CURSO AUTODIDACTICO DE BASIC I

AMSTRAD

CPC-464

AMSTRAD

BASIC

Curso Autodidáctico de Basic I

Editado por **INDESCOMP, S.A.**
Avda. Mediterráneo, 9 - 28007 MADRID (ESPAÑA)

Traduce, compone e imprime: **CONORG, S.A.**

I.S.B.N.: 84-86176-18-2

Déposito Legal: M-12931-1985

AMSTRAD
BASIC

Curso Autodidáctico de Basic I

Parte 1
PARA PRINCIPIANTES

First edition 1984

Amstrad Consumer Electronics plc
Brentwood House
169 Kings Road
Brentwood
Essex

Amstrad BASIC

A Tutorial Guide
Part 1: First Steps

SOFT 111 ISBN 1 85084 000 8

- Programming by Dave Atherton
- Written by Dave Collier and George Tappenden
- Production by Peter Hill and Ray Smith
- Printed in England by Horwood Printers

CONTENIDOS

PREFACIO

Esta es la Parte 1 de un curso autodidáctico sobre programación en BASIC usando el ordenador personal a color Amstrad CPC464. Las dos cassettes con programas que acompañan a este texto, forman parte integral del curso.

La cassette de datos A contiene:

- Programas que ayudan a explicar los principios de programación simple y de entretenimiento.
- Juegos para tu diversión que ayudan a familiarizarte con tu CPC464.

La cassette de datos B contiene:

- Auto-evaluaciones para que puedas cerciorarte que has comprendido los conceptos descritos en cada capítulo.

Los fundamentos de la programación avanzada se cubren en la Parte 2 de este curso, **Más sobre BASIC.**

1 ¿DE QUE SE TRATA?

Si estás realizando este curso, eres casi con toda certeza el orgulloso propietario de un microordenador Amstrad CPC464. Su soberbia imagen y excelente calidad de sonido te habrán abierto ya un nuevo y excitante mundo de diversión y juego. Además, te habrás percatado que para hacer algo más que jugar con los programas standard, necesitas aprender el lenguaje del CPC464: BASIC.

BASIC

BASIC es el lenguaje de programación más popular del mundo, también es el lenguaje ideal para el principiante, dado que es posible escribir el primer programa después de unas pocas horas de estudio. La satisfacción que se obtiene con esto es enorme dado que, y no hay que darle vueltas, la programación es divertida en sí misma. Puede incluso convertirse en un pasatiempo.

Entonces ¿qué es programación? Bien, tienes que recordar que un ordenador puede hacer muchas cosas pero no puede pensar. Tendrás que suplirle en eso. Ese pensamiento -es decir, determinando lo que necesita hacerse para conseguir un objetivo- viene en la forma de una **serie de instrucciones** conocida como **programa.** Un programa puede hacer que se convierta en un juego de batallas en la galaxia, un auxiliar en la redacción de textos, o una máquina que cuida de tus asuntos contables.

Probablemente, habrás visto que los programas escritos específicamente para el CPC464 no funcionan en otros ordenadores. Es debido a que el BASIC Amstrad usado en el CPC464 permite muchos **comandos** y **funciones** que son únicas y no están disponibles en equipos menos avanzados.

El BASIC tiene su propio vocabulario, al igual que cualquier otro lenguaje. Este vocabulario se compone de **claves,** o palabras **reservadas** y estás a punto de aprender lo que son y lo que significan para el CPC464. Cada vez que en este curso se emplea una nueva **clave,** la mostramos en el margen izquierdo de manera que puedas fácilmente consultar el libro para refrescar tu memoria sobre cualquiera de ellas.

Cómo usar este libro

Cada capítulo de este libro representa aproximadamente el trabajo de una o dos tardes. Generalmente, contienen:

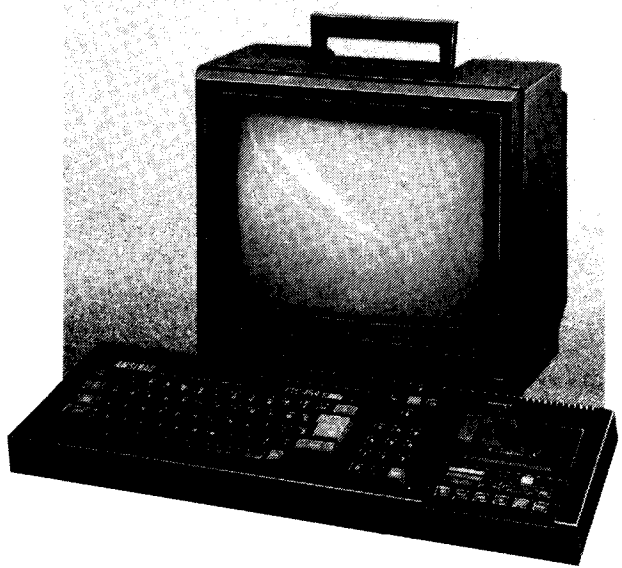
- Explicación escrita.
- Trabajo práctico con la máquina.
- Ejemplos para que tú mismo los programes.

Hay ejercicios que refuerzan lo que has aprendido, y en cada capítulo hay una **autoevaluación** para que compruebes el conocimiento adquirido.

No te 'saltes' capítulos. Progresivamente vamos presentando nueva información que se basa en lo aprendido en capítulos anteriores. Si crees que un capítulo o una sección de un capítulo, parece un poco complicada, léela rápidamente una o dos veces y luego trabaja con ella más minuciosamente. Asegúrate que la has comprendido, por medio de las autoevaluaciones.

Al concluir esta parte del curso, serás capaz de escribir programas simples y fiables para tus propios fines. La Parte 2 de este curso, **Más sobre BASIC,** explica las facultades avanzadas de Amstrad BASIC y te enseñará cómo escribir programas bastante más complicados.

2 INSTALACION Y PUESTA EN MARCHA



En principio, has tenido que desembalar tu CPC464. Si ya lo has hecho, te puedes saltar los siguientes párrafos.

Cuando abras las cajas, deberás encontrar los siguientes artículos:

- Ordenador personal a color CPC464
- Monitor GT64 - Monitor a color CTM640
- Alimentador-Modulador MP1 (opcional)
- **Guía del Usuario del Amstrad CPC464**
- Cassette de demostración

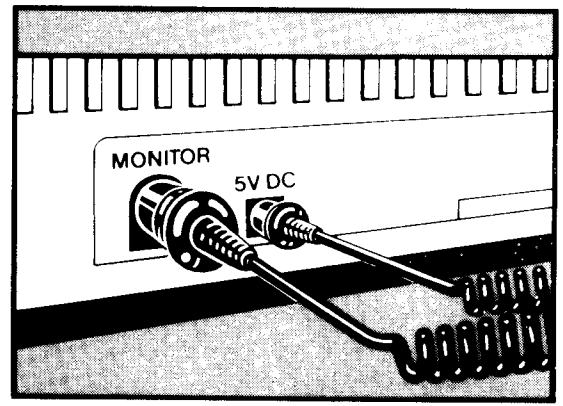
Busca una mesa o despacho razonablemente grande en una parte tranquila de la casa y coloca el CPC464 y su **monitor** (o el receptor de TV). Cuidadosamente, conecta las dos clavijas de la parte del frontal del monitor o del MP1 en sus zócalos del panel posterior del CPC464 (mira el diagrama). No conectes todavía el monitor o el MP1 en la toma de red.

Asegúrate que todas las clavijas están bien conectadas, pero no uses demasiada fuerza. Las clavijas y zócalos pueden dañarse si estás constantemente manipulándolos.

Si es posible, intenta encontrar un lugar permanente para tu CPC464; como mínimo, durante este curso.

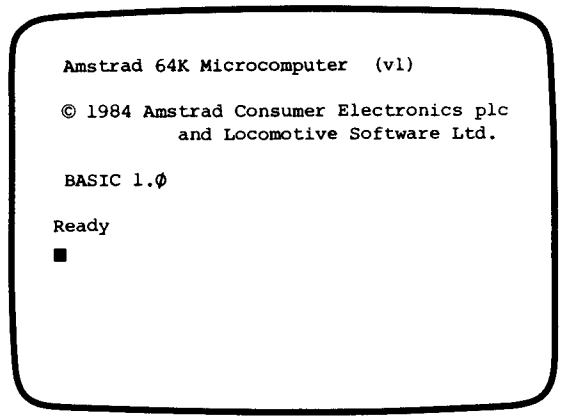
Si estás usando un receptor de TV con tu CPC464, cerciórate que dispones de dos enchufes de 13 amps, o usa un adaptador doble, de forma que puedas conectar el televisor y el MP1 al mismo tiempo. Luego conecta el cable coaxial del MP1 en la entrada de antena del receptor, y sintoniza el televisor al canal 36.

Acomódate en una buena silla, y luego coloca el monitor o el televisor a una distancia de 1 metro de tu cara; si trabajas demasiado cerca de una pantalla, puede ser muy cansado y puede producir fatiga ocular.



Coloca este libro de manera que puedas leerlo fácilmente mientras estás usando el teclado y también puedas observar la pantalla. También es conveniente colocar la Guía de Usuario Amstrad CPC464 a tu alcance.

Ahora conecta a la red y enciende. Deberá aparecer en pantalla lo siguiente:



Si no lo ves (las letras en amarillo sobre un fondo azul, si tienes un monitor de color CTM640 o un TV de color) apaga e inténtalo de nuevo. Si estás usando un televisor, comprueba que está sintonizado adecuadamente al canal 36. Si todavía sigues con dificultades, comprueba que el indicador ON del CPC464, se ilumina en rojo. Si no es así, verifica que:

- El interruptor ON/OFF del CPC464 está puesto (ON).
- No hay fallo general de energía.
- La clavija de alimentación está conectada firmemente en tu CPC464.
- El fusible de 13 amps incorporado en la clavija de red, está intacto.

Si todas estas pruebas fallan, ponte en contacto con tu distribuidor para otros consejos.

HOLA

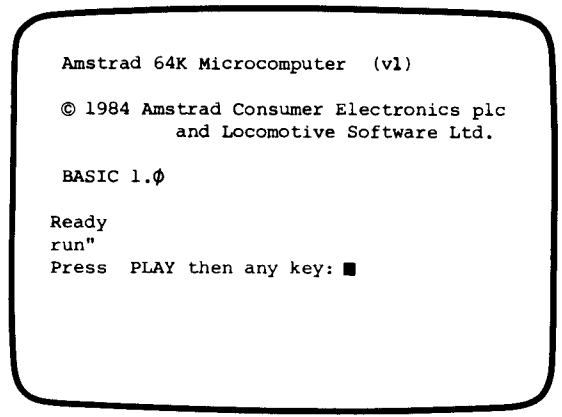
Una vez que veas el **saludo** en tu pantalla, puedes comenzar a funcionar. La palabra 'Ready' significa que el ordenador está 'dispuesto' para que le des **comandos** o **instrucciones.** El cuadradito que aparece se llama 'cursor', y es el que se mueve por la pantalla indicando el sitio donde aparecerá escrito lo que teclees.

Así que iniciemos el camino. Inserta el cassette "Cassette A" en la **lecto-grabadora** (datacorder) y teclea letra a letra:

run" y pulsa **ENTER**

Para teclear " has tenido que oprimir una de las teclas marcada SHIFT y mientras estaba presionada has apretado la tecla marcada **"** **2** .

Pulsando la tecla ENTER, indicas al CPC464 que has terminado de teclear una frase, y que la debe memorizar y obedecer lo que le acabas de mandar. Si has tecleado correctamente este **comando,** el CPC464 replicará añadiendo otra línea al mensaje que ya tienes en pantalla:



En el que te dice literalmente que: "Oprime PLAY luego cualquier tecla:", Asegúrate por lo tanto que la cinta está al principio (y aprieta el botón REW para rebobinar), luego haz lo que el ordenador te solicita en pantalla. Presiona PLAY y luego pulsa cualquier tecla, v.g. ENTER. Oirás un breve pitido del altavoz incorporado en el CPC464, ya que el **primer programa** está siendo leído de la lecto-grabadora y traspasado a la memoria, donde inmediatamente será "ejecutado".

