo que puede ser evitado.



El enlace entre las fases de diseño y confección del programa en lenguaje, viene dado por la traducción del algoritmo a un organigrama.

\*

Antes de comenzar la confección del programa en lenguaje, hemos de seleccionar aquel que se encuentre mejor orientado a la resolución del problema que nos ocupe.



*La realización completa de cualquier programa consta de dos fases: diseño y confección en lenguaje.*

permite afirmar, ni mucho menos, que dominamos ni conocemos el «arte» de programar (dentro del más humilde significado de ARTE), ni tan siquiera la técnica de la programación, puesto que ésta traspasa el umbral del lenguaje BASIC, y entra la mayoría de las veces en el campo de la teoría general compartida por muchos otros lenguajes.

Pretendemos en este capítulo exponer una serie de puntos sobre los cuales debe poner especial atención cualquier persona que quiera llegar a considerarse programador, o más concretamente, programador en BASIC; no obstante, no suponen más que una tarea a desarrollar por aquel que persiga tal objetivo, que será alcanzado en mayor o menor medida, y en un tiempo más o me-

*El organigrama es el punto de enlace entre las fases de diseño y confección.*

nos prolongado, según la habilidad particular de cada uno, y la experiencia adquirida en el tema.

## DISEÑO DE UN PROGRAMA

La realización completa de cualquier programa envuelve dos fases bien diferenciadas: el diseño y la confección en lenguaje. En la primera, debemos edificar el soporte necesario para que en la fase de confección, no se presente problema alguno en la traducción al lenguaje escogido por nosotros mismos. Esta fase de diseño es por tanto de vital importancia, y merece sin duda un tratamiento en profundidad.

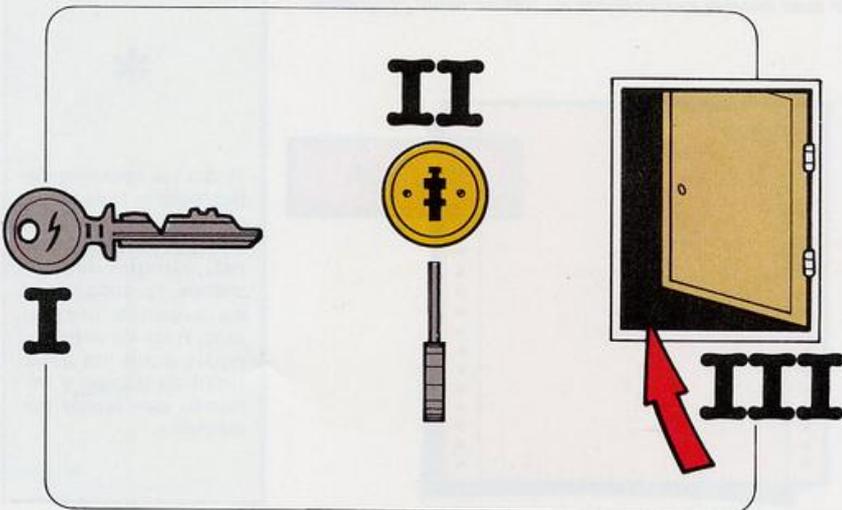
Como bien sabemos, el ordenador no es una máquina capaz de resolver un problema por sí misma, sino que precisa de la recepción de instrucciones muy concretas por parte de los humanos, expresadas en un lenguaje informático, que le muestren el algoritmo para la resolución de dicho problema, sin posible ambigüedad; dicho algoritmo se plasma en un PROGRAMA.

De esta definición de programa se desprende que nosotros, aun auxiliados de nuestro ordenador, por potente que este sea (descontando por supuesto los pocos conocimientos que incorpora su *firmware*: sumar, escribir en la pantalla, etc...) jamás seremos capaces de resolver cualquier problema que no sepamos resolver por nosotros mismos.

Entonces, ¿para qué sirve un ordenador? Sencillamente para que en los casos en que el algoritmo sea conocido, pero la forma de llevarlo a término sea costosa, este «trabajo duro» pase al ordenador, liberándonos de la pesada carga, y por otra parte, para que una vez realizado un programa general, y almacenado convenientemente, podamos utilizarlo para resolver cualquier problema de ese tipo que se nos vuelva a plantear.

## ESTUDIO DEL PROBLEMA

Así pues, el primer paso para el diseño de un programa, es el ESTUDIO DEL PROBLEMA. Ello pue-



de resultar sencillo en determinadas ocasiones, o bastante complejo en otras. En este último caso, es importante que obtengamos la mayor documentación posible, a través de libros, otros programas, etc... y que esta información sea almacenada (la fotocopia suele ser un buen sistema) en una carpeta que denominaremos CARPETA DE DOCUMENTACION, la cual deberá comenzar, antes incluso que la documentación mencionada, con una descripción del objetivo de nuestro programa.

Acercas de los elementos que iremos incluyendo en esta carpeta de documentación, diremos que no es importante su forma sino su fondo; así pues, no debemos poner gran empeño en redacción, buena letra, etc..., sino que basta con que se encuentren ordenados y en perfecto estado de legibilidad, no importa por ejemplo que se escriban en taquigrafía, siempre y cuando nosotros los entendamos.

En todo caso, y si no representa un gran esfuerzo, es conveniente que sean lo más claros posible para cualquier persona. Pensemos que quizá para nuestra documentación, hayamos recibido la carpeta de documentación del programa de un amigo, y nos resultará agradable que sea fácilmente comprensible, o no precise de un «Champlion» para descifrar los jeroglíficos.

## EL ALGORITMO

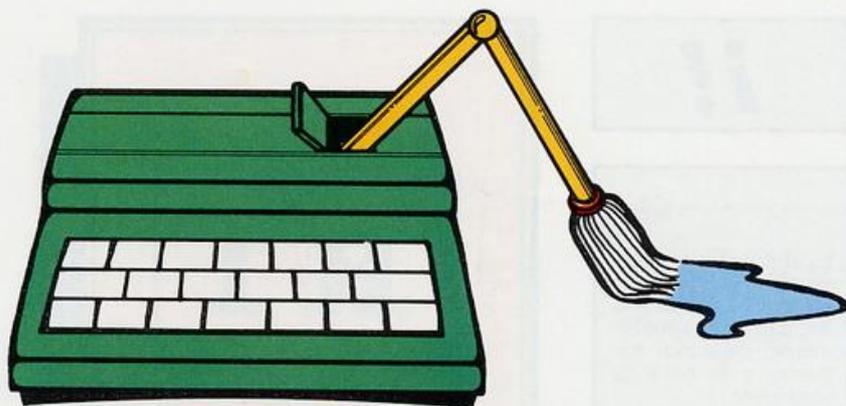
Podemos decir que algoritmo es el conjunto de pasos a seguir para la resolución de un determinado problema. Así por ejemplo, el algoritmo para entrar en nuestra casa podría expresarse de la siguiente forma:

- Buscar la llave.
- Meter la llave en la cerradura.
- Entrar en casa.

O también...

- Situarse ante la puerta.
- Meter la mano en el bolsillo.
- Localizar la llave.
- Extraer la llave.
- Introducir la llave en la cerradura.
- Girar la llave.
- Empujar la puerta.
- Entrar en casa.

Como vemos, los algoritmos se encuentran en nuestra mente, pero la forma de plasmarlos en palabras, o más concretamente, el nivel de detalle con que los plasmemos, depende de cada uno. En general, debemos decir que el enunciado de

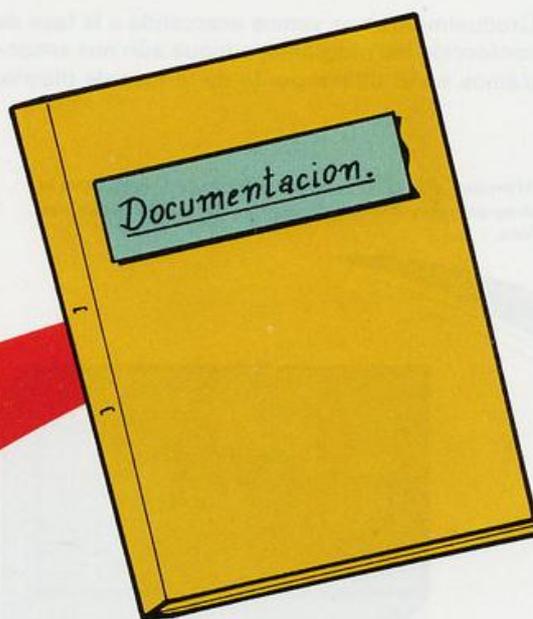


*El ordenador sin duda puede liberarnos de los trabajos intelectuales más pesados.*

un algoritmo debe expresar claramente y sin posible ambigüedad el sistema para la resolución de un problema, aunque dejando al margen los pequeños detalles, que por obvios no contribuyen a la clarificación del mismo.

Así pues, el siguiente paso en el diseño de un programa es, en base al estudio de la documentación acumulada, la confección del correspondiente ALGORITMO DEL PROBLEMA, y su enunciado para la inclusión en la carpeta de documentación.

