Quinta Parte

Usar el Spectrum hasta el límite

CAPÍTULO DÉCIMO

¡Refuerce sus programas!

MEJORAR LA PROGRAMACIÓN

Una de las cosas más esenciales que se necesitan para programar bien es pensar de un modo lógico y estructurado en el problema que se nos presenta. También debe procurar hacer los programas inteligibles para los demás (a menos que tenga una razón específica para no hacerlo), porque así también serán claros para usted cuando vuelva a ellos después de un tiempo.

Estructurar el programa claramente le ayudará mucho. Si la capacidad de memoria se lo permite, no escatime sentencias REM para clarificar el programa, dividiéndolo en partes e indicando qué trabajo efectúan cada una de ellas. Para hacerlo más fácil de leer puede usar REMs en blanco alrededor de los verdaderos REMs. Si el espacio en memoria se lo permite puede incluir líneas en blanco que también ayudarán a una mejor lectura del programa. El Spectrum acepta líneas en blanco escribiendo el número de línea seguido al menos de un espacio. Esto coloca agujeros en el listado haciéndolo más fácil de leer pero sin gastar demasiada memoria:

```
10 LET A = 4,87
20
30 A < B THEN PRINT "HA PERDIDO"
```

En el ejemplo de arriba, la línea 20, que está vacía, nos marca la diferencia entre el programa y la rutina que comprueba si el jugador ha perdido.

El Spectrum también tiene la particularidad de poder colorear el listado sin que esto tenga efecto cuando el

programa es ejecutado. Recuerde que esto se hace poniendo el cursor en modo E y pulsando la tecla del color correspondiente.

Tenga cuidado cuando quiera listar programas por la impresora pues ésta puede producir problemas al intentar leer estos caracteres de control.

Si el programa contiene un cierto número de rutinas que se deben ejecutar una sola vez entonces hágalo al principio del programa. Coloque las subrutinas al final del programa y acceda a ellas desde las primeras líneas poniendo un REM antes del GOSUB que indique lo que va a hacer:

```
10 REM CREACIÓN DE UN CARÁCTER DE
INVASOR
20 GOSUB 9000
```

El Spectrum también permite dar nombres completos a las variables numéricas, y este hecho se puede usar para crear un BASIC semiestructurado en su ordenador. Si usted asigna a una variable llamada PRUEBA un número que no es más que el número de línea donde ocurre la PRUEBA, entonces podrá decir GOTO PRUEBA o GOSUB PRUEBA en cualquier parte del programa. Esto proporciona ventajas similares a las "procedures" de otros ordenadores (que permiten definir un proceso y llamarlo desde cualquier punto del programa). Observe que asignando números de línea a las variables, hará mucho más fácil la tarea de mirar un área específica de su programa. Por ejemplo, si cada subrutina de su programa tiene un nombre y este nombre se usa como variable conteniendo el número de línea donde empieza la subrutina, entonces simplemente haciendo LIST nombre tendrá la rutina en la pantalla. Por ejemplo:

```
10 LET PUNTUACIÓN = 1000
1000 REM RUTINA DEL CÁLCULO DE LA
PUNTUACIÓN
```

Así, LIST PUNTUACIÓN, listará de la línea 1000 en adelante. Esto también tiene la ventaja de que si usted usa una rutina de renumeración que deja los mismos números en los GOTO y GOSUB, solamente necesitará cambiar

estas variables al principio del programa en lugar de tener que buscar todos los GOSUBs y GOTOs.

También puede hacer que todos los procedimientos similares aparezcan en el listado del mismo color, las líneas de datos, de otro color, etc.

Las variables que use con mucha frecuencia, defínalas al principio del programa. Esto lo digo porque cuando el Spectrum encuentra una variable la busca empezando desde el principio del programa hasta que la encuentra. El programa irá por lo tanto más rápido, si la variable se encuentra enseguida.

Piénselo un poco antes de hacer la parte visual de su programa. Muchos programas excelentes pierden mucho debido a una pobre presentación gráfica, textos mal centrados, colores que no ligan, etc. Si durante el programa el usuario tiene que leer un texto, use PAUSE 0 para que pueda hacerlo durante el rato que desee, hasta pulsar una tecla.

Pruebe exhaustivamente sus programas, pero no sólo introduciendo lo que ya cabe esperar sino todo lo que se le ocurra. Controle las entradas de datos mediante las rutinas que ya se explicaron en su momento. No use preguntas ambiguas que pueden no significar lo mismo para usted que para los demás.

RENUMERACIÓN DE LÍNEAS

Una vez terminado un programa, rara vez quedan las líneas a la misma distancia (los números, claro) sino que hay áreas en las que se han tenido que añadir líneas y sus números sólo difieren en una o dos unidades, mientras que en otros lugares se han borrado, con el consiguiente desequilibrio en la numeración. Lo peor de todo esto no es el desorden, sino que a veces puede suceder que no disponga de más números para colocar en un área del programa que necesita ser desarrollada. La solución está en una rutina de renumeración, que lo que hace es renumerar el programa empezando a partir de un número de línea dado y con un paso marcado. Le sugiero que grabe este programa en cinta con estos mismos números de líneas que contiene, y así cada vez que termine un programa puede recuperarlo con MERGE:

```
9990 RENUMBER
9991 LET X = PEEK 23635 + 256*PEEK 23636
9992 LET FLN = 10: Primer n.º de línea
9993 LET LNS = 10: Paso entre líneas
9994 IF PEEK (X + 1) + 256*PEEK X = 9990 THEN
     STOP
9995 POKE X, INT(FLN/256)
9996 POKE X + 1,FLN - 256*INT (FLN/256)
9997 LET FLN = FLN + LNS
9998 LET X = X + 4 + PEEK(X + 2) + 256*PEEK
     (X + 3)
9999 GOTO 9994
```

CLEAR Y RESTORE

Si tiene un programa que ya contiene sus propias variables y no quiere que se mezclen con las de un programa anterior (si ha usado MERGE), o con las de él mismo si es la segunda vez que lo ejecuta, entonces empiécelo con esta línea:

```
1 CLEAR: RESTORE
```

VELOCIDAD

Para conseguir que sus programas se ejecuten a la máxima velocidad, intente usar el menor número de GOTOs posible y reserve los GOSUBs para las primeras líneas. El uso de números directamente en lugar de variables también aumentará la velocidad de ejecución. Si tiene una lista de condiciones de modo que si una es falsa, las otras no necesitan ser comprobadas; reemplácelo:

```
10 IF A < > 1 AND A < > 2 AND A < > 3 AND . . .
```

por:

```
10 IF A < > 1 THEN IF A < > 2
   THEN IF A < > 3 THEN . . .
```

La segunda versión irá a la línea siguiente en cuanto no se cumpla una condición, mientras que en la primera se evaluarán todas.

Cuando tenga que entrar largas listas de datos, le recomiendo que antes haga el siguiente POKE para cambiar el "clic" del teclado con objeto de que se oiga más. De este modo reducirá bastante los errores:

POKE 23609, 100

También se puede cambiar el tiempo que tarda una tecla en repetirse, y el tiempo que pasa entre dos repeticiones. Esto ayudará cuando tenga que mover el cursor por largas líneas de datos.

TIEMPO PREVIO A LA REPETICIÓN:
POKE 23561,x (pruebe con x = 10)

TIEMPO ENTRE REPETICIONES:
POKE 23562,x (x = es demasiado rápido,
x = 3 es más manejable)