RUN

Si cometes una equivocación mientras tecleas (y la detectas antes de pulsar ENTER), usa la tecla

DEL

Esta tecla **suprime** el último carácter tecleado y retrocede el cursor. Luego puedes teclear la letra o símbolo correcto.

Si detectas la equivocación después de pulsar ENTER, no te preocupes.

El CPC464 simplemente pondrá otra línea de texto en la pantalla, tal y como te mostramos en el siguiente ejemplo:

```
Amstrad 64K Microcomputer    (v1)

© 1984 Amstrad Consumer Electronics plc
          and Locomotive Software Ltd.

BASIC 1.ϕ

Ready
ren"
Syntax error
Ready
■
```

Has provocado un **'error sintáctico'**, y tendrás que comenzar de nuevo y volver a teclear el comando.

Pero como todo habrá ido bien, en breve aparecerá en pantalla:

```
Amstrad 64K Microcomputer    (v1)

© 1984 Amstrad Consumer Electronics plc
          and Locomotive Software Ltd.

BASIC 1.ϕ

Ready
run"
Press  PLAY then any key:
Loading HOLA block 1
```

que indica que está 'cargando' el **programa** llamado HOLA; que inmediatamente pasará a controlar la actuación del ordenador que -siguiendo sus **instrucciones**- te dará una bienvenida amistosa al mundo de los ordenadores.

Cuando hayas visto varias veces seguidas la actuación del programa HOLA, observa en la esquina superior izquierda del

teclado una tecla pintada de rojo. Es la tecla ESCape. Púlsala dos veces:

Con esta acción, se permite al ordenador que 'escape' del control del programa, lo que hace y te muestra el mensaje:

Break in 140
Ready
■

Puede que el número no sea 140, ni tampoco eso significa que tú o el CPC464 debáis esperar a que transcurran 140 segundos, minutos, horas o días. Ni mucho menos que irrumpas en la casa número 140 de tu calle. Cualquiera que sea el número, ignóralo. Ya aprenderás sobre **números de línea** más adelante. En cualquier caso, has obligado al CPC464 a provocar una **ruptura** en el seguimiento del programa HOLA, y así lo ha hecho; y ahora ya estás preparado para proseguir con el resto de este capítulo.

Aprovechando el inciso, observa que el CPC464 insiste en mostrar clara diferencia entre Ø (nada o cero) y O (como en 'hola') poniendo esa barra inclinada a través del circulito cuando **no es** una letra.

Juego número 1

Vamos a jugar un poco. Los juegos con ordenador puede que no sean para todo el mundo la mejor manera de emplear el tiempo y el dinero, pero hay en ellos mucho más de lo que parece a primera vista. Primordialmente, con ellos te familiarizas con la máquina sin el tedio de los ejercicios formales; y en segundo lugar pueden hacer que te des cuenta que los ordenadores son divertidos y didácticos.

Haremos algo diferente esta vez. Teclea en el CPC464 el siguiente **comando:**

LOAD

load"simon" ENTER

No olvides mantener presionada la tecla SHIFT para conseguir las comillas (**"**). El CPC464 contestará con el siguiente mensaje:

Press PLAY then any key:

Sigue otra vez las instrucciones. Oprime PLAY en la lecto-grabadora y luego pulsa una tecla cualquiera en el teclado principal. Si has escrito correctamente el nombre y todo está bien, el CPC464 comenzará a rodar la cinta en el cassette y a cargar un programa llamado SIMON. Los programas siempre tienen **nombres,** de manera que puedas distinguirlos unos de otros. Esta vez, el programa sólo será trasvasado a la memoria del CPC464, que no hará nada más hasta que se lo mandes.

El CPC464 mostrará el mensaje:

Loading SIMON block 1

y luego **cambiará** a:

Loading SIMON block 2

No te preocupes de lo que significa. Después de que se haya detenido el cassette, volverá a darte el **aviso** habitual:

Ready

Y en ese momento, tienes que ordenar al CPC464 lo que quieres hacer con el programa que está ahora alojado en la

memoria. Teclea:

run **ENTER**

Y diviértete.

Autoevaluación

Cuando te canses de jugar con SIMON, deten la ejecución pulsando dos veces la tecla ESCape como ya explicamos.

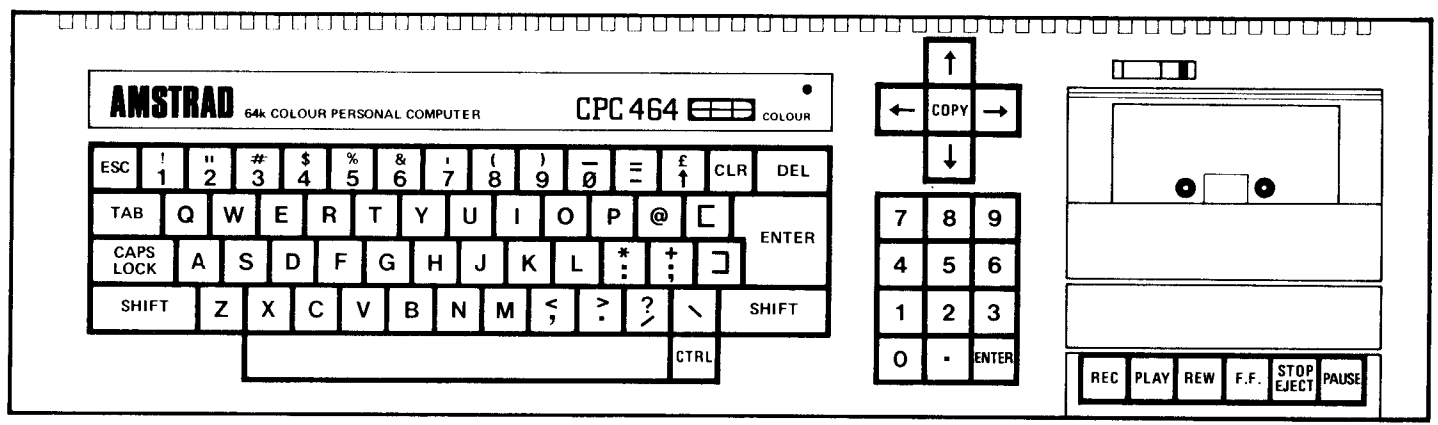
Tu primer "test" es ensayar el mismo método que usaste con SIMON, para cargar la primera de las autoevaluaciones TAE2, de la "cassette B". Cuando actúe este programa, te hará preguntas sobre este capítulo para que puedas ver si necesitas repasarlo.

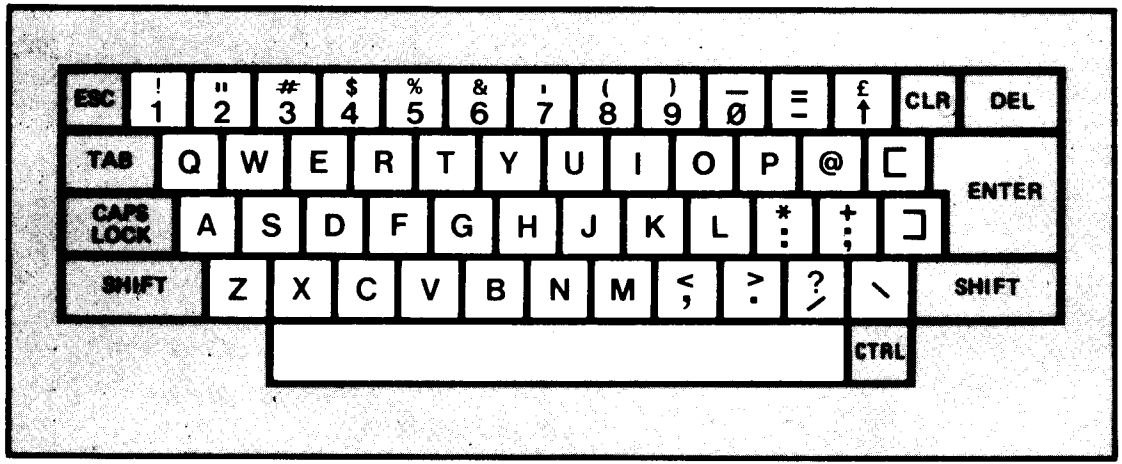
3 USO DEL TECLADO

Ya hemos dicho anteriormente que tendrás que aprender el lenguaje BASIC para comunicarte con el CPC464. Como te habrás dado cuenta, decirle al CPC464 lo que hay que hacer es teclearle palabras y números usando el teclado. Este capítulo trata de todo eso, aprendiendo las posiciones de las teclas y cómo pulsar las convenientes en el momento oportuno.

El CPC464 tiene cinco grupos separados de teclas:

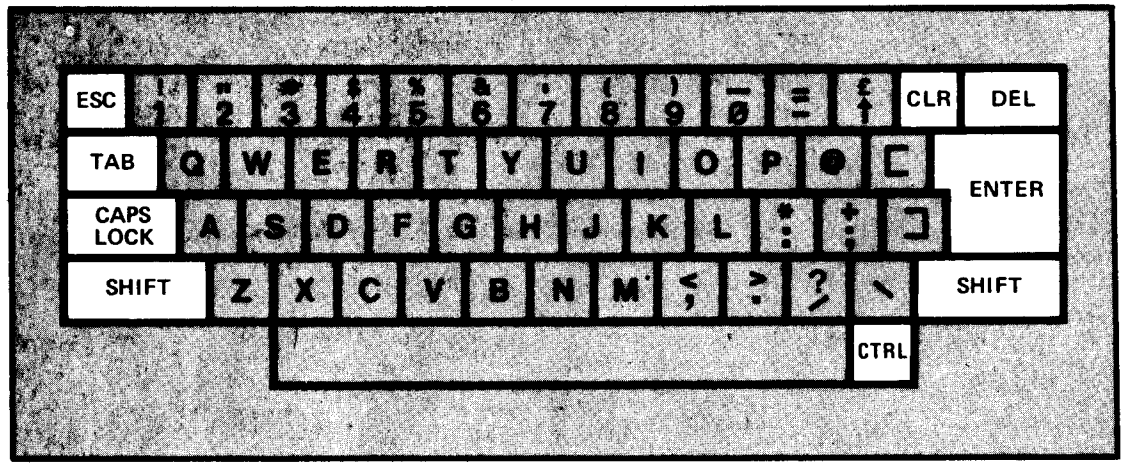
- Teclas de caracteres
- Teclas de control
- Tablero numérico
- Teclas de cursor
- Controles de la lecto-grabadora.





Teclado principal: teclas de caracteres

Ya las has estado usando. Si alguna vez usaste una máquina de escribir, no necesitarás más explicaciones. Esta parte del teclado consta de letras, cifras, un montón de signos de puntuación, que denominamos **"caracteres especiales"**, y una barra **'espaciadora'** para colocar espacios en blanco. Si mantienes pulsada cualquiera de estas teclas durante más de medio segundo, repetirá automáticamente la pulsación y continuará hasta que levantes el dedo.



Teclado principal: teclas de control

ESC

ESC corresponde a ESCape. Si la pulsas mientras está ejecutando un programa, detendrá al CPC464 en su seguimiento del programa, pero podrás hacer que lo reanude pulsando a continuación cualquier otra tecla. Pulsando por segunda vez la misma tecla ESC, hará que el CPC464 vuelva a la condición de 'preparado' para tus órdenes.

TAB

Si pulsas la tecla TAB verás una flecha apuntando hacia la derecha. Esta tecla no se usará en esta parte del curso, por lo que la explicaremos completamente en la parte 2.