*La información incorporada a la carpeta de documentación, debe ser lo más clara posible, evitando los «jeroglíficos» que no entendamos pasados unos días.*



! !

La realización completa de cualquier programa envuelve dos fases bien diferenciadas: el diseño y la confección en lenguaje.

\*

El ordenador tiene la enorme utilidad de evitarnos el trabajo pesado, intelectualmente hablando, y posibilitar el enfrentarnos ante el problema una sola vez, y no cada vez que se plante, es por tanto una «inversión» rentable el aprender a programar correctamente.

**i!**

La utilización del código máquina no es en absoluto un sinónimo de calidad; simplemente, de mayor esfuerzo a la hora de programar.

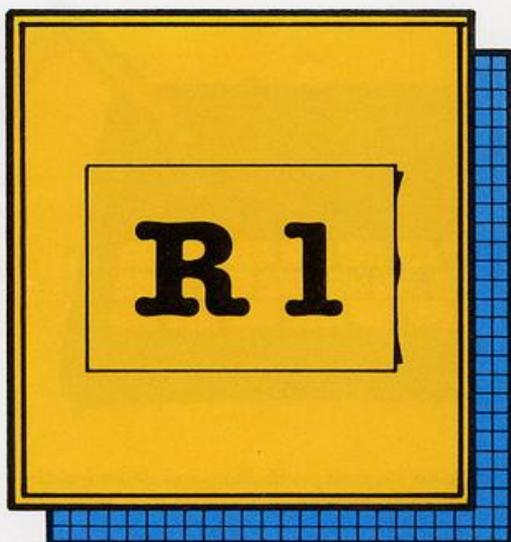
\*

La confección del algoritmo de nuestro problema es un paso esencial en la programación de calidad.

\*

En resumen, podemos expresar los siguientes puntos tendentes a la realización de un buen programa:

- Estudio previo del problema.
- Apertura de una carpeta de documentación.
- Diseño y enunciado del algoritmo del problema.
- Confección de organigramas.
- Elección de un lenguaje orientado al problema.



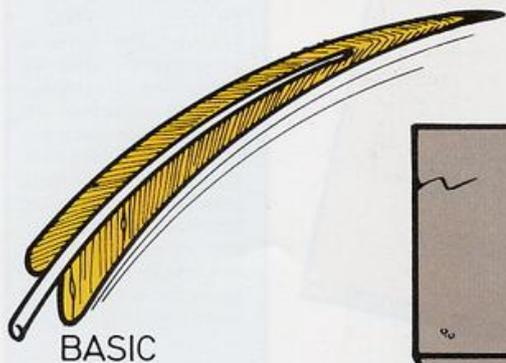
¿COMO SE LLAMA ESA REVISTA?

*Los algoritmos pueden expresarse de manera más o menos detallada.*

## ORGANIGRAMAS

Gradualmente nos vamos acercando a la fase de confección del programa, aunque aún nos encontramos en el último punto de la fase de diseño,

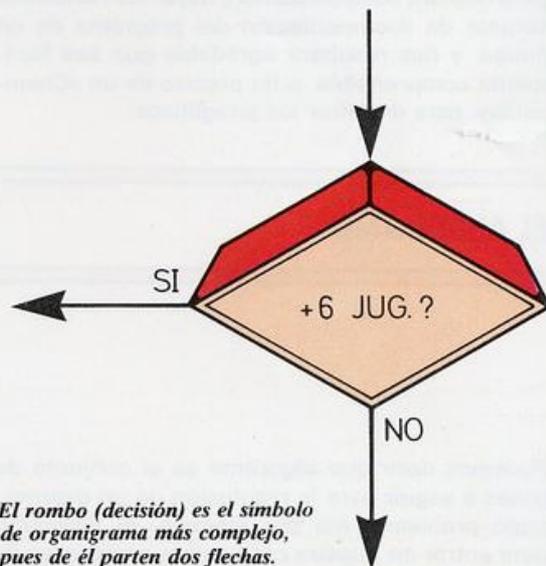
*Mientras el BASIC resulta un lenguaje liviano para el programador, el código máquina es pesado como una losa.*



el cual sirve de enlace con la posterior confección.

La estructura rígida de un ordenador, implica en ocasiones la adaptación de los algoritmos para que éstos sean fácilmente plasmables en forma de programa. Por tanto, el paso siguiente al enunciado en lenguaje humano del algoritmo, es un enunciado del mismo en un «lenguaje» puente entre el humano y el de programación, que facilite el «gran salto». Este pseudo-lenguaje es el ORGANIGRAMA.

Un organigrama es un conjunto de símbolos, de significado estandarizado, que expresan claramente un determinado proceso informático (el algoritmo). Los elementos de un organigrama son una serie de figuras geométricas, cada una de las cuales tiene un determinado significado que se concreta mediante la escritura de un breve comentario en su interior. Naturalmente, estos elementos deben seguir un orden, y éste viene expresado mediante unos segmentos dotados de sentido (flechas), que indican de forma inequívoca qué elemento sigue al elemento anterior.



*El rombo (decisión) es el símbolo de organigrama más complejo, pues de él parten dos flechas.*

Así por ejemplo, podemos encontrar un rombo con la inscripción interior «MAS DE SEIS JUGADORES?». Esto implicará que este punto del algoritmo, precisa conocer si existen más de seis jugadores (el rombo siempre simboliza las decisiones). Pues bien, de este rombo partirán dos flechas con destino a otro elemento. En una de ellas se expresará la leyenda «SI», indicando que es el camino a seguir cuando existan más de seis jugadores, y en otro la leyenda «NO», significando el caso contrario. Como vemos, la pregunta ha de ser siempre tan concreta que sólo admita una respuesta afirmativa o negativa.

Aunque hemos elegido quizá el elemento de organigrama más complejo (ROMBO), puesto que simboliza la decisión y es el único del cual par-



ten dos flechas, creemos que la idea general de organigrama habrá quedado suficientemente clara. En todo caso, la realización de un buen organigrama es algo muy importante con vistas a la posterior fase de confección, y por ello será un tema a tratar independientemente en el próximo capítulo.

Por último, baste decir que el organigrama del mismo modo que el algoritmo, debe ser lo suficientemente concreto como para que no deje lugar a dudas, aunque no es necesario, e incluso resulta conveniente, que evite los detalles superfluos.

En lo referente a su claridad, o más concretamente a la claridad de las leyendas interiores a las figuras, puesto que su simbología es estándar y no deja lugar a dudas, ésta debe ser la máxima posible, con vistas a un uso posterior por otra persona diferente a la que diseñó el organigrama. En todo caso, es fundamental incluir el organigrama o los organigramas, puesto que su longitud puede hacer necesario el seccionamiento del mismo en grupos de pasos del algoritmo, dentro de la carpeta de documentación.

---

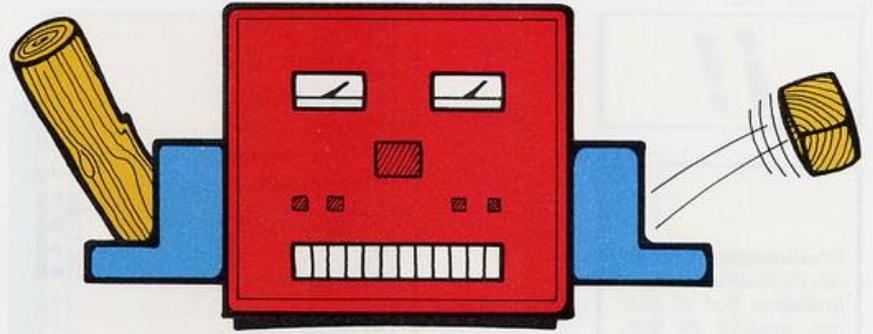
## CONFECCION EN LENGUAJE

---

Por fin entramos en la fase de confección del programa concreto, en forma de lenguaje de programación. Lógicamente, el primer problema que se plantea, antes de zambullirnos en el mar de instrucciones y sentencias, es ¿cuál es el lenguaje que debemos emplear? La contestación a esta pregunta es sin duda de vital importancia, puesto que de ella depende en gran medida el éxito de nuestro programa, y sobre todo el esfuerzo de programación que nos exija.

Como bien sabemos, no podemos asegurar tajantemente que un lenguaje sea mejor que otro, sino simplemente más adecuado para tal o cual tipo de problema. Al fin y al cabo, los diferentes lenguajes de alto nivel no son más que LENGUAJES ORIENTADOS AL PROBLEMA. En principio sólo haremos aquí un breve comentario de introducción acerca de la relación BASIC-Código máquina, por ser los únicos dos en que el Spectrum puede ser programado directamente, en su configuración básica.

Sin duda en muchas ocasiones hemos oído cantar las excelencias del código máquina, o a algún amigo fanfarronear: «este programa lo he hecho en código máquina». Recordemos siempre que

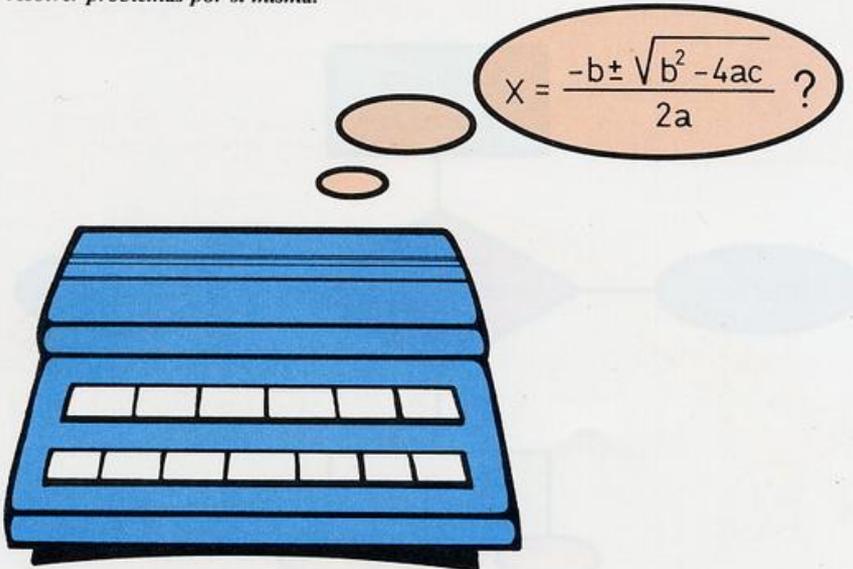


*El ordenador es un sistema de entrada, proceso y salida.*

cualquier programa BASIC puede ser confeccionado en código máquina y de hecho, ese trabajo lo lleva a cabo el intérprete BASIC presente en nuestra ROM. De esta afirmación se desprende que un programa en código máquina puede ser igualmente malo que en BASIC; es decir, no nos dejemos deslumbrar por las palabras código máquina, que no son en absoluto sinónimo de calidad.

Sí son, sin embargo, sinónimo de esfuerzo. No cabe duda que la confección de un programa en código máquina, sobre todo sin el auxilio de un programa ensamblador, es un trabajo mucho más costoso que el de uno equivalente en BASIC; no obstante, un programa no es mejor porque nos haya costado un gran esfuerzo el realizarlo. Por el contrario, aunque resulte paradójico, es muy fácil hacer las cosas complicadas, lo difícil está en hacer las cosas lo más sencillas posible.

*El ordenador no es en absoluto una máquina capaz de resolver problemas por sí misma.*



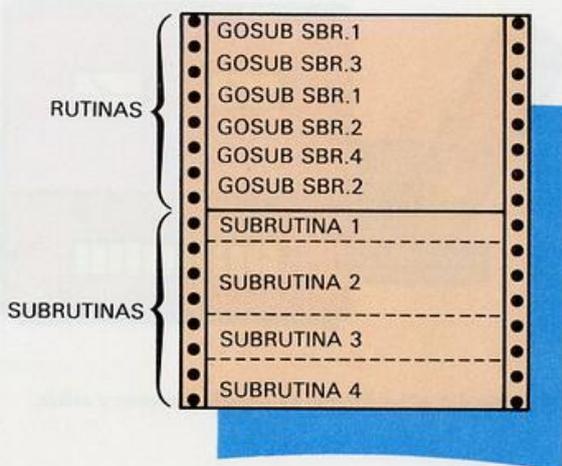


# i!

El ordenador no es capaz de resolver ningún problema por sí mismo, precisa de que, mediante un programa, el humano le explique cómo debe hacerlo.

\*

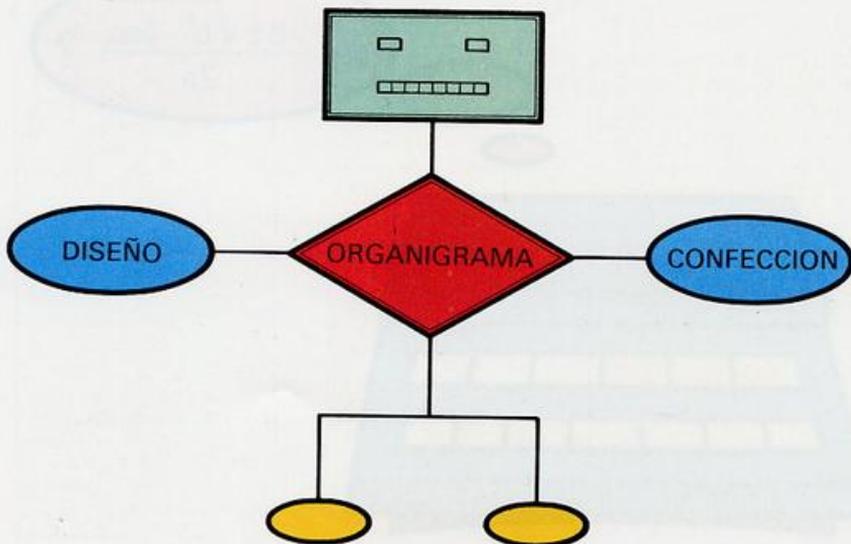
La correcta programación se traduce en mejoras tales como: ahorro de memoria, de tiempo de ejecución, y en general, de disgustos por el mal funcionamiento de los programas.



*Las subrutinas son bloques de programa que son llamados desde diversos puntos del área de rutinas.*

Seguro que si deseamos desplazarnos de un pueblo de Madrid a otro de Barcelona se nos ocurren multitud de malos caminos: podemos ir vía Cádiz, o ¿por qué no hacerle una visita al tío Manolo, misionero en el Zaire? Evidentemente, lo difícil está en buscar el camino más corto, rápido y seguro. Pues bien, extrapolando este ejemplo automovilístico a la programación, este camino ideal no es siempre el código máquina. Así pues, recurramos al código máquina sólo cuando nos sea necesario. La solución ideal suele ser la confección de programas en BASIC en los cuales, un determinado número de rutinas,

*Organigrama es el punto de enlace entre las fases de diseño y confección.*



se encuentran en código máquina, pues precisan de una gran velocidad. Ese es el secreto del código máquina: su enorme velocidad. No la malgastemos; no existe nada más absurdo que una subrutina de retardo en código máquina. Una vez finalizado este aserto en pro del cumplimiento de la ley del mínimo esfuerzo, entremos de lleno en el tema que nos ocupa: la confección del programa en el lenguaje escogido. En nuestro caso, y dado el título de esta sección, será el BASIC.

## FASES DE UN PROGRAMA

Un ordenador puede ser definido de forma sencilla como un sistema destinado a la toma de datos, proceso de los mismos y salida de los datos procesados. En estos tres bloques debe dividirse por tanto un programa. En ciertas ocasiones alguna de estas tres fases es mínima, pero su reducidísima longitud no implica su inexistencia. Antes de enunciar algunos consejos de interés general sobre instrucciones concretas, veamos algunas normas bajo las cuales debería regirse por lo general cualquier programa. En primer lugar, lo deseable en un programa es que las mencionadas fases (entrada, proceso y salida) se encuentren en el orden expresado. Por otra parte, y en relación con el sistema de programación en el cual se concretan estos tres bloques, debemos considerar un orden estricto en la situación de las rutinas y subrutinas y en el uso de las mismas. En primer lugar debemos considerar que las subrutinas son bloques de instrucciones que son llamados varias veces desde diversos puntos de un programa. Por tanto, es tan grande el error cometido al no hacer subrutinas de bloques que podrían serlo, como el de hacer una subrutina de algo que no se utiliza más que en un punto del programa. En este último sentido, existe otra teoría, parte de la denominada programación estructurada, o más concretamente, y en su aplicación al uso de subrutinas, programación modular, que preconiza la utilización de las mismas, de forma que cada una de ellas constituya un bloque de programa con una misión muy concreta, que tenga la ventaja de poder ser puesta a prueba independientemente en una fase de depuración. Así el programa, se compone de una reducidísima zona de rutinas, que tan solo contiene las llamadas a las múltiples subrutinas.

Cada uno deberemos emplear el sistema que más adecuado nos parezca, pero desde luego de una forma consecuyente, es decir, hasta el último extremo, y no una mezcla de ambos métodos de lo que nada bueno puede salir.

Por otra parte, en lo referente a la situación de las subrutinas dentro del texto BASIC, ya sea siguiendo cualquiera de los dos métodos de programación expresados, deberán encontrarse siempre juntas, constituyéndose siempre una zona de rutinas y otra de subrutinas, generalmente posterior, lo cual permite una primera ejecución del programa con un simple **RUN**.

Para terminar con los consejos generales, resaltaremos la importancia de que los programas sean fácilmente comprensibles en su ejecución por cualquiera. Así pues, es de gran utilidad que vayan acompañados de una o varias pantallas de instrucciones, y que en todo momento el usuario sepa qué está ocurriendo, y qué debe hacer a continuación.