SHIFT

SHIFT cambia el símbolo enviado al CPC464 cuando pulsas las teclas de caracteres. Manteniéndo pulsada SHIFT, conseguirás las letras mayúsculas al pulsar una tecla de letras, y el símbolo de la parte superior cuando pulses las teclas de números y de caracteres especiales.

CAPS LOCK

Pulsando CAPS LOCK queda el teclado **'enclavado** en mayúsculas', con lo que las letras aparecerán siempre así. Tiene casi el mismo efecto que mantener tu dedo sobre la tecla SHIFT; pero tienes que seguir pulsando dicha tecla SHIFT cuando desees enviar los símbolos marcados encima de los números o los caracteres especiales de la tecla.

CLR

La tecla CLR (sacada de la palabra CLeaR='clarear') es bastante similar a la tecla DEL. Se suprime un carácter, pero no el situado a la izquierda del cursor, como hace la tecla DEL. CLR "se come" el carácter situado precisamente debajo del cursor, que no se mueve y sí lo hacen todos los situados a la derecha del cursor.

DEL

Puedes recordar la tecla DEL del capítulo anterior. Cuando estás tecleando líneas, pulsando la tecla DEL se 'suprime' el carácter situado a la izquierda del cursor, que retrocede una posición.

ENTER

La tecla ENTER es parecida a la tecla de 'retorno de carro' de una máquina de escribir eléctrica. La has pulsado al final de cada línea que has tecleado en el CPC464 para hacerle saber que has terminado de mandarle algo. Normalmente, el cursor será desplazado desde el extremo de la última palabra o número tecleado, hasta el comienzo de la siguiente línea inferior. Algunas veces, el CPC464 también te dirá:

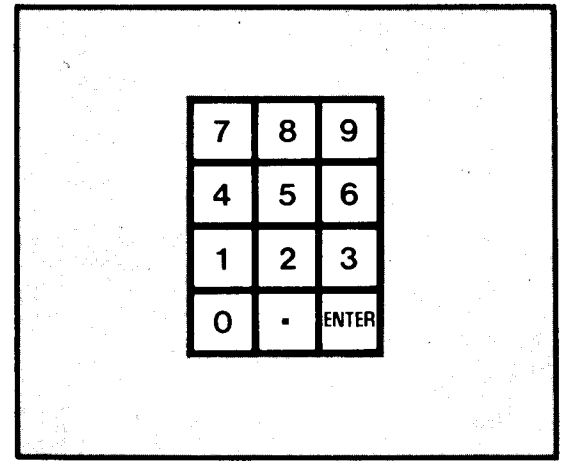
Syntax error

Esto es BASIC para decirte que **'mí** no comprender', y es uno de los llamados **mensajes de error.** Ya veremos más en el curso.

CTRL

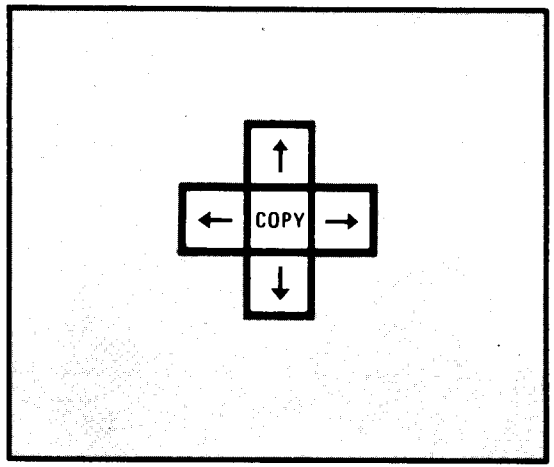
Corresponde a la palabra ConTRoL. Manteniéndola pulsada, conseguirás otro

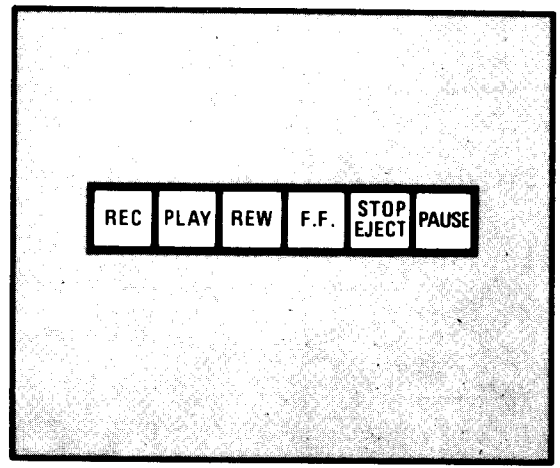
conjunto de símbolos al pulsar las teclas de letras y algunas de las teclas de cifras además de los que están marcados en la tecla. Cuando se usa con otras teclas de control, esta tecla CTRL ordena al CPC-464 que haga ciertas cosas -que ya veremos posteriormente.



Tablero numérico

Estas teclas están dispuestas convenientemente para teclear montones de números. Aparte de la tecla extra ENTER, son idénticas a las teclas de cifras que aparecen en la fila superior del teclado principal. Excepto en que a no ser que sean **programadas** especialmente, no se ven afectadas al pulsar simultáneamente SHIFT, o CTRL. Esta 'programación especial' no se comentará en esta parte del curso, pero en este mismo capítulo, verás cómo esta tecla ENTER puede tener una función extra.





Cursor

Las teclas de 'flechas' se usan para mover el cursor por toda la pantalla en la dirección correspondiente. La tecla COPY sirve para 'copiar' lo que hay en pantalla, como ya comentaremos más adelante.

Controles de la lecto-grabadora

Hay sólo una diferencia entre estos botones y los mandos de un magnetófono de cassettes de audio. Después de oprimir los botones PLAY (o bien PLAY y REC), el equipo no funcionará hasta que no le envíe las señales oportunas el CPC464.

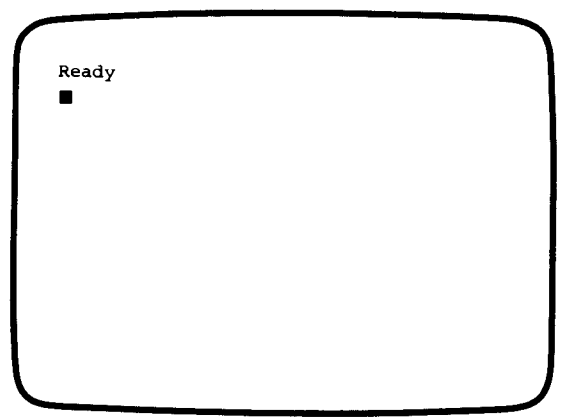
Trabajo práctico

Comprueba que la cassette está apropiadamente metida en la lecto-grabadora. Si todavía tienes restos del programa TAE2 en toda la pantalla, por tu trabajo en el capítulo anterior, puedes quitártelo de encima tecleando el siguiente comando:

CLS

cls ENTER

Es abreviatura de 'CLear Screen' (Clarear pantalla) y como verás, eso es lo que hace. Ensáyalo. No importa lo que estuvieras haciendo antes, el resultado será el siguiente:



RUN

Es mágico ¿verdad? Ahora podemos **ejecutar** el siguiente programa de la cassette A, tecleando el comando:

run "letras" ENTER

Ahora, puedes simplemente pulsar teclas a tu antojo y ver lo que sucede. Intenta pulsar más de una tecla al mismo tiempo.

Mientras se usa este programa, puedes ver muchos de los caracteres que pueden aparecer en la pantalla y sin embargo no están marcados en las teclas. Los consigues manteniendo pulsada la tecla CTRL al mismo tiempo que pulsas las teclas de cifras o letras; aunque no todas las teclas tienen esos caracteres 'escondidos'.

Una vez que has comenzado a **ejecutar** un programa de ordenador, continuará hasta que llegue al final. O bien, si está diseñado para repetirse continuamente, como LETRAS, tendrás que pararlo. Ya hemos visto antes que pulsando dos veces ESC lo conseguías. Aquí la manera de hacerlo es 'forzar la restauración' de las condiciones iniciales. Lo que tiene el mismo efecto que si apagaras durante unos segundos y volvieras a encender.

Esta **restauración** la consigues manteniendo pulsadas

CTRL y SHIFT

y pulsando

ESC

Como verás, el CPC464 vuelve al **'saludo'** que ves cuando lo pones en marcha. A partir de ahora, nos referiremos a esta operación como CTRL/SHIFT/ESC.

Aviso Cuando haces esto, el CPC 'olvida' todo lo que pudiera tener en su memoria.

Hasta ahora ¿todo bien? El siguiente programa es muy simple, por lo que aprovecharemos para aprender una nueva manera de decirle que lo ejecute.

Mantén pulsada

CTRL

y pulsa

ENTER

del tablero numérico (la pequeña, no la grande que usas habitualmente).

El CPC464 replicará en pantalla exactamente como si le hubieras tecleado todo un comando entero. ¿Lo reconoces? Recordarás que ya te dijimos antes que esta tecla ENTER tiene una función adicional!

REPITE NOMBRE (el programa que acabas de cargar en virtud del comando 'escondido' anterior) no te mantendrá distraído por mucho tiempo, así que usa CTRL/SHIFT/ESC para restaurar el CPC464 a sus condiciones iniciales, y luego **cargar** y **ejecutar** el siguiente de los programas: TECLADO.

Este programa TECLADO es un programa de entrenamiento para que te acostumbres a las teclas del CPC464. Una vez que hayas empleado un poco de tiempo con él, comenzarás a recordar dónde están las cosas. Y probablemente, merecerá la pena volver a este programa de vez en cuando para mejorar tu velocidad al teclear.

Juego número 2

Seguro que has jugado a este juego anteriormente en papel. Es el del 'ahorcado'. Ahora sabes varias formas de cargar y ejecutar programas, pero simplemente para que te asegures, aquí hay de nuevo una de ellas:

run "ahorcado" **ENTER**

Autoevaluación

Antes de pasar al siguiente capítulo, carga y ejecuta TAE3, para poder evaluar lo aprendido.

4 PONIENDO LAS COSAS EN SU SITIO

Hemos visto por la cassette de demostración y por los programas previos de este curso que el CPC464 tiene unos gráficos soberbios. En este capítulo vamos a iniciarte en lo que necesitas saber para dibujar las líneas apropiadas y las figuras en pantalla según us propios diseños.

El CPC464 muestra todos los caracteres y gráficos en una **'ventana** enmarcada' en la pantalla del monitor. Los laterales del marco, y la parte superior e inferior del mismo, se conocen también como **borde,** aunque no sea exactamente un borde, y nunca se usan. Puedes cambiar el color de este marco (borde), mediante:

BORDER

border 0 ENTER

En negro, ¿verdad? Ensaya ahora el comando:

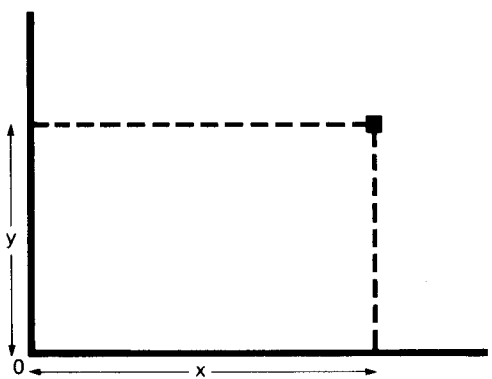
border 26 ENTER

Es el extremo opuesto. Ahora puedes divertirte ensayando todos los números comprendidos en esa gama.

Si ya has leído la **Guía del Usuario,** sabrás que el CPC464 tiene un surtido de 27 colores para que escojas. Incluso aunque no tengas un monitor polícromo CTM640, o un receptor de TV a color, al recorrer los números de esa gama no habrás desperdiciado el tiempo, dado que el orden ascendente de números corresponde con la equivalente **escala de grises** en los televisores blanco y negro.