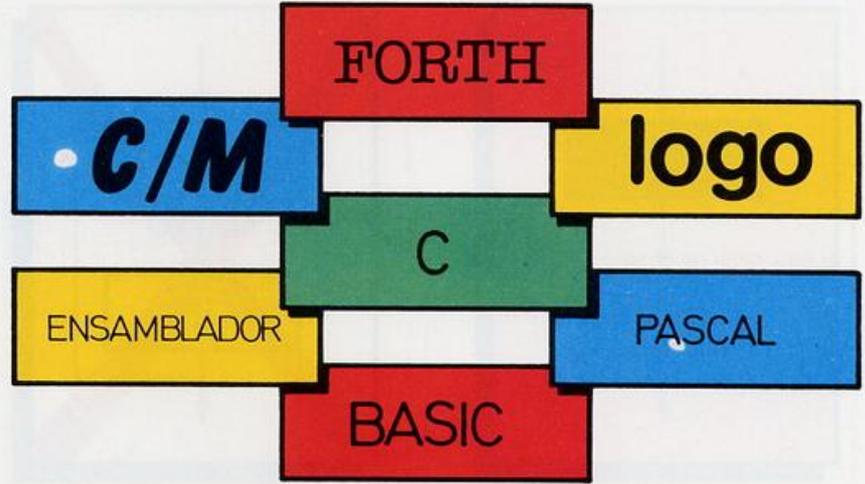
No hay nada más desesperante que observar un simple **INPUT** en una pantalla vacía, sin saber qué dato se nos está pidiendo, o una pantalla en blanco durante mucho tiempo, que sin informarnos del proceso que el programa está llevando a cabo, o al menos de que «su corazón late», nos hace pensar que el programa no funciona y estamos esperando inútilmente un final que nunca llegará.

Otros aspectos como la estética o la depuración de errores hasta el máximo posible (*debugging*), son puntos fundamentales que serán objeto de un tratamiento especial en los próximos capítulos.

Como último consejo sobre metodología general, resaltaremos la conveniencia de una vez finalizada la versión definitiva del programa, incluir en la carpeta de documentación el LISTADO COMPLETO del mismo, que en caso de una «catástrofe magnética» pueda posibilitarnos recuperar el programa perdido, o cómo no, modificar, o completar el programa, que es al fin y al cabo el objetivo primordial de la carpeta de documentación, que puede finalizar con una breve ficha técnica del programa, con datos como nombre, cinta o cartucho en que se ha almacenado, memoria que ocupa, etc...

Así pues, en un breve repaso antes de entrar en los consejos de programación concretos, recordemos los puntos esenciales que conducen a una buena programación, los cuales han ido siendo resaltados en mayúsculas a lo largo de este capítulo:

- Estudio previo del problema.
- Apertura de una carpeta de documentación.
- Diseño y enunciado del algoritmo del problema.
- Confección de organigramas.
- Elección de un lenguaje orientado al problema.

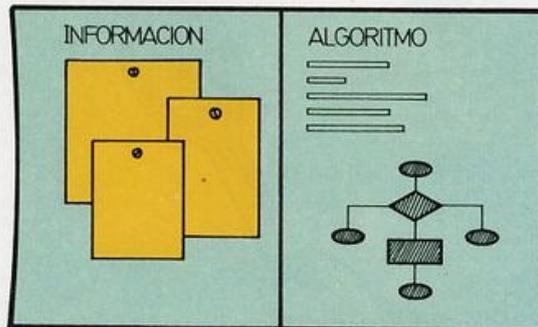


*Es importante elegir un buen lenguaje; es decir, un lenguaje orientado al problema que queremos resolver.*

## CONCRETANDO

Los puntos que a continuación se exponen, son simplemente un reflejo de los «vicios de programación» más ampliamente difundidos, aunque por supuesto, existen muchas otras deficiencias que aquí no se mencionarán, pues serían necesarias hojas y hojas para ello. Con vistas al estudio de estos errores desgraciadamente comu-

*La carpeta de documentación contiene: información sobre el problema, enunciado del algoritmo, organigrama/s y listado/s.*

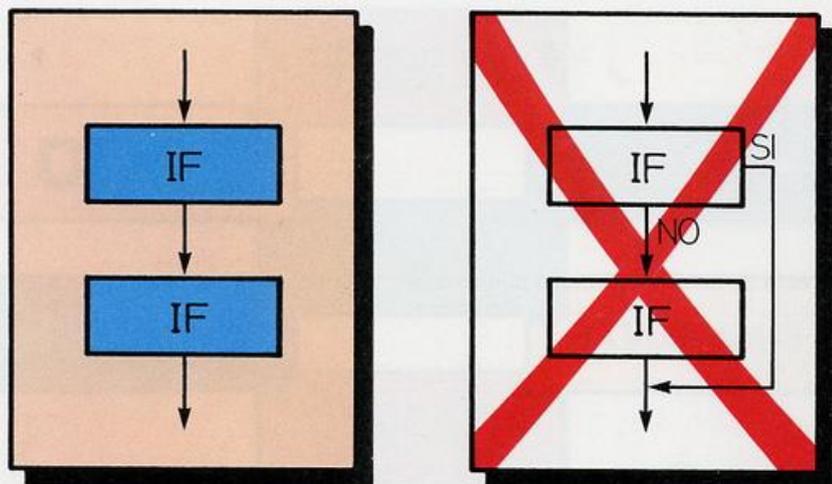


**i!**

Tanto el enunciado del algoritmo, como el propio organigrama, deberán estar, desde el punto de vista de su nivel de detalle, en estricta consonancia con las necesidades para la comprensión del mismo. Ni debemos detallar excesivamente, ni permitir lagunas o ambigüedades.

\*

Programar no es sólo juntar un puñado de instrucciones en determinado lenguaje.



Debemos utilizar siempre estructuras «puente» en la toma de decisiones exclusivas.

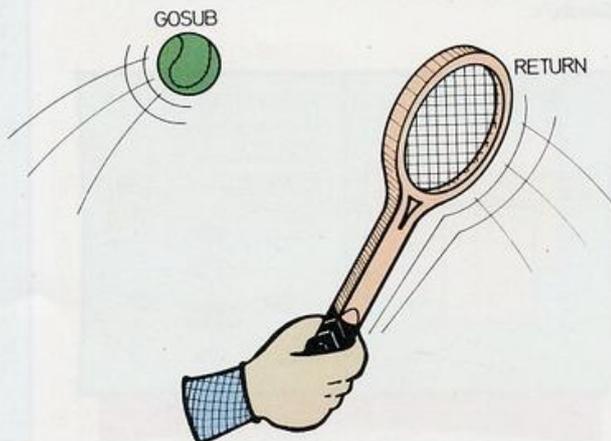
nes, dividiremos el BASIC en diversas zonas de aplicación o bloques de instrucciones.

- Estructuras cíclicas: Hemos de vigilar el máximo aprovechamiento de los valores que toma la variable índice. Estructuras como `FOR I=1 TO 10: PRINT AT I+7,0;"A": NEXT I`, deben ser sustituidas por `FOR I=8 TO 17: PRINT AT I,0;"A": NEXT I`.

Del mismo modo, en estructuras cíclicas vacías, empleadas generalmente como retardos, tengamos en cuenta que el cero es un número tan «válido» como el uno y una línea del tipo `FOR I=1 TO 100: NEXT I`, es tan ilógica como `FOR I=89 TO 198: NEXT I`; la correcta sería `FOR I=0 TO 99: NEXT I`.

- Funciones matemáticas: En el empleo de las funciones matemáticas es necesario depurar siempre las condiciones de error. Es uno de los puntos en que los programas con cálculos fallan

Cualquier vuelta de un GO SUB se debe canalizar por el correspondiente RETURN, pero nunca por un GO TO.



con mayor frecuencia. No permitamos líneas desprotegidas como `PRINT 5/X` (error para X igual a cero), `PRINT SQR X` (error para X menor que cero), o `PRINT TAN X` (error para X igual  $\pi/2$  ó  $3/2\pi$  y sus múltiplos de  $2\pi$ ). En los ejemplos, debemos anteponer líneas de filtro que impidan el paso de los valores erróneos de X.

- Funciones lógicas: La potencia del evaluador lógico Sinclair es muy destacable, y su mal aprovechamiento sustituyéndolo por sentencias de decisión (IF) es sin duda algo a combatir. La siguiente estructura establece una variable «péndulo» entre los valores 1 y 0:

```
10 IF A=0 THEN LET A=1: GO TO 30
20 LET A=0
30 ...
```

No obstante, la misma función podría ser llevada a cabo mediante `LET A=A<>1`.

- Alta resolución: Con mucha frecuencia, sobre todo en el empleo de `DRAW`, se producen errores del tipo **B Integer out of range** (B Entero fuera de rango). Sin duda se deben a que la ejecución del comando le llevaría a trazar una línea que excedería los límites de la pantalla; pongamos por tanto especial cuidado en los parámetros seleccionados para `DRAW`.
- Decisiones: Las decisiones (IF) son sin duda una de las estructuras que más retardan la ejecución de un programa, minimicémoslas mediante el uso de los operadores lógicos, y decisiones excluyentes que nos permitan utilizar «puentes»:

```
10 REM A SOLO PUEDE VALER UNO O CERO EN EL EJEMPLO
20 IF A=1 THEN...
30 IF A=0 THEN...
```

Podría sustituirse por un «puente» del tipo:

```
10 REM A SOLO PUEDE VALER UNO O CERO EN EL EJEMPLO
20 IF A THEN... : GO TO 40
30 ... (LO QUE ANTES SEGUIA A "IF A=0")
40 ... (CONTINUACION DEL PROGRAMA)
```

- Transferencias de control: Por último, hemos de evitar la proliferación de los `GO TO` que puedan marear a cualquiera que pretenda seguir la secuencia del programa; sólo contribuyen a dificultar la depuración y posterior modificación del mismo. Asimismo, es fundamental respetar el punto de entrada a un bloque de programa. Si hemos empleado `GO SUB` siempre deberemos abandonarlo mediante el correspondiente `RETURN`, pero nunca por `GO TO` o introduciéndonos simplemente por secuencia de programa en el área de rutinas (esto sólo sería posible siendo la última subrutina, y encontrándose este área por encima del de rutinas).