Coordenadas

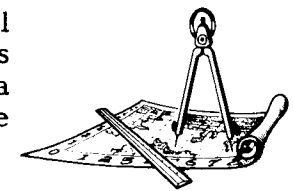
Cada línea, figura o símbolo que ves en la pantalla, está compuesto de un cierto número de elementos de imagen, llamados ("pixels" y **motas**). La posición de cada una de estas motas se describe mediante sus **coordenadas** 'x' e 'y'.



La posición horizontal es dada por la 'abscisa' o coordenada 'x', y la posición vertical por la ordenada o coordenada 'y'.

Ahora ejecuta el siguiente de los programas. Se llama DIBUJAR.

Puedes trazar líneas rectas en la pantalla si mantienes pulsada la tecla TAB al mismo tiempo que pulsas una de las teclas de cursor. Si quieres mover la posición sin que deje **rastro**, simplemente pulsa la tecla SHIFT al mismo tiempo.



Como puedes ver, el programa te va dando las coordenadas X e Y de la posición vigente del cursor. Si mantienes el cursor subiendo, finalmente desaparecerá de la ventana. Observa el valor de la coordenada Y cuando esto pasa. Haz lo mismo con la coordenada X. Con eso descubrirás que en el CPC464 se dispone de una pantalla para gráficos en 640 **puntos** de anchura (x) y 400 **puntos** de altura (y). Intenta dibujar un cuadrado de determinadas dimensiones en una **posición** cualquiera de la pantalla. Por ejemplo:

Sitúa un cuadrado 50 x 50 en la pantalla, con su esquina inferior izquierda en la posición x=300, y=150.

Primero, debes mover el cursor a lo largo del eje X hasta que se lea 300.

MODE

Luego moverlo según el eje Y hasta que sea 150. Si el cuadrado es de 50 x 50 puntos, dibujaremos una línea a partir de ese punto hasta que x=300+50=350. Traza esa línea. Si te pasas puedes 'comerte' el exceso de línea haciendo que el cursor vaya en la otra dirección hasta que la coordenada X sea la correcta. La posición será ahora: x=350 y=150.
Ahora vamos verticalmente hasta que y=150+50=200. Luego horizontalmente a x=350-50=300. Y finalmente, bajando hasta y=200-50=150, con lo que volvemos al punto original de partida con el cuadrado trazado.
No te preocupes si no te sale exactamente a la primera; es el concepto lo que interesa.

A la moda

Antes de continuar con las coordenadas, vamos a explorar otra faceta de nuestra pantalla. Más que "a la moda" deberemos decir "cambios de modalidad o de **modo**".

Cuando enciendes el CPC464, el 'saludo' aparece en caracteres aproximadamente dos veces el tamaño de los de esta página. Habrás notado, sin embargo, que algunos de los programas usan caracteres mayores. Hay de hecho, tres tamaños para los caracteres y se elige diciéndole la clave MODE.

Ensaya ahora. Consigue en tu pantalla el saludo inicial, forzando una **restauración** (CTRL/SHIFT/ESC, como recordarás), y luego teclea el comando:

mode 0 ENTER

Puedes ver que los caracteres son ahora dos veces más anchos que antes. Ahora el comando:

mode 1 ENTER

Nos hace volver al tamaño de caracteres con el que empezamos. Y finalmente, el comando:

mode 2 ENTER

Y tenemos los caracteres a mitad de anchura que el caso más reciente.

Por lo tanto, disponemos de tres modos:

- Modo 0 - 20 caracteres por línea
- Modo 1 - 40 caracteres por línea
- Modo 2 - 80 caracteres por línea

Observa que sólo cambia la anchura de los caracteres y que todavía se dispone de 25 **filas** -líneas horizontales- de caracteres en pantalla, en cualquier momento.

La clave MODE también afecta a la pantalla de gráficos. Pero en el caso de gráficos no hablamos de caracteres, sino de **motas** (puntos de imagen o "pixels").

Una **mota** es la 'mancha cuadradita' más pequeña que puedes dirigir y tratar en pantalla. Hemos visto que la pantalla de gráficos tiene 640 **puntos** por 400 **puntos;** pero el tamaño de la **mota** es distinto para cada uno de los modos posibles:

- Modo 0 - 4 puntos de ancho
  160 x 200 motas
- Modo 1 - 2 puntos de ancho
  320 x 200 motas
- Modo 2 - 1 punto de ancho
  640 x 200 motas

Observa que la **altura** siempre es de 2 **puntos** y no varía con el modo.

En posición de comienzo

Volvamos a las coordenadas. Más adelante veremos cómo situar letras, cifras y signos en la pantalla, pero por el momento nos concentraremos en **gráficos.** El siguiente de los programas se llama COORGEOM, pero antes de cargarlo, veamos una clave muy utilizada: CAT.

CAT corresponde a **catálogo**, y se usa para saber los nombres de los programas que hay en una cassette. Rebobina la cassette hasta el principio, apretando la tecla REW, y luego la tecla:

cat **ENTER**

CAT

El CPC464 te responderá con el mensaje habitual:

Press PLAY and then any key ■

Es simplemente como si le hubieras dado un comando LOAD o RUN, excepto en que en lugar de traer a memoria un programa, el CPC464 pondrá un mensaje en pantalla tal y como:

HOLA block 1 $

Los programas siempre están grabados en cassette separados en **bloques** de 2.000 caracteres. Los programas largos pueden estar compuestos de muchos bloques individualmente grabados. El CPC464 no solamente saca un mensaje en pantalla por cada bloque que encuentra, sino también comprueba al mismo tiempo que no hay errores de grabación y pondrá la frase 'Ok' al final de la línea.

Si le pides que te cargue un programa mencionando su nombre, también obtendrás mensajes de 'encontrado' como el anterior, para todos los otros programas que haya delante del pedido, pero no efectuará la comprobación comentada más arriba.

En cualquier caso, ahora lo que tendríamos que haber encontrado es COORGEOM, para repasar las cordenadas geométricas. Así que cárgalo y vayamos al trabajo.

Juego número 3 - BOMBARDERO

Es uno de los que has estado esperando. Va de bombardeos de una nave extraterrestre. Dale las coordenadas a tu bombardero robot y acierta con la bomba de plutonio exactamente en la diana, o si no **perecerás!**

Autoevaluación

Si has sobrevivido a esta cruenta batalla, comprueba el progreso conseguido en tu entrenamiento, mediante la autoevaluación TAE4, antes de pasar al capítulo siguiente.

5 DIBUJANDO UNA FIGURA

La **palabra clave** PLOT va a ser el primer **'comando'** BASIC que vamos a estudiar en detalle. Hasta ahora hemos estado tecleando palabras como RUN y BORDER sin tener en cuenta que son **comandos** y que tienen que seguir un conjunto concreto de reglas.

Para crear comandos, a menudo tienes que añadir **'parámetros'** a la palabra clave para dar los detalles de la operación a realizar. En el caso de PLOT, los parámetros fijan la posición a **pintar** en la pantalla, según especifiques las coordenadas 'x', 'y'. Por ejemplo, teclea:

DRAW

PLOT

plot 319,199 **ENTER**

Tendrás ahora una **mota** amarilla -un 'pixel' exactamente en medio de la pantalla.

Si tienes problemas recordando quién es 'x' y quien es 'y', no olvides que primero tienes que atravesar la puerta de una casa antes de subir las escaleras, i.e. izquierda-derecha antes de abajo-arriba.

Después de ejecutar el comando PLOT; el CPC464 deja el **cursor de gráficos** en esas coordenadas 'x', 'y', hasta que se le diga lo contrario.

Ahora vamos con otro comando de gráficos:

draw 0,0 **ENTER**

El comando DRAW tiene idéntica estructura a PLOT, excepto que los parámetros dan el punto hasta el que debe **trazarse** la línea -especificado, desde luego, por las coordenadas x, y. La línea diagonal que ahora tienes en la pantalla, comienza en el centro (x=319, y=199) y baja a la esquina inferior izquierda (x=0, y=0). Intenta ahora dibujar el mismo cuadrado que ensayamos en el capítulo anterior.

Aquí están los comandos:

move 3ØØ,15Ø ENTER
draw 35Ø,15Ø ENTER
draw 35Ø,2ØØ ENTER
draw 3ØØ,2ØØ ENTER
draw 3ØØ,15Ø ENTER

El comando MOVE 'mueve' el cursor de gráficos a las coordenadas x, y, especificadas en sus parámetros, sin pintar nada en pantalla.

Un programa cuadrado

Hasta ahora, hemos estado introduciendo comandos para ejecución directa por el CPC464. Vamos ahora a aprender cómo se introduce y deposita un **programa** para su ejecución posterior. Antes de hacer eso tenemos que 'vaciar' el CPC464 tecleando el comando:

new ENTER

NEW

Eso tiene el mismo efecto en la memoria del CPC464, que un paño húmedo en una pizarra. Aunque no se limpia la pantalla, NEW borra cualquier programa que previamente hubiera sido cargado o introducido. Usa sólo este comando cuando estés seguro que ¡no provocará ningún daño!

MOVE

Y ahora vamos con nuestro primer programa almacenado. Es otra vez el mismo cuadrado, sólo que en esta ocasión hemos añadido **números de línea:**

```
1Ø clg
2Ø move 3ØØ,15Ø
3Ø draw 35Ø,15Ø
```

```
40 draw 350,200
50 draw 300,200
60 draw 300,150
```

La línea 10 es para ti otro nuevo comando. Mientras el comando CLS limpia la pantalla y sitúa el **cursor de texto** en la esquina superior izquierda, el comando CLG también **limpia** la pantalla pero luego sitúa el **cursor de gráficos** en la esquina inferior izquierda (x=0, y=0). Estos dos comandos puede que te asombren un poco dado que aparentemente son muy similares. En la Parte 2, aprenderás, sin embargo, a manejar textos y gráficos en la pantalla al mismo tiempo, y su utilidad se hará patente.

CLG

Teclea las seis líneas de este programa **exactamente** como se muestran, y pulsando ENTER al final de cada línea. A partir de ahora y en este curso, debes recordar que has de pulsar la tecla ENTER para terminar cada línea de instrucciones o de comandos.

LIST

Los **números de línea** son necesarios para que puedas indicar al CPC464 en qué **orden** quieres que las instrucciones se almacenen. A no ser que además le digas lo contrario, también ejecutará los comandos en ese orden. Los números de línea van en decenas de forma que puedas incluir líneas si es necesario. No importa en qué orden tecleas las líneas. De hecho, si cometes una equivocación en una de las líneas y no lo detectas hasta después de pulsar la tecla ENTER, puedes reemplazar la línea incorrecta que hay en la memoria, simplemente volviendo a teclear la línea correcta con el mismo número.

Cuando estés satisfecho y pienses que toda está bien, ejecuta el programa tecleando:

run ENTER

Bastante bonito. El CPC464 almacena tu programa en memoria y sólo lo ejecuta cuando le das el comando RUN. Y todavía está ahí. Puedes observarlo si tecleas:

list ENTER

Cambiando el color

Los tres comandos gráficos que acabamos de aprender, tienen un parámetro extra opcional, la **'tinta'**. Intenta volver a introducir la línea 50 del programa en la forma:

5Ø draw 3ØØ,2ØØ,3

Cuando ejecutes esta vez el programa, el lado izquierdo y el lado superior del cuadrado aparecerán en rojo. Es porque el '3' que añadimos en el comando DRAW le indica que debe ejecutarlo usando la INK número 3. El CPC464 continuará luego usando esta 'TINTA' hasta que se le diga que cambie. La gama de diferentes TINTAS que pueden especificarse en un comando DRAW, depende del modo que se esté usando.

El número máximo de diferentes **tintas** que puede usarse en cada modo son como sigue:

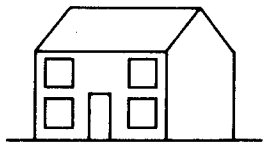
- Modo 0 - 16 tintas
- Modo 1 - 4 tintas
- Modo 2 - 2 tintas

Ahora ensaya tecleando:

ink 3,Ø ENTER

Inmediatamente verás que la línea roja cambia a negra. Haz la misma cosa que hicimos con BORDER en el capítulo anterior y ensaya algunos otros colores para la tinta número 3. Puedes también intentar cambiar las tintas 0, 1 y 2.

INK



REM

PAPER

La casa

El siguiente programa en la cassette A, se denomina CASA. Usa los comandos que hemos aprendido hasta ahora y dos más. El primero es REM que corresponde a REMARK (=memorandum, comentario). Cuando el CPC464 tropieza con la clave REM, 'rememora' el resto de esa línea. Eso te permite colocar explicaciones en el programa de manera que otra gente (o tú mismo, si lo has olvidado después de un cierto tiempo) pueda comprender más fácilmente de qué trata el programa.

El otro comando es PAPER, que te permite especificar el color del **fondo** ('papel') de la ventana en pantalla. El parámetro para PAPER debe ser el número de una de las INK especificadas para el modo vigente. Si has cambiado INK 0 cuando te lo sugerimos antes, habrás visto que éste fue el que se eligió automáticamente para el PAPEL cuando el CPC464 fue puesto en marcha.

Antes de ejecutar el programa, estudia el siguiente **listado.**

```
10 REM Dibujando una casa
20 MODE 0
30 CLS
40 REM ** comienzo **
50 BORDER 12
60 INK 0,12:REM amarillo
70 INK 1,3:REM rojo
80 INK 2,6:REM rojo brillante
90 INK 3,9:REM verde
100 PAPER 0
110 REM parte frontal
120 MOVE 100,50
130 DRAW 100,250,1
140 DRAW 400,250
150 DRAW 400,50
160 DRAW 100,50
170 REM lateral
180 MOVE 400,250
190 DRAW 600,250
200 DRAW 600,50
210 DRAW 400,50
220 DRAW 400,250
230 REM fin de tejado
240 REM vuelta al origen
250 REM no se necesita MOVE
260 DRAW 500,350
```

```
270 DRAW 600,250
280 DRAW 400,250
290 REM tejado
300 REM solo se necesitan dos lineas
310 MOVE 100,250
320 DRAW 200,350
330 DRAW 500,350
340 REM puerta
350 MOVE 225,50
360 DRAW 225,140,2
370 DRAW 275,140
380 DRAW 275,50
390 REM ventanas
400 REM abajo izquierda
410 MOVE 120,70
420 DRAW 120,130,3
430 DRAW 180,130
440 DRAW 180,70
450 DRAW 120,70
460 REM arriba izquierda
470 MOVE 120,170
480 DRAW 120,230
490 DRAW 180,230
500 DRAW 180,170
510 DRAW 120,170
520 REM arriba derecha
```

```
530 MOVE 320,170
540 DRAW 320,230
550 DRAW 380,230
560 DRAW 380,170
570 DRAW 320,170
580 REM abajo derecha
590 MOVE 320,70
600 DRAW 320,130
610 DRAW 380,130
620 DRAW 380,70
630 DRAW 320,70
```

Autoevaluación

Puede que te guste repasar de nuevo este capítulo antes de autoevaluarte con TAE5. Hemos comentado bastantes fundamentos en poco tiempo, pero ahora ya estamos llegando a la programación real.

6 CIFRAS, LETRAS Y PALABRAS

PRINT

LET

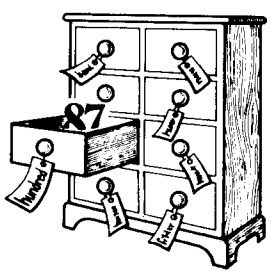
Ya es hora de que aprendamos algunas de las claves relacionadas con **sacar** números y palabras en la pantalla. La primera de ellas es PRINT. Intenta teclear directamente:

print 2+2 **ENTER**

Como habrás adivinado, la respuesta sale en la siguiente línea de la pantalla y desde luego es 4. Ensaya ahora esto:

print "Hola" **ENTER**

En este caso, hemos tenido que encerrar **entre comillas** lo que queríamos sacar en pantalla. La razón se hará obvia más adelante. Por el momento exploraremos otra **clave.**



Depositando en memoria

Algunas veces, el mundo informático parece ser verdaderamente un lugar muy extraño. Las claves BASIC son palabras inglesas normales (la mayoría de las veces) pero pueden tener significado especial para el CPC464. Una de ellas es la palabra clave LET. En el álgebra ordinaria escribirías; ('Sea x=5'), aunque es mejor 'Haga x=5').

Let x=5

Sin embargo, no podrías escribir:

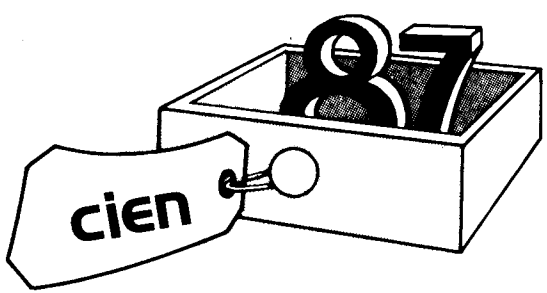
Let x=x+5

Y eso es perfectamente aceptable en BASIC. Y significa: 'Tome el valor previo de x, súmale 5 y luego haga que este número sea el nuevo valor de x'. Pero, ¿por qué x? Bien, hemos topado simplemente con algo que llamamos **variable,** y por muy buenas razones.

Variables son como algo salido de **'Alicia y el espejo maravilloso'.** Contienen lo que te hayas preocupado de **depositar** dentro. No hay nada en BASIC que te impida escribir:

Let cien = 87

Y si no te gusta 'cien' puedes usar 'h' o 'C', dado que lo que estamos hablando es de **rótulos** en cajas vacías. Cuando usas LET para **dotar** a una variable de algo, el CPC464 escribe el **nombre** en unos cajetínes vacíos contíguos y luego mete dentro de ellos una serie de cifras que le representan el **valor** que especificas a la derecha del signo '='.



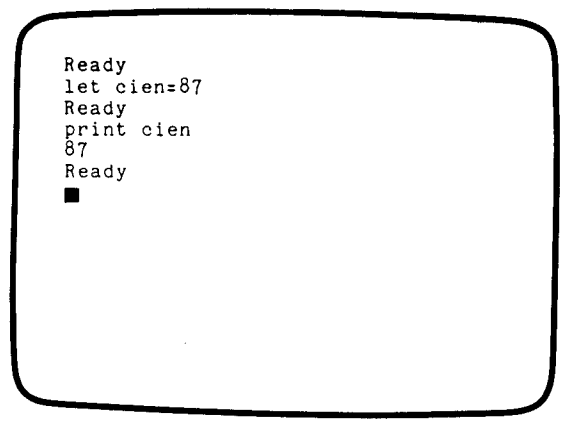
Ensaya tecleando en tu CPC464, lo siguiente:

Let cien = 87 **ENTER**

Obtendrás como réplica "Ready". Luego teclea:

print cien **ENTER**

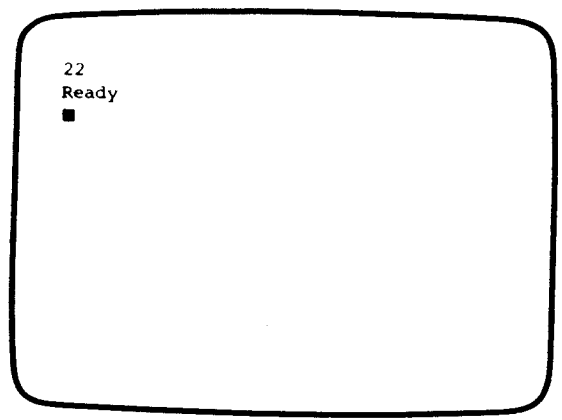
Tendrás ahora en la pantalla lo siguiente:



Así, hemos llegado al punto en que podemos teclear nuestro segundo programa. Las siguientes cinco líneas deberás teclearlas exactamente como aparecen aquí:

```
10 cls
20 let a=15
30 let b=7
40 let a=a+b
50 print a
```

Recuerda pulsar ENTER para terminar cada línea. Ahora ejecuta este programa. Deberás conseguir el siguiente resultado:



Puede que te haya parecido bastante obvio, pero el siguiente quizás no.

```
10 let a=5
20 let b=10
30 let a=b
40 let b=a
50 print a+b
```

Ensáyalo por ti mismo. ¿Debe ser 20 la respuesta? Puede que te ayude pensar en los números moviéndose de unas cajas a otras. Recuerda que es la variable a la **izquierda** del signo '=' la que es cambiada al ponerle dentro un nuevo valor, y que el CPC464 va a seguir el programa una instrucción después de otra.

Cadenas de cosas

Hemos visto que podemos definir una variable para números, y usarla como si fuera realmente un número.

BASIC también nos permite definir variables que pueden contener una **serie** de letras, cifras y caracteres especiales. Se denominan en inglés 'strings', porque considera a los caracteres ensartados unos a otros, formando una sola cosa. También se dice que son variables **literales** (i.e. datos al 'pie de la letra').

Una variable **literal** es prácticamente lo mismo que una variable **numeral**, excepto que debes terminar el nombre de la variable con un signo 'dólar'. Además, debes siempre poner **comillas** (") en ambos lados de los caracteres que deseas contenga la variable; en caso contrario, el CPC464 piensa que le están diciendo algo sobre otra variable. Teclea la siguiente línea:

let a$=Hola ENTER

Obtendrás el mensaje de error:

Type mismatch

El CPC464 pensó que estabas intentando meter en la **variable literal** a$ el **valor numérico** de otra variable llamada 'hola'.

Lo que deberías haber tecleado es:

let a$="Hola" ENTER

Y con ello, cuando le dijeras:

print a$ ENTER

Saldría lo que deseas. Ensáyalo. Y comprende claramente que son datos de diferente **índole**; y de ahí el mensaje de "discordancia ('mismatch') de clase (type)" que te dió anteriormente.

Ahora teclea el siguiente programa. Mira a ver si puedes calcular cuál va a ser el resultado antes de que lo compruebes en tu CPC464.

```
10 let a$="5"
20 let b$="12"
```

```
30 let a=5
40 let b=12
50 let c=a+b
60 let c$=a$+b$
70 print c
80 print c$
```

¿Qué hay en los nombres?

Puedes usar cualquier nombre que te guste para una variable, excepto que el CPC464 te rechazará el uso de cualquier **clave** BASIC. Por ejemplo, no aceptará:

let save=s+p ENTER

Eso da un error de sintáxis. Tendrías que cambiar ese nombre por el de 'savers' o 'asaven'. La lista completa de **palabras reservadas** en la Guía del Usuario CPC464 te indicará las que debes evitar.

Habrás notado probablemente que al CPC464 no le importa si le tecleas los **comandos** en mayúsculas o en minúsculas. Cualquier cosa que hagas, él siempre te mostrará las claves en mayúsculas al listar el programa.

Si mantienes todos tus nombres de variables en minúculas, será más fácil encontrarlas en el listado.

Guardar

Hasta ahora, has estado **cargando** programas de las dos cassettes que vienen con el curso, o introduciéndolos mediante el teclado. Ahora es la oportunidad de **guardarlos** para la posteridad, o simplemente para poderlos usar de nuevo en este capítulo y en el siguiente sin tener que teclearlos otra vez. Quita la cassette de la lecto-grabadora y sustitúyela con una nueva que no tenga quitada la lengüeta de grabación. Las cassettes ordinarias bastarán, aunque debieras evitar las de duración superior a C60 porque la cinta es demasiado delgada. También debes ser consciente del trozo de cabecera que existe en las cassettes ordinarias.