# CALIDAD DE IMPACTO



N el capítulo anterior efectuamos un recorrido alrededor de los rasgos más significativos, de carácter general, comunes a una amplia serie de impresoras matriciales compatibles con nuestro Spectrum.

En el presente analizaremos sus virtudes y defectos, juegos de caracteres disponibles, posibilidades gráficas, características especiales, etc... de cada uno de los equipos en particular. Todas se suministran con interface en paralelo, pero opcionalmente, pueden ser conectadas en serie.

Antes de seguir, es conveniente hacer hincapié en un detalle que en muchas ocasiones se pasa por alto: cualquiera de las impresoras a continuación mencionadas sirven para funcionar con una amplia gama de ordenadores; su uso no está restringido al Spectrum.

Por ello, y pensando en futuras ampliaciones de nuestro sistema informático con un ordenador de la misma marca o de otra diferente, la elección debe realizarse bajo la premisa de una mayor compatibilidad, pues tengamos presente que el

precio de estos periféricos puede triplicar o quintuplicar el de nuestro pequeño micro.

## UNA SINFONIA GRAFICA

La nueva impresora STAR SG-10 se revela como un equipo con unas posibilidades gráficas realmente formidables. Sus 96 caracteres ASCII estándar, 88 internacionales, 96 itálicos, además de varios grupos especiales, junto con 7 modos

*Cualquiera de las impresoras que tratamos en este capítulo, sirve para funcionar con una amplia gama de ordenadores.*



**i!**

La impresora Star SG-10 incorpora 96 caracteres ASCII, 88 internacionales, 96 itálicos y 7 modos gráficos definibles por el usuario.

\*

Aunque las impresoras adoptan por defecto el estándar Centronics como interface, no existe ningún problema en solicitarlas incorporando el serie RS-232, con tan solo un pequeño incremento en el precio.

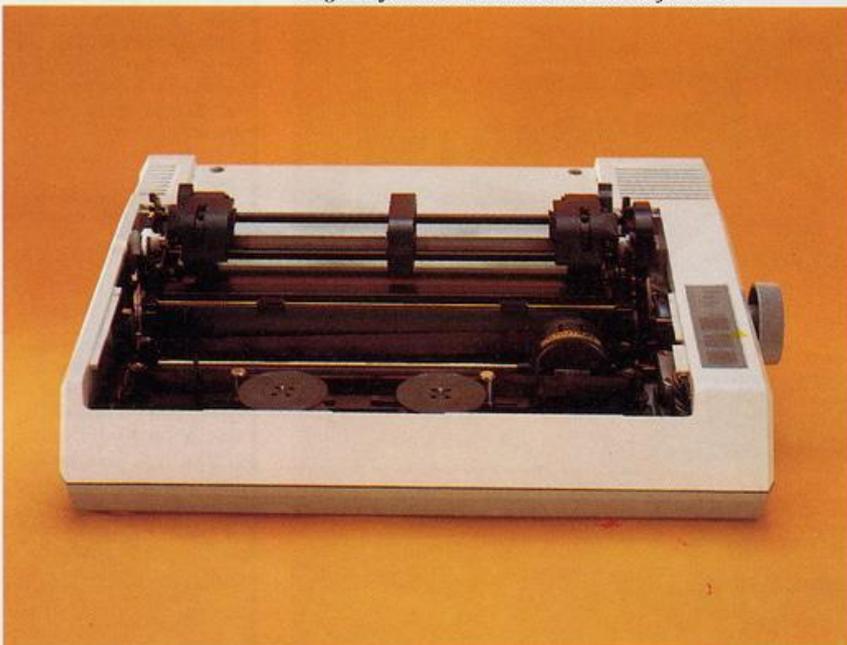


*La nueva impresora STAR SG-10 se revela como un equipo con unas posibilidades gráficas realmente formidables.*

gráficos definibles por el usuario, así lo corroboran.

Pero comencemos por el principio. Tras su desembalado y una vez liberada de los anclajes que fijan los elementos móviles durante el transporte, debemos situar la impresora sobre una superficie plana exenta de vibraciones. Una vez allí, levantando la cubierta superior (se mantiene el di-

*Un selector T/F en la impresora STAR SG-10 permite elegir la forma de arrastre: tracción o fricción.*



seño de los modelos anteriores y sería deseable que se transformara en abatible además de desmontable) aparece ante nuestros ojos todo el mecanismo de impresión. Procedemos a la instalación de la cinta, que es del tipo empleado en las máquinas de escribir, monocroma negra de dos rollos, lo cual garantiza una mayor duración de este elemento y recambio fácil de encontrar, además de económico.

Instalamos en los orificios correspondientes dos rejillas destinadas a servir de guía y separador del papel, tras lo cual, ya podemos cargar éste. Un selector (T-F) permite elegir la forma de arrastre, tracción o fricción. Si escogemos la primera opción, debemos proveernos de papel continuo con los márgenes perforados, de manera que estos ajusten sobre las ruedas dentadas de la unidad de tracción.

Si preferimos trabajar con DIN A4, folio o cualquier otro formato, siempre que su ancho no supere las 10 pulgadas y el espesor esté comprendido entre 0,07 y 0,1 mm, procedemos a desmontar el bloque tractor presionando sobre dos palanquitas colocadas en los laterales de éste y tiramos suavemente hacia afuera, con lo cual abandona su alojamiento. Finalmente, situamos el selector en la posición F e introducimos el papel pulsando LF en el panel de control, para el giro automático del rodillo.

Un selector de cinco posiciones ajusta, en función del espesor del papel, la distancia de separación entre éste y la cabeza de impresión, superándose de esta manera los inconvenientes que en otros equipos provoca esta circunstancia.

Mediante la función de autoprueba podemos hacernos una primera idea de las posibilidades gráficas. Para ejecutarla bastará mantener pulsado el botón LF y encender la impresora. Liberándolo abandonamos este modo.

Las modificaciones del ancho de los caracteres, posibilidad común a las tres impresoras comentadas, adquiere en la STAR toda su dimensión, pues pueden combinarse los distintos tipos entre ellos, proporcionándonos una gama francamente amplia.

En el formato estándar (pica), son impresos 10 caracteres por pulgada, hasta un total de 80 por línea y a una velocidad muy superior a la de sus competidoras: ¡160 caracteres por segundo!, que suponen una media de aproximadamente 120 considerando los retornos de carro y los saltos de línea.

Pero empleando modo ELITE ampliamos el número de columnas a 96, y si todavía no estamos satisfechos, disponemos de 136 en escritura condensada. Los tres tipos anteriores pueden combinarse con el modo EXPANDIDO. Si este juego de caracteres no nos contenta, basta pasar al itálico para disfrutar de otras seis combinaciones básicas.

La SG-10 dispone de por sí de una buena densi-



dad gráfica (matriz estándar de 9×11 puntos), pero si le exigimos todavía mayor calidad, podemos recurrir a la escritura enfatizada y a la doble impresión, incluso a ambas simultáneamente, además de subrayado, caracteres superescritos y subscriptos.

Ocho juegos de caracteres internacionales, entre ellos el español, se encuentran almacenados en la ROM del Sistema, pudiendo ser intercambiados actuando sobre los *dip switches* o enviando el código de control específico de cada uno.

Es factible manejar macrocódigos de control, es decir, en una sola instrucción podemos especificar varios de uso corriente de la misma forma que definimos subrutinas en BASIC para los trabajos repetitivos.

Pero cuando realmente se aprecian las formidables virtudes del equipo es pasando del modo texto al gráfico. En principio, podemos definir nuestros propios caracteres (hasta 240 diferentes) quedando almacenados en la RAM de la impresora, si los 597 contenidos en la ROM no nos satisfacen.

Otras muchas labores como la confección de gráficos en alta resolución, manejo de la impresora como plotter, confección de logotipos, etc., pueden efectuarse con la calidad como norma. En la sección de programa se da un ejemplo concreto del manejo de los gráficos definidos.

El manual es excelente, pero está escrito en inglés y llegar a dominar un equipo de esta envergadura si no conocemos este idioma puede resultar bastante problemático.