Rebobina la cassette hasta el principio, y teclea:

save "variables" ENTER

Puedes elegir un **título, un nombre de programa,** diferente, si lo deseas, y con mezcla de mayúsculas, minúsculas y blancos si lo requieres. Debes, sin embargo, poner siempre comillas en ambos lados del título. El CPC464 replicará:

Press REC and PLAY and then any key:

para indicarte que oprimas ambos botones y poder grabar en cassette el programa. Cuando lo hagas, recibirás el mensaje:

Saving VARIABLES block 1

El CPC464 pondrá en marcha la grabadora y podrás oir cómo se transfiere el programa de la memoria al cassette. Mientras está **guardando** ('saving') el programa, desaparecerá de la pantalla el cursor, pero volverá cuando el CPC464 haya terminado esta operación, junto con la **divisa** 'Ready'. Siempre es una buena práctica hacer dos copias de un programa para casos de percances con el cassette, por tanto, repite lo anterior para guardar el programa por segunda vez. No olvides liberar el botón REC del cassette, después de eso.

SAVE

Más exposiciones

Si has 'ejecutado' el prograna anterior, el CPC tendrá todavía las variables en memoria. Si no, carga y ejecuta el programa de nuevo de forma que puedas usar las variables para investigar algunas otras cosas sobre el comando PRINT. Teclea:

print a,b,c ENTER

Ensaya esto mismo en el modo 0 y verás el porqué. Si no quieres que los números aparezcan separados, puedes hacer lo siguiente:

print a;b;c ENTER

Esta vez, los números se exponen en la misma línea pero sin espacios en blanco entre ellos, por lo que a menudo tendrás que añadir espacios como en el siguiente ejemplo:

print "el valor de c es";c;" no "; c$

Puedes colocar letras y números en cualquier parte de la pantalla que quieras, por medio del comando LOCATE. La estructura de este comando de **'ubicación'** es:

locate x,y

Lo que te suena familiar, aunque ten cuidado.

Estas coordenadas x, y no son las mismas que las coordenadas en gráficos. El comando LOCATE sitúa el cursor de texto en la posición de la pantalla especificada por los parámetros del comando. Las coordenadas para texto comienzan en la esquina superior izquierda de la pantalla (que tiene como coordenadas x=1, y=1), y luego se cuentan **a lo ancho** y **hacia abajo** de la pantalla.

Teclea la instrucción:

75 locate 2Ø,13 ENTER

Limpia la pantalla con CLS. Si ahora ejecutas de nuevo el programa con esta nueva línea, verás que el valor de c$ aparece en mitad de la pantalla, empezando en la posición número 20 (**columna** 20) de la **fila** 13 (llamando filas a las líneas horizontales).

BLOQUES

'Histogramas' es el nombre de los **diagramas de barras,** y el programa correspondiente se llama BLOQUES. Te da una representación visual de 4 números entre 0 y 290, que es la clase de cosas que ves en los resultados de las elecciones y en los sondeos de opinión. Esta clase de programa tiene que esperar en ciertos puntos a que le **impongas** números mediante el teclado, para depositarlos en lugares concretos de su memoria, y luego continuar. La palabra clave que hace esto es INPUT y la forma típica del comando es:

1Ø INPUT nombre$

El CPC464 esperará pacientemente en esa línea 10 hasta que se le teclee algo que se termine pulsando la tecla ENTER. El dato tecleado lo 'impone' como valor de la variable literal 'nombre$' (el **lugar** en la memoria) y continúa con el seguimiento del programa.

Ahora ejecuta BLOQUES antes de estudiar el listado que te presentamos. Verás que el CPC464 muestra un signo de interrogación (?), seguido del cursor, cuando está esperando que le introduzcas un dato. También puede poner un **aviso** para que el usuario sepa la clase de información que está esperando. Para hacerlo, se coloca el mensaje de aviso después de la clave INPUT, en la forma siguiente:

INPUT "Nuro Votos (1-29Ø)";a

El mensaje de aviso debe encerrarse entre comillas (") y separarse de la variable por un punto-y-coma (;).

Puedes ver que varias de las líneas del programa constan de varios comandos separados por dos-puntos (:). Este signo de puntuación sirve para el mismo propósito que si comenzaras un comando con un número de línea y lo terminaras con ENTER. Es una buena manera de mantener una **serie** de comandos relacionados, particularmente cuando son breves.

INPUT

Uno de los usos mejores para los **dos-puntos** es:

5Ø b=5Ø:REM tamaño

Poniendo un REM al final del comando, para explicar a qué se refiere, hace el programa más legible en momentos posteriores.

```
10 REM 3D Bar Chart
20 REM by Dave Atherton
30 MODE 1
40 BORDER 14:INK 0,14:INK 1,0:INK 2,8:INK 3,24
50 b=50:REM bar size
60 LOCATE 1,23:INPUT "Altura (1-290)";a
70 x=100:PLOT x,55,1
80 DRAW x-b*2,55:DRAW x-b*2,a+55
90 DRAW x-b,a+55+b:DRAW x+b,a+55+b
100 DRAW x,a+55:DRAW x-b*2,a+55
110 MOVE x+b,a+b+55:DRAW x+b,b+55
120 DRAW x,55:DRAW x,55+a
130 LOCATE 1,23
140 PRINT"                                                    "
150 LOCATE 1,23:INPUT "Altura (1-290)";a
160 x=260:PLOT x,55,2
170 DRAW x-b*2,55:DRAW x-b*2,a+55
180 DRAW x-b,a+55+b:DRAW x+b,a+55+b
190 DRAW x,a+55:DRAW x-b*2,a+55
200 MOVE x+b,a+b+55:DRAW x+b,b+55
210 DRAW x,55:DRAW x,55+a
220 LOCATE 1,23
230 PRINT"                                                    "
240 LOCATE 1,23:INPUT "Altura (1-290)";a
250 x=420:PLOT x,55,3
```

```
260 DRAW x-b*2,55:DRAW x-b*2,a+55
270 DRAW x-b,a+55+b:DRAW x+b,a+55+b
280 DRAW x,a+55:DRAW x-b*2,a+55
290 MOVE x+b,a+b+55:DRAW x+b,b+55
300 DRAW x,55:DRAW x,55+a
310 LOCATE 1,23
320 PRINT'                                                    '
330 LOCATE 1,23:INPUT 'Altura (1-290)';a
340 x=580:PLOT x,55,1
350 DRAW x-b*2,55:DRAW x-b*2,a+55
360 DRAW x-b,a+55+b:DRAW x+b,a+55+b
370 DRAW x,a+55:DRAW x-b*2,a+55
380 MOVE x+b,a+b+55:DRAW x+b,b+55
390 DRAW x,55:DRAW x,55+a
```

Juego número 4

Alguna gente usa un montón de palabras rimbombantes y que suenan a simple pedantería -especialmente en el mundo de los ordenadores. Nuestro generador automático de 'cháchara grandilocuente' -programa que llamamos GENERADOR DE FRASES para que te ayude a producir **'insensateces'** de la misma categoría que las de ellos.

Autoevaluación

Ejecuta el programa TAE6 para autoevaluarte y ver si necesitas repasar algo de este capítulo antes de pasar al siguiente.

7 CONSEGUIRLO BIEN

Una gran parte de este capítulo trata de correcciones y cambios a los programas. El programa que introdujiste en el capítulo anterior será ideal para este fin, así que rebobina la cassette y carga de nuevo el programa en memoria, tecleando:

load "variables" **ENTER**

'Variables' es el nombre que diste al programa cuando usaste el comando SAVE para guardarlo en el cassette.

Cambio de líneas

Cuando estás metiendo un programa por medio del teclado, es casi seguro que cometas equivocaciones, incluso aunque sólo estés copiando de una página impresa. La equivocación más común es cuando le das a la tecla que no quieres. Habitualmente, te das cuenta enseguida que lo has hecho, antes de pulsar la tecla ENTER, así que puedes retroceder y quitar el carácter que te molesta usando la tecla DEL (de 'delete'=suprimir)

Mientras que estás tecleando una línea (la línea **corriente**), puedes mover el cursor atrás y adelante en esa línea, con las teclas de cursor. Puedes también insertar nuevos caracteres, o puedes suprimir caracteres mediante las teclas CLR (de 'clear'=limpiar, **aclarar**) y DEL.

En el capítulo anterior vimos también que puedes sustituir líneas completas, simplemente volviéndolas a teclear. El CPC464 automáticamente colocará la nueva línea en lugar de la vieja.

A medida que escribas más programas propios, vas a darte cuenta que nunca funcionan la primera vez apropiadamente. El CP464 te dará algunos mensajes de error cuando tecleas las líneas y otros cuando intentes ejecutar el programa. Pero no puede decirte, por ejemplo, si te has dejado fuera comandos o te has olvidado de mover el cursor a una nueva posición antes de dibujar una línea. Esa es la razón de que los números de línea vayan **normalmente** en decenas.

Como vimos en el capítulo anterior, añadimos una línea entre la 70 y la 80, si tecleamos:

75 locate 2Ø,13

En casos extremos, no hay nada que te impida añadir hasta nueve líneas en esa o en cualquier otra posición. De la misma manera, puedes suprimir líneas enteras si tecleas una línea en blanco. La línea que hemos añadido antes, podremos quitarla simplemente con:

75 ENTER

El CPC464 eliminará del programa la línea 75. Ensáyalo y luego LISTa el programa para ver lo que ha pasado.

Corrección de errores

EDIT

Cuando necesitas modificar un programa que ya ha sido tecleado o cargado, puedes usar la función EDIT para ENMENDARLO, CORREGIRLO (**'editarlo'**). Usando el ejemplo anterior, teclea:

edit 40 ENTER

Como puedes ver, la línea 40 aparece en pantalla con el cursor sobre el primer carácter de esa línea. Luego usa las teclas de cursor izquierda y derecha para situarlo sobre la parte de la línea que quieres cambiar, al igual que si estuvieras trabajando en la línea **vigente**. Ensáyalo por ti mismo, cambiando el valor de b de 15 a 26. Sitúa el cursor sobre el '1' del '15' y luego pulsa dos veces la tecla CLR. Con eso se borrará el 15. Ahora teclea 26 y pulsa la tecla ENTER. El cursor no necesita ser llevado al extremo de la línea. Si ahora LISTAS o EJECUTAS el programa, verás que el CPC464 ha alterado la línea en la forma que has dicho.

Pero una manera más fácil de **'editar'** líneas es usar la tecla COPY. Un segundo cursor, llamado el **cursor de copia**, se emplea para **recoger** líneas o partes de línea de cualquier sitio de la pantalla. Consigues que aparezca el cursor de copia si mantienes pulsado SHIFT mientras pulsas una de las teclas de cursor. Lista de nuevo el programa y ensaya, situando el cursor de copia al comienzo de cualquiera de las líneas. Verás que el cursor 'normal' permanece en su sitio. Pulsa la tecla COPY para copiar cada símbolo mostrado en la pantalla, sobre la línea **vigente** con el cursor normal. Si mantienes apretada la tecla COPY se repetirá automáticamente. Puedes dejar de copiar en cualquier posición de la línea original y luego teclear cualquier información en la **corriente** línea, antes de reanudar la operación de copiar de la pantalla.

Una vez que te hayas acostumbrado al cursor de copia, probablemente no usarás el comando EDIT muy a menudo.

El efecto es casi idéntico, pero con la ventaja añadida de poder dejar fuera de una línea los caracteres que ya no quieres más.

Deja el "HAGA"

Puede que te haya parecido un poco aburrido teclear la clave LET al comienzo de las líneas del último capítulo. Es hora de hacer una confesión. No necesitas usar esta clave en el BASIC Amstrad.