---

---

## MANEJABLE Y COMODA

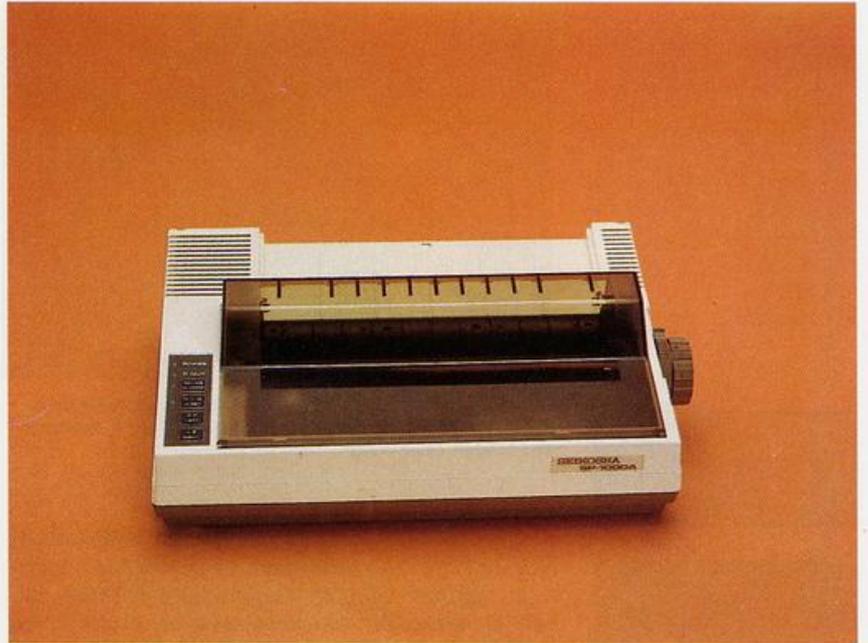
---

---

La SEIKOSHA SP-1000 sigue la línea emprendida por esta firma tras la aparición en el mercado de la SP-800, pero aventajando a esta última en la posibilidad de construir hasta 96 caracteres programables en la memoria RAM de la impresora.

Es posible trabajar en los dos modos de arrastre del papel, fricción y tracción, siendo un detalle significativo el cargador automático de papel, el cual permite cuando nos encontramos en la modalidad hoja a hoja, continuar con el proceso de impresión sin necesidad de detenerlo para efectuar el cambio.

Si el pomo de carga colocado a la derecha de la unidad está girado hacia delante, cuando sea detectado un final de papel, el rodillo de fricción girará hasta colocar la siguiente hoja en posición de trabajo.



*La SEIKOSHA SP-1000 tiene la posibilidad de construir hasta 96 caracteres programables en la RAM de la impresora.*

La cinta gráfica es continua, encontrándose alojada en el interior de un cartucho que le sirve de protección. Una vez instalada podemos hacer uso de la función de autochequeo y a la par regular el cabezal mediante el conmutador de 7 posiciones habilitado al efecto, hasta conseguir la mejor calidad de impresión sobre el papel elegido. Los pulsadores del panel de control tienen sus

*La principal virtud de la ASTRON 1400 es su velocidad.*





Para las pruebas efectuadas, así como el programa de la próxima sección, se ha empleado un interface Centronics de la firma Indescomp.



La ventaja de las impresoras que se analizan es que no sólo son aplicables al Spectrum, sino a multitud de equipos diferentes.



La cinta que emplea la Star SG-10 es la típica de cualquier máquina de escribir: monocroma de dos rollos, lo cual supone una economía a la hora del mantenimiento de la impresora.



La impresora SEIKOSHA SP-1000 incorpora entre sus curiosas características la de realizar automáticamente listados de códigos hexadecimales; algo muy útil, como bien sabemos los programadores de código máquina.

funciones duplicadas, es decir, además de las habituales ON LINE (en línea), FF (Form Feed o salto de página) y LF (Line Feed o salto de línea), caben la selección de caracteres en alta calidad NLQ (Near Quality Letter) o la posibilidad de pasar al modo margen (M MODE), en el cual, tanto el izquierdo como el derecho pueden elegirse a nuestro antojo.

Otra posibilidad es la de obtener listados hexadecimales de los datos recibidos por la impresora. Para ello, basta con mantener pulsado el interruptor FF mientras conectamos la alimentación. En este modo son impresos 32 pares de datos hexadecimales por línea.

El cabezal de impresión, de nueve agujas, está capacitado para imprimir 96 caracteres estándar más otros 32 correspondientes a 11 juegos de caracteres internacionales directamente seleccionables por el usuario.

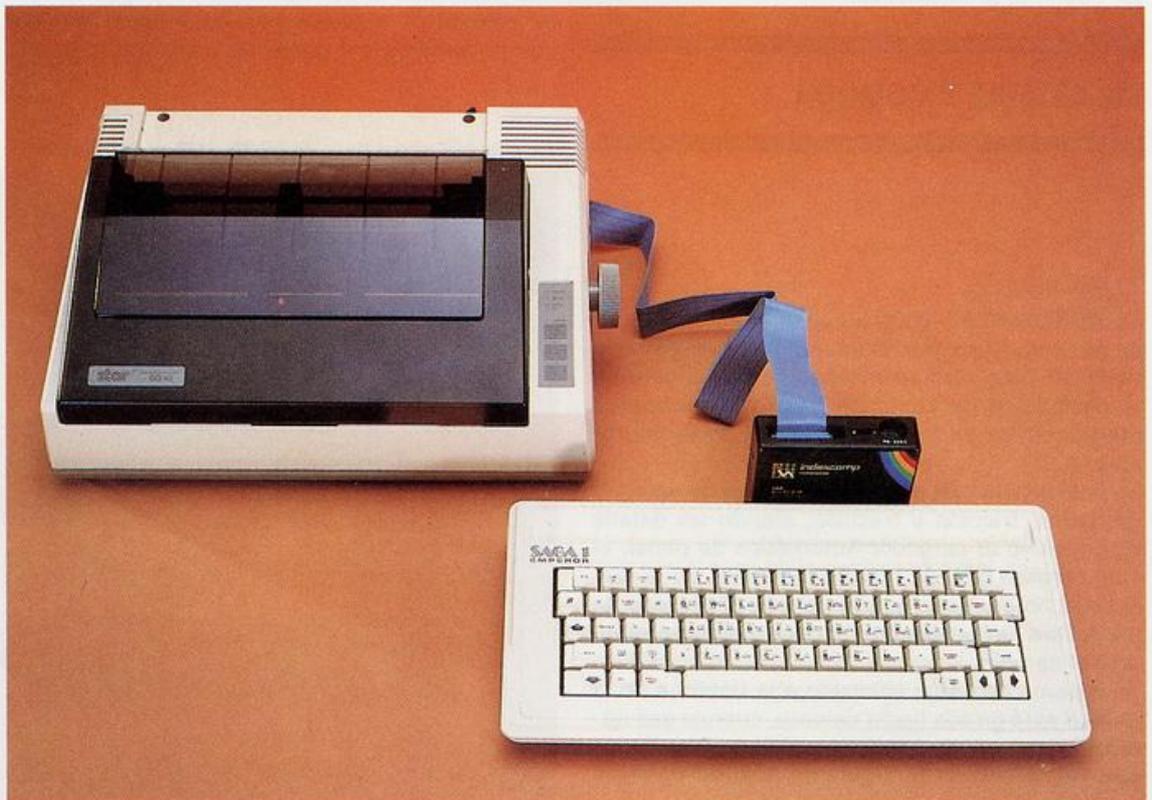
Quizá los más exigentes encuentren algo deficiente la calidad de letra en el modo normal. Pero puede ser solventada en gran medida, efectuando la impresión con el interruptor NLQ conectado. Los tipos de impresión siguen las prioridades que se indican a continuación, ordenados en dos grupos y de mayor a menor influencia:

1. Elite, proporcional, enfatizado, comprimido y pica o estándar.
2. Exponente/subíndice, alta calidad y doble pasada.

Por tanto, deberemos tenerlas presentes, pues al activar varios tipos a la vez, quizá recibamos la sorpresa de que el resultado no es el esperado al no haber anulado otro tipo preferencial.

En modo gráfico se pueden imprimir hasta nueve puntos verticales y 1920 por línea. Situando el dip switch 2-4 en posición ON accedemos a la RAM programable donde como hemos señalado, tienen cabida hasta 96 caracteres definidos por el usuario. Para terminar señalar que aunque el manual está escrito en lengua inglesa, se acompaña a la impresora de uno resumido en castellano.

*Una impresora del tipo de las mencionadas, combinada con un teclado profesional, pueden hacer de nuestro Spectrum un ordenador «nuevo».*



## SOBRIA, PERO EFICIENTE

La ASTRON 1400, distribuida por MICROBYTE, quizá en su aspecto exterior y prestaciones no resulte tan espectacular como las anteriores, pero su eficiencia trabajando queda fuera de toda duda.

Su principal virtud es la velocidad (alrededor de 140 caracteres en modo pica, por segundo). Pero también puede convertirse en su peor enemigo, pues si no colocamos el papel cuidadosamente en la posición correcta surgen inevitablemente problemas de desviación.

Igualmente, es necesario prestar atención a la introducción del pequeño cartucho que contiene la cinta rodeando el cabezal de impresión, pues un error en este sentido puede provocar una siempre molesta avería.

Se puede trabajar indistintamente en modo fricción o tracción, aunque si lo vamos a hacer con frecuencia en el segundo, es aconsejable proveernos de la unidad de tracción opcional, pues-

to que las pequeñas ruedas dentadas cumplen a duras penas su cometido.

Los *dip switches* permiten seleccionar entre cuatro juegos de caracteres internacionales, español entre ellos. Además, en virtud de la posición del microconmutador 7, es posible alterar el diseño del cero (con barra o sin ella), útil sin duda, para evitar molestas confusiones en los listados.

Varios modelos de letra pueden combinarse entre sí hasta conseguir una calidad que sin ser demasiado elevada resulta aceptable. Caracteres estándar, ensanchados, comprimidos, enfatizados, doble impresión, subrayados son algunos de los diferentes tipos disponibles.

En este equipo, no podemos definir caracteres gráficos, pues carece de memoria RAM, aunque sí es factible pasar del modo texto al gráfico. En este último destaca el denominado 1:1, en el cual las escalas horizontal y vertical son iguales.

El manual está en castellano, lo cual es un alivio, aunque en muchas de sus zonas las explicaciones se quedan realmente cortas y, resumido diremos que se trata de una impresora destinada a aquellos que pretendan rápidos listados sin gran alarde gráfico.



*Las tres impresoras mencionadas se suministran inicialmente con interface paralelo, aunque existen versiones en serie.*

**i!**

En general, las tres impresoras tratadas pueden ser consideradas de un bajo coste, en relación con sus grandes prestaciones.

\*

En la impresora ASTRON 1400 es necesario prestar atención en la introducción del cartucho que contiene la cinta rodeando el cabezal de impresión.

\*

Los mecanismos de fricción y tracción nos permiten realizar la salida de documentos impresos tanto en forma de típicos listados, con papel «pijama» o blanco continuo (*fan-fold*), como con hojas sueltas, en muy diversos formatos: folio, DIN A 4, etc...





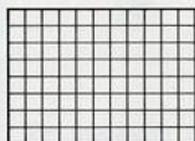
## POSTERS



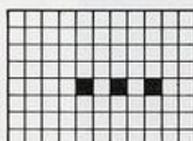
cuántas veces hemos quedado absortos observando una pantalla de **Ultimate** o **U.S. Gold** mientras carga nuestro **Spectrum**. Efectivamente, esas minúsculas «obras de arte» llaman la atención de cualquiera que tiene la oportunidad de verlas, y más aún de aquellos que, introducidos en la programación del **Spectrum**, sabemos de la complejidad inherente a conseguir tales efectos plásticos con nuestro aparato.

Por ello, la mayoría de los embobados admiradores de estos cuadros electrónicos, sentimos la inmediata tentación de dejar constancia por escrito de los mismos, porque como de todos es conocido: «Las pantallas se las lleva el **CLS**, pero lo escrito, escrito está».

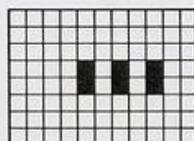
*Tonos de grises definidos para el programa POSTERS.*



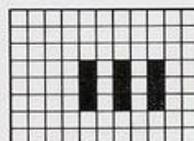
BLANCO BRILLO



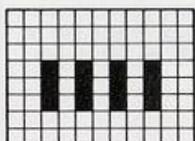
BLANCO MATE



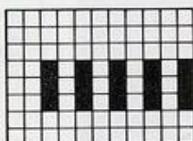
AMARILLO BRILLO



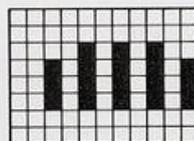
AMARILLO MATE



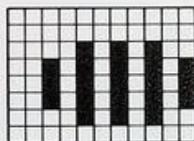
CYAN BRILLO



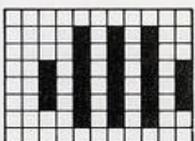
CYAN MATE



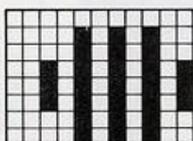
VERDE BRILLO



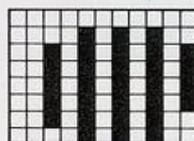
VERDE MATE



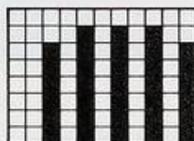
MAGENTA BRILLO



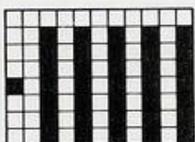
MAGENTA MATE



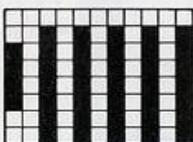
ROJO BRILLO



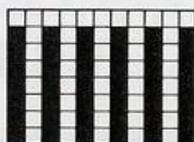
ROJO MATE



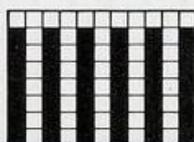
AZUL BRILLO



AZUL MATE



NEGRO BRILLO



NEGRO MATE

## PANTALLAS DE PARED

El programa que presentamos a continuación tiene una doble misión: por una parte, servirá para ilustrar el comentario que sobre impresoras ha venido desarrollándose en los dos últimos capítulos de **TU SPECTRUM**; por otra, hará realidad el pequeño sueño de poner en la pared toda una pantalla de televisor ¡con sólo cuatro chinchetas! Ahora, eso sí, por necesidades obvias, los posters generados serán en blanco y negro, pero como salidos de un televisor de 26 pulgadas, pues sus dimensiones son aproximadamente ¡56x48 cm! El secreto del programa es la lectura de las dos zonas de pantalla de **Spectrum**: archivo de imagen y atributos, no sólo como un simple **COPY**, que se encarga únicamente de representar la primera de ellas. Así, al averiguar el color en que se encuentra cada punto de la pantalla (no teniendo en cuenta el atributo **FLASH** pero sí el **BRIGHT**), se selecciona un carácter definido que genera el correspondiente tono de gris.

## ADOPCION DEL PROGRAMA

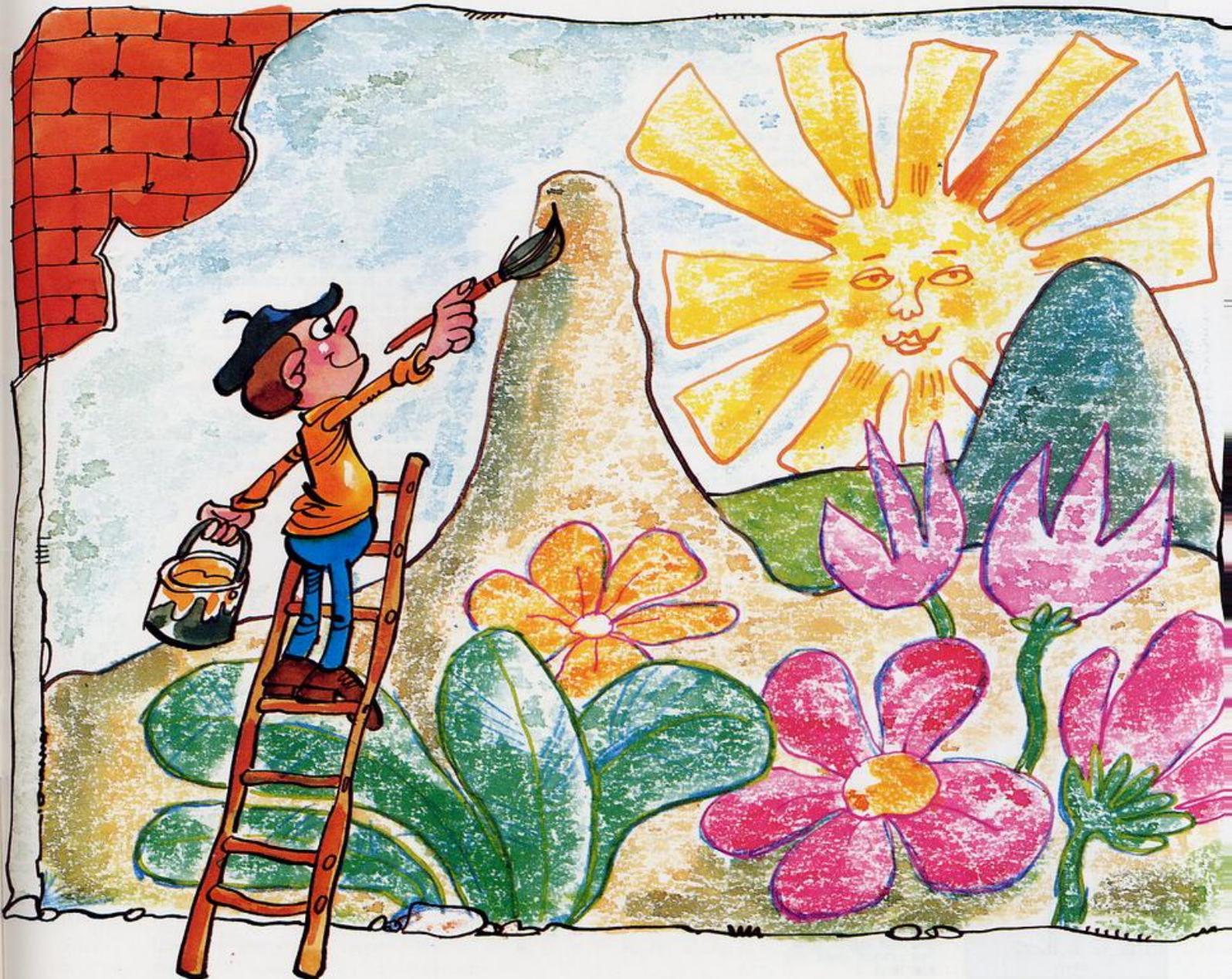
Aunque la filosofía del programa es prácticamente idéntica para cualquier impresora, existen pequeñas variaciones acorde con el sistema particular de definición de caracteres de cada impresora. En todo caso, hemos realizado dos versiones para la **SEIKOSHA SP-1000 A** y **STAR SG-10**. En primer lugar deberemos introducir el listado de inicialización correspondiente a la impresora



de que dispongamos, a continuación la zona común a ambas versiones, que contiene la definición de las diferentes tonalidades de grises, y por último, la zona de listado correspondiente a 176 ó 192 líneas, que utilizaremos para obtener un copiado de la pantalla completa (segundo caso),

o sin incluir el contenido de las dos últimas líneas reservadas al Sistema.