Como recuerdas, sirve para asignar, para **dotar** a una variable de un valor. Sin la palabra LET sigue funcionando igual, aunque en cierto modo si lo haces, rompes la estructura del BASIC en que cada comando empieza con una palabra **clave.** Ensaya la corrección del programa VARIABLES, para quitar todas las claves LET. Si ahora lo ejecutas, una vez modificado, te darás cuenta que no hay ninguna diferencia en absoluto.

Brincos y saltos

Hemos dicho antes que los números de línea muestran al CPC464 el orden en que debe almacenar las instrucciones, pero también dijimos que no siempre se ejecutan en ese orden. Hay veces, en que queremos dar un **salto** hacia adelante o hacia atrás a otra línea del programa y le mandamos que **vaya** allí, usando la clave GOTO. Este comando se forma simplemente añadiendo el número de línea. Por ejemplo:

GOTO

7Ø goto 13Ø

Esta línea hará que el CPC464 pegue un **brinco,** saltándose todas las líneas, y VAYA directamente a ejecutar la línea 130. Y puedes hacer la misma cosa pero al revés, con un salto **'regresivo':**

13Ø goto 7Ø

El programa ahora 'iría' a la línea 70 y ejecutaría lo que se le dice allí, que es que salte todas y vaya a la línea 130, en la que se le manda de nuevo a la 70... y continuaría haciéndolo así hasta que apagaras o pulsaras la tecla ESC para escapar.

Y luego, ¿qué pasa?

ELSE

Afrontamos **decisiones** durante cada hora de las que estamos despiertos, incluso aunque sólo sean en cosas triviales. Habiendo previsto lo que vamos a hacer a continuación, e.g. beber café, té, o nada; o ponernos zapatos negros, rojos o ningunos; podemos adoptar el **'curso'** de acción apropiado. El CPC464 también sigue cursos **alternativos** de acciones, si le decimos la palabra clave IF, que es equivalente a nuestro **condicional** (Si..., pues...).

IF

La línea siguiente podría ser de un programa que comprueba la cantidad de dinero que has ahorrado, siendo la variable 'dinero' la que te dice el saldo de tu cuenta:

THEN

IF dinero=Ø THEN PRINT "apurado"

Lo interesante es que entonces el comando PRINT se ejecutará **sólo** si 'dinero' es 0 **-en** los **demás** casos, el CPC464 simplemente pasa a obedecer la siguiente línea del programa.

Hay una manera mejor de tomar una decisión entre dos alternativas; y es incluir otra cláusula en la instrucción, encabezada por la clave ELSE, para que sea ejecutado **en** las **demás** situaciones de la forma siguiente:

IF dinero >Ø THEN PRINT "Rico" ELSE PRINT "apurado"

El signo ' >' significa **mayor que** y se preocupa de la situación cuando debes dinero al banco. Si 'dinero' es 0 o menos (números rojos), el CPC464 ignora el comando entre las claves THEN y ELSE, y ejecuta el comando que venga detrás de ELSE. ELSE (**'en demás...'**) te da la posibilidad de seguir una dirección alternativa en las acciones, antes de pasar a la línea siguiente del programa.

En el capítulo anterior examinamos un programa llamado BLOQUES. Había una instrucción de entrada que preguntaba:

altura (Ø-29Ø)?

Si intentaste teclear un número mayor de 290, observarías que el programa parecía no percatarse y dibujaba la barra fuera, por arriba de la pantalla. Puedes evitar que esto suceda, si limitas la variable 'a' a su máximo valor. Por ejemplo con:

```
IF a>29Ø THEN LET a=29Ø
```

Como puedes ver, el comando IF... THEN... no está restringido a instrucciones PRINT, y con esa línea de programa "SI a es mayor de 290 ENTONCES HACES que a sea igual a 290", que es el máximo valor admitido.

SI es así, PUES que se VAYA

La palabra clave IF, realmente aprovecha su significado cuando la usas con GOTO. Es como un poste en la carretera que le dice al CPC464 el **'curso'** que el programa debe seguir a continuación. Supongamos que quieres impedir que la gente teclee números mayores de 295, en el programa BLOQUES. Las siguientes líneas te indican un buen truco:

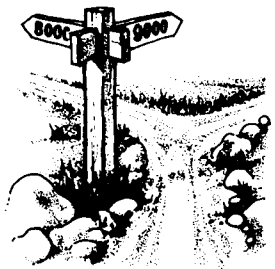
```
6Ø INPUT "Altura (Ø-29Ø) ";a
65 IF a>29Ø THEN GOTO 6Ø
```

Hasta que un número menor o igual a 290 sea tecleado, el CPC464 continuará volviendo a la línea 60. Esta 'trampa' realmente evitará que se impongan números fuera de la gama permitida, en lugar de como antes simplemente reducirlos a la fuerza.

Obviamente, la cláusula encabezada por ELSE (**en** las **demás** situaciones...), es **opcional**, no es obligatoria; y la instrucción 65 anterior equivale exactamente a:

```
65 IF a>29Ø THEN GOTO 6Ø
   ELSE GOTO 7Ø
```

siempre que sea la 70 la línea **siguiente** a la 65).



Fuera pifias

Puedes imaginarte que si haces un pequeño error al escribir una de las instrucciones anteriores, el CPC464 puede **irse** a un lugar equivocado del programa. Y acertarás. Además, has tropezado con una de las palabras clásicas de la jerga informática: la **pifia** (en inglés el "bug": la chinche).

Las máquinas tienen defectos, la gente comete equivocaciones. Anteriormente dijimos que los ordenadores no pueden pensar y que el programador tiene que hacer el **razonamiento** por él. Sí, el programador comete una equivocación, e incluso los mejores no sois inmunes a ellas, habitualmente sólo se hace patente después de intentar ejecutar el programa.

El CPC464 hace lo que se le ha dicho que haga, pero puede que no sea lo que el programador intentaba originalmente.

La **depuración** (en inglés, 'debugging') es el proceso de quitar las **pifias** de un programa y corregir los errores de razonamiento que contiene. No hay que pasar ninguna vergüenza, ni es innominioso escribir un programa que tenga pifias. Admitamos, que los programadores expertos tienen menos pifias al acabar un programa que los programadores novatos, pero es simplemente cuestión de conocimiento, y sobre todo de **práctica.** Con el tiempo, verás que cada vez cometes menos y menos errores en tus programas.

Es práctico, sin embargo, comprobar bien un programa antes de ejecutarlo. Y una manera de hacerlo es actuar como ordenador y hacer tú una pasada -recorriendo línea a línea el programa y escribiendo al lado los valores de las variables y los resultados de las operaciones. Vayamos con un ejemplo:

```
10 let a=0
20 print a
30 let a=a+1
40 if a<4 goto 20
```

Antes de tecleárselo al CPC464, consigue una hoja de papel y escribe las siguientes tres cabeceras de una tabla:

Paso Número de línea Valor de a

Ahora **sigue** tú el programa, ejecutando cada instrucción, exactamente igual a como lo haría el CPC464. Esto es lo que obtendrías.

Paso	Número de línea	Valor de a
1	10	0
2	20	0
3	30	1
4	40	1
5	20	1
6	30	2
7	40	2
8	20	2
9	30	3
10	40	3
11	20	3
12	30	4
13	40	4

La técnica de usar estas comprobaciones es una buena manera de forzarte a ver el programa desde el punto de vista de la máquina. De ese modo, cualquiera de los problemas se hace obvio. Cuando acabas de escribir un programa, tu dificultad principal puede ser que sepas exactamente cómo crees que funciona, y este método puede llevarte a ver ¡que así no lo hace!

Otra técnica es incluir momentáneamente instrucciones PRINT en ciertos puntos del recorrido del programa, para mostrar el valor de las variables en ese momento.

Un ejemplo simple sería añadir la siguiente línea en el programa anterior.

```
35 print "Línea 35 a= ";a
```

También puedes usar el comando STOP y una vez PARADO, analizar los valores de las variables:

```
35 stop
```

STOP

El CPC464 detendrá el seguimiento del programa en ese punto, y puedes hacerle que te exponga la variable o variables que te interesen, dándole comandos directamente. Se hace que el programa SIGA, tecleando el comando:

CONT

cont ENTER

Que corresponde a CONTinúe. Recuerda que no funcionará si has añadido o suprimido líneas en el intermedio, después de hacer que PARE y antes de decirle que SIGA.

Decorar la casa

Sí, volvemos otra vez con la casa. Esta vez vamos a dar unas manos de pintura aquí y allí, así que saca los diagramas de colores y carga el programa DECO, que vamos a **deco**rarla.

Este no es realmente un juego, sino una manera entretenida de combinar todos los comandos que has aprendido hasta ahora. Puede también que te guste modificar el programa y dar a tu imagen un poco de individualismo. Debieras ahora saber bastante como para hacerlo, usando las técnicas descritas al comienzo de este capítulo.

No te preocupes sobre las palabras clave que no te sean familiares en el listado de DECO que te damos en las siguientes páginas. Todo se aclarará en capítulos posteriores.

```
10 REM deco
20 MODE 0
30 CLS
40 REM ** inicio **
50 BORDER 12
60 INK 0,12:REM amarillo
70 INK 3,3:REM rojo
80 INK 6,6:REM rojo brillante
90 INK 9,9:REM verde
100 PAPER 0
110 REM dibuja frente
120 MOVE 100,50
130 DRAW 100,250,3
140 DRAW 400,250
150 DRAW 400,50
160 DRAW 100,50
170 REM dibuja costado
180 MOVE 400,250
190 DRAW 600,250
200 DRAW 600,50
210 DRAW 400,50
220 DRAW 400,250
230 REM dibuja faldon
240 REM situado YA en comienzo
250 REM no necesitas 'MUEVA'
260 DRAW 500,350
```

```
270 DRAW 600,250
280 DRAW 400,250
290 REM dibuja tejado
300 REM basta con dos 'trazos'
310 MOVE 100,250
320 DRAW 200,350
330 DRAW 500,350
340 REM dibuja puerta
350 MOVE 225,50
360 DRAW 225,140,6
370 DRAW 275,140
380 DRAW 275,50
390 REM dibuja ventanas
400 REM inferior izquierda
410 MOVE 120,70
420 DRAW 120,130,9
430 DRAW 180,130
440 DRAW 180,70
450 DRAW 120,70
460 REM superior izquierda
470 MOVE 120,170
480 DRAW 120,230
490 DRAW 180,230
500 DRAW 180,170
510 DRAW 120,170
520 REM superior derecha
530 MOVE 320,170
540 DRAW 320,230
```

```
550 DRAW 380,230
560 DRAW 380,170
570 DRAW 320,170
580 REM inferior derecha
590 MOVE 320,70
600 DRAW 320,130
610 DRAW 380,130
620 DRAW 380,70
630 DRAW 320,70
640 REM ** PINTANDO **
650 r$=CHR$(18)
660 LOCATE 1,25
670 PRINT"Teclea color (1-15)";
680 FOR i=1 TO 15
690 INK i,i:NEXT i
700 LOCATE 1,1:PRINT r$;
710 INPUT "Color del tejado";r
720 FOR i=107 TO 399 STEP 2
730 MOVE i,252:DRAW i+96,349,r
740 NEXT i
750 LOCATE 1,1:PRINT r$;
760 INPUT "Faldon del tejado";g
770 FOR i=252 TO 346
780 MOVE i+154,i:DRAW 848-i,i,g
790 NEXT i
800 LOCATE 1,1:PRINT r$;
810 INPUT "Pared del costado";e
820 FOR i=52 TO 248 STEP 2
```

```
830 MOVE 404,i:DRAW 598,i,e
840 NEXT i
850 LOCATE 1,1:PRINT r$;
860 INPUT "Color de fachada";f
870 FOR i=52 TO 248 STEP 2
880 MOVE 104,i:DRAW 398,i,f
890 NEXT i
900 LOCATE 1,1:PRINT r$;
910 INPUT "Color de puerta";d
920 FOR i=52 TO 138
930 MOVE 229,i:DRAW 268,i,d
940 NEXT i
950 LOCATE 1,1:PRINT r$;
960 INPUT "Color de ventanas";w
970 FOR j=0 TO 100 STEP 100
980 FOR i=70+j TO 130+j
990 MOVE 120,i:DRAW 180,i,w
1000 MOVE 320,i:DRAW 380,i
1010 NEXT i
1020 NEXT j
1030 END
```

Autoevaluación

El programa TAE7 te permitirá comprobar si has comprendido bien este capítulo. No te preocupes si tienes que ojear otra vez estas páginas antes de contestar las preguntas. Muchos programadores trabajan con un **libro de referencia** a su lado.