Una vez hecho ésto, no tenemos más que grabar el producto final conseguido y a copiar. El poster aparecerá en cuatro tiras que al concluir debemos componer: ¡pegar y colgar!





## INICIALIZACION SEIKOSHA

```

10 REM POSTER $ F. LOPEZ MARTINEZ & C. DE LA OSSA $
SEIKOSHA SP-1000A
20 REM CARGA DE ORIGINAL POSTER
30 LOAD *'M':1;'PANTALLA'SCREEN$
40 REM INICIALIZACION INTERFACE INDESCOMP
50 RANDOMIZE USR 64973
60 REM ESPACIO COMPENSADO Y ESPACIADO ENTRE LINEAS
21/216 PULGADAS
70 LPRINT CHR$ 27;'p1';
80 LPRINT CHR$ 27;'3';CHR$ 21;
90 REM SELECCIONA JUEGO DE CARACTERES DEFINIDOS
100 REM !EL SWITCH 2-4 DEBE ESTAR ON!
110 LPRINT CHR$ 27;'%'CHR$ 1;CHR$ 0;
120 REM CARACTERES A DEFINIR 65 ('A') HASTA 80 ('P')
130 LPRINT CHR$ 27;'&';CHR$ 0;CHR$ 65;CHR$ 80;
140 REM DEFINICION DE LOS CARACTERES
150 FOR I=0 TO 15
160 REM ATRIBUTO=10 (BIN 0000 1010)
170 LPRINT CHR$ 10;
180 FOR J=0 TO 10
190 READ A
200 LPRINT CHR$ A;
210 NEXT J
220 NEXT I

```

## INICIALIZACION STAR

```

10 REM POSTER $ F. LOPEZ MARTINEZ & C. DE LA OSSA $
STAR SG-10
20 REM CARGA DE ORIGINAL POSTER
30 LOAD *'M':1;'PANTALLA'SCREEN$
40 REM INICIALIZACION INTERFACE INDESCOMP
50 RANDOMIZE USR 64973
60 REM ESPACIO COMPENSADO Y ESPACIADO ENTRE LINEAS
14/144 PULGADAS
70 LPRINT CHR$ 27;'p1';
80 LPRINT CHR$ 27;'3';CHR$ 14;
90 REM SELECCIONA JUEGO DE CARACTERES DEFINIDOS
100 REM !EL SWITCH 1-5 DEBE ESTAR OFF!
110 LPRINT CHR$ 27;'#1';
120 REM CARACTERES A DEFINIR 65 ('A') HASTA 80 ('P')
130 LPRINT CHR$ 27;'*';CHR$ 1;CHR$ 65;CHR$ 80;
140 REM DEFINICION DE LOS CARACTERES
150 FOR I=0 TO 15
160 REM ATRIBUTO=11 (BIN 0000 1011)
170 LPRINT CHR$ 11;
180 FOR J=0 TO 10
190 READ A
200 LPRINT CHR$ A;
210 NEXT J
220 NEXT I

```

# !

Por cada punto de la pantalla se genera en el programa un carácter completo en el papel, con el correspondiente tono de gris.

# \*

Las diversas zonas del programa pueden ser grabadas separadamente, y posteriormente compuestas mediante MERGE.

# \*

Según la velocidad de la impresora, el poster tardará más o menos tiempo en estar listo para colgar. El proceso es realmente muy lento, aunque el resultado final sin duda merece la pena.

## IMPRESION DE 192 LINEAS

```

550 REM IMPRESION POSTER
560 LET D=0: LPRINT : LPRINT
570 FOR Z=16384 TO 16408 STEP 8: LET L=0
580 FOR J=Z TO Z+4096 STEP 2048
590 FOR I=J TO J+224 STEP 32
600 GO SUB 1000
610 FOR N=I TO I+2047 STEP 256
620 LET V=1
630 FOR K=N TO N+7: POKE 16384,PEEK K
670 FOR Q=0 TO 7: LET S=(VAL B$(3*V-1) AND NOT POINT
(Q,175))+(VAL B$(3*V) AND POINT (Q,175))
680 LET G=65+S*2+VAL B$(3*V-2): LPRINT CHR$ G; NEXT
Q
690 LET V=V+1
700 NEXT K: LPRINT : NEXT N
710 LET L=L+1
720 NEXT I: NEXT J
730 LPRINT CHR$ 12;
740 LET D=D+8: NEXT Z
750 STOP
1000 REM CALCULO DEL ATRIBUTO
1010 LET B$=""
1020 FOR C=D TO D+7
1030 LET A=ATTR (L,C)
1040 LET A=A-(128 AND A>127)
1050 LET BR=0: IF A>63 THEN LET BR=1: LET A=A-64
1060 LET F=INT (A/8)
1070 LET T=A-8*F
1080 LET B$=B$+STR$ BR+STR$ F+STR$ T
1090 NEXT C
1100 RETURN

```

## DEFINICION DE TONOS GRISES

```

230 REM NEGRO MATE
240 DATA 127,0,127,0,127,0,127,0,127,0,127
250 REM NEGRO BRILLO
260 DATA 127,0,127,0,127,0,127,0,127,0,127
270 REM AZUL MATE
280 DATA 60,0,127,0,127,0,127,0,127,0,127
290 REM AZUL BRILLO
300 DATA 0,0,127,0,127,0,127,0,127,0,127
310 REM ROJO MATE
320 DATA 0,0,63,0,127,0,127,0,127,0,63
330 REM ROJO BRILLO
340 DATA 0,0,62,0,127,0,127,0,127,0,60
350 REM MAGENTA MATE
360 DATA 0,0,28,0,127,0,127,0,127,0,28
370 REM MAGENTA BRILLO
380 DATA 0,0,28,0,126,0,126,0,126,0,28
390 REM VERDE MATE
400 DATA 0,0,28,0,62,0,62,0,62,0,28
410 REM VERDE BRILLO
420 DATA 0,0,28,0,60,0,60,0,60,0,28
430 REM CIAN MATE
440 DATA 0,0,28,0,28,0,28,0,28,0,28
450 REM CIAN BRILLO
460 DATA 0,0,28,0,28,0,28,0,28,0,0
470 REM AMARILLO MATE
480 DATA 0,0,0,0,28,0,28,0,28,0,0
490 REM AMARILLO BRILLO
500 DATA 0,0,0,0,24,0,24,0,24,0,0
510 REM BLANCO MATE
520 DATA 0,0,0,0,8,0,8,0,8,0,0
530 REM BLANCO BRILLO
540 DATA 0,0,0,0,0,0,0,0,0,0

```

## IMPRESION DE 176 LINEAS

```

550 REM IMPRESION POSTER
560 LPRINT : LPRINT
570 REM ZONAS (COLUMNS 0-63-127-191-255)
580 REM I=COMIENZO DE ZONAS
590 FOR I=0 TO 192 STEP 64
600 REM J=LINEAS
610 FOR J=175 TO 0 STEP -1
620 REM K=COLUMNAS
630 FOR K=I TO I+63
640 REM A=ATRIBUTO DE PUNTO
650 LET A=ATTR (INT ((175-J)/8),INT (K/8))
660 REM F=FONDO, T=TINTA, B=BRILLO (ELIMINA FLASH)
670 LET A=A-(128 AND A>127)
680 LET B=0: IF A>63 THEN LET B=1: LET A=A-64
690 LET F=INT (A/8)
700 LET T=A-8*F
710 REM SELECCION DE GRIS A IMPRIMIR
720 LET C=(F AND NOT POINT (K,J))+(T AND POINT (K,J))
730 LET G=65+C*2+B
740 LPRINT CHR$ G;
750 NEXT K
760 LPRINT
770 NEXT J
780 REM NUEVA ZONA
790 LPRINT CHR$ 12;
800 NEXT I

```