8 MEJORAS DOMESTICAS

En la programación de computadoras, hay algunas **tareas,** que como el trabajo doméstico, necesitan hacerse una y otra vez. El CPC464, como cualquier otro ordenador, es muy bueno para las **tareas repetitivas.** Si le damos el programa apropiado, estará continuamente repitiendo hasta que le digamos otra cosa. Así que, volvamos otra vez a la casa, para ver cómo podemos rellenar un montón de detalles sin necesidad de escribir demasiado.

Carga el siguiente programa en la cassette, MANSION. Puede parecer que es la misma vieja casa, pero esta vez vamos a ver cómo podemos producir imágenes visuales ingeniosas usando los fundamentos de **bucles** y de **subrutinas.**

Otra ronda

Sorprendente puede parecer, pero ya has visto un **bucle** en el capítulo anterior. Aunque no lo llamamos así en esa ocasión, el programa contenía un **bucle:**

```
10 let a=0
20 print a <----------
30 let a=a+1          |
40 if a<4 goto 20 ----|
```

Lo que este programa significa es que 'exponga el valor de a, le sume una unidad, se VAYA a la línea 20, y continúe **repitiendo** esta ronda hasta que el valor de a sea igual a 4'. (Si es menor, se va a la 20 y da 'otra' ronda).

Podríamos haber escrito este programa en la forma:

```
10 for a=0 to 3
20 print a
30 next a
```

En el que el principio del bucle se señala en la línea 10 que le dice 'CON a variando de 0 a 3', repita las instrucciones comprendidas entre ésta y la línea 30 que señala 'OTRA a'; momento en que vuelve a la línea 10 para tomar el siguiente valor de a, incrementarlo en la unidad y analizar si procede dar otra ronda con ese nuevo valor de a; o si ya debe salirse del bucle (y pasar a la 40).

Si examinas el listado del programa MANSION, impreso más adelante, verás que ya no termina en la línea 640. Primeramente examinemos las nuevas líneas entre la 640 a la 680, ambas inclusive.

Si pasas el programa, dibujará la casa como siempre. Cuando llega a la instrucción STOP puedes ejecutar el siguiente trozo tecleando:

CONT **ENTER**

Ensáyalo. ¿Qué piensas de la valla? Puede incluso que mantenga al perro del vecino fuera del jardín!

La clave de esta operación es la línea 650, así que examinémosla más detenidamente:

65Ø FOR F=Ø TO 62Ø STEP 2Ø

FOR

El comando FOR le dice al CPC464 que justamente **entra** en un bucle, CON una **variable de control** llamada F. 'F=0' le dice el valor con el que debe **comenzar** la primera ronda; 'TO 620' le indica cuándo debe finalizar y **salirse** del bucle.

STEP

La línea 660 usa el valor vigente de la variable 'F' para mover el cursor de gráficos al siguiente poste de la valla y que dibuje la línea de dicho poste.

La instrucción que señala dónde termina una ronda, es la instrucción NEXT, que puedes ver en la línea 670. Significa realmente 'mira a ver si procede dar OTRA ronda', y para ello vuelve a la 650 y 'coge otro valor para F' usando el 'incremento' estipulado y lo compara con el valor final estipulado.

NEXT

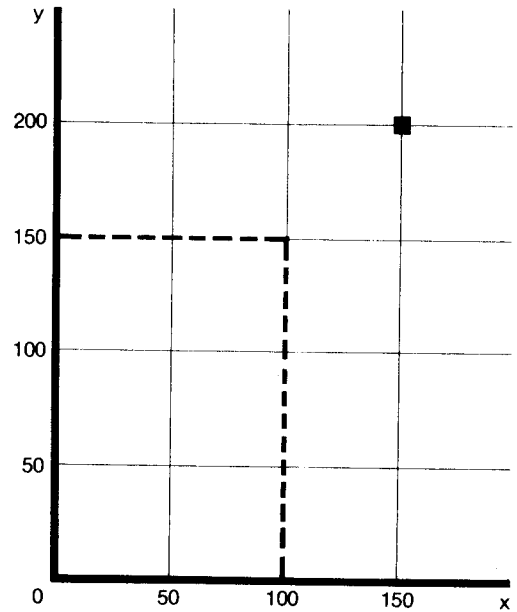
Por lo tanto, para resumir, el CPC464 repetirá las instrucciones que forman el bucle, dando valores sucesivos a la variable de control, en la forma F = 20, 40, 60, 80, etc., hasta que el 'contador F' sea 640, i.e. **pasándose** del valor final marcado. El CPC464 atravesará en esa situación final la instrucción NEXT, y entra a dar otra ronda con el nuevo valor, o se sale del bucle, según procede y ejecutará la línea número 680 para dibujar los largueros que cruzan la valla.

PLOTR
MOVER
DRAWR

Eso es relativo

Antes de que continuemos una pizca más, finalicemos nuestro estudio de los comandos gráficos con la versión **relativa** de PLOT, DRAW y MOVE. Recordarás que estos tres comandos usan como parámetros las coordenadas 'x' e 'y' medidas a partir del **origen** de gráficos que es 0,0 y está situado en la esquina inferior izquierda.

Para los comandos PLOTR, DRAWR y MOVER, los parámetros son los **desplazamientos** requeridos según 'x' e 'y', a partir de la 'vigente' posición del cursor de gráficos.



Si como consecuencia de instrucciones anteriores, tuvieramos como **corriente** posición (la que tiene en ese preciso momento) del cursor de gráficos el punto de coordenadas x=100, y=150; con la instrucción:

mover 5Ø,5Ø ENTER

moveríamos el cursor a la posición de coordenadas x=150, y=200, como puedes ver en el diagrama adjunto.

La misma filosofía se aplica a los comandos DRAWR y PLOTR. Sus parámetros siempre hacen referencia a **coordenadas relativas** contadas desde la posición vigente del cursor, y evitando por tanto al programador tener que calcular las coordenadas **absolutas** (tomadas con respecto al origen 0,0), todas las veces. Su utilidad se hará más patente hacia el final del capítulo.

Abriendo ventanas

No habrás dejado de notar que todavía no hemos terminado con la casa. Después del bucle que dibuja la valla, hay otra instrucción STOP en la línea 685, seguida de otras líneas que incluyen una nueva clave: GOSUB. Ejecuta el programa hasta el final, tecleando CONT, y trata de entender lo que sucede.

Ya era hora de tener ventanas, y todas son iguales. Por eso lo hemos hecho por medio de una **subrutina**, que es el término usado en programación para denominar a una serie de instrucciones -un **párrafo** del programa- que termina con la instrucción RETURN, y ejecuta una **tarea** determinada, que se puede 'encargar' cuando y donde sea necesario. Si no hubieramos usado una subrutina para las ventanas, habríamos tenido que escribir la misma serie de instrucciones 16 veces. GOSUB es contracción de GO (IR) y SUBrutina, y siempre va seguida del número de línea al que queremos **vaya**, para ejecutar la tarea allí descrita; y al encontrarse con RETURN, **vuelva** al punto en que se desvió.

GOSUB

```
10 REM mansion
20 MODE 0
30 CLS
40 REM ** start **
50 BORDER 12
60 INK 0,12:REM amarillo
70 INK 3,3:REM rojo
80 INK 6,6:REM rojo brillante
90 INK 9,9:REM verde
95 INK 15,15:REM amarillo
100 PAPER 0
110 REM dibuja frente
120 MOVE 100,50
130 DRAW 100,250,3
140 DRAW 400,250
150 DRAW 400,50
160 DRAW 100,50
170 REM dibuja costado
180 MOVE 400,250
190 DRAW 600,250
200 DRAW 600,50
210 DRAW 400,50
220 DRAW 400,250
230 REM dibuja faldon
240 REM situado YA en comienzo
250 REM no necesitas 'MUEVA'
260 DRAW 500,350
270 DRAW 600,250
```

```
280 DRAW 400,250
290 REM dibuja tejado
300 REM basta con dos 'trazos'
310 MOVE 100,250
320 DRAW 200,350
330 DRAW 500,350
340 REM dibuja puerta
350 MOVE 225,50
360 DRAW 225,140,6
370 DRAW 275,140
380 DRAW 275,50
390 REM dibuja ventanas
400 REM inferior izquierda
410 MOVE 120,70
420 DRAW 120,130,9
430 DRAW 180,130
440 DRAW 180,70
450 DRAW 120,70
460 REM superior izquierda
470 MOVE 120,170
480 DRAW 120,230
490 DRAW 180,230
500 DRAW 180,170
510 DRAW 120,170
520 REM superior derecha
530 MOVE 320,170
540 DRAW 320,230
550 DRAW 380,230
```

```
560 DRAW 380,170
570 DRAW 320,170
580 REM inferior derecha
590 MOVE 320,70
600 DRAW 320,130
610 DRAW 380,130
620 DRAW 380,70
630 DRAW 320,70
635 STOP
640 REM valla de jardin
650 FOR F=0 TO 620 STEP 20
660 MOVE F,0:DRAW F,60,15
670 NEXT F
680 MOVE 0,45:DRAW 620,45
685 STOP
690 REM marcos de ventanas
700 REM apela al 'traza_cuadrados'
710 size=18
720 MOVE 130,78:GOSUB 900
730 MOVE 156,78:GOSUB 900
740 MOVE 130,103:GOSUB 900
750 MOVE 156,103:GOSUB 900
760 MOVE 130,178:GOSUB 900
770 MOVE 156,178:GOSUB 900
780 MOVE 130,203:GOSUB 900
790 MOVE 156,203:GOSUB 900
800 MOVE 330,78:GOSUB 900
810 MOVE 356,78:GOSUB 900
```

```
820 MOVE 330,103:GOSUB 900
830 MOVE 356,103:GOSUB 900
840 MOVE 330,178:GOSUB 900
850 MOVE 356,178:GOSUB 900
860 MOVE 330,203:GOSUB 900
870 MOVE 356,203:GOSUB 900
880 END
890 REM 'encargado' de trazar cuadrados
895 REM   del SIZE dado al 'citarLE'
900 DRAWR 0,size,9
910 DRAWR size,0
920 DRAWR 0,-size
930 DRAWR -size,0
940 RETURN
```

Ahora puedes ver por qué hemos tenido que abordar los comandos gráficos relativos antes de continuar con esta parte del capítulo. Si tenemos una **subrutina** estándar para una **tarea** concreta, en este caso, dibujar un cuadrado, debe ser posible describir esa tarea de manera 'universal'.

En nuestra subrutina que dibuja ventanas, podemos ver el uso de DRAWR y el **parámetro formal** 'lado' nos permite dibujar un cuadrado con cualquier longitud de lado, y comenzando en cualquier posición de la pantalla.

RETURN

Una subrutina debiera comenzar con una REM como mínimo, para describir lo que hace, en un programa largo con muchas subrutinas, sería tonto no hacerlo. En el capítulo 9 veremos lo importante que es y aprenderemos qué otra información debe dársele. Debes, sin embargo, terminar siempre una subrutina con el comando RETURN (para que **vuelva**).

Cuando el CPC464 se encuentra un comando GOSUB, **toma nota** de dónde está en el programa en ese momento, antes de desviarse y ejecutar los comandos de la subrutina. (Y así cualquiera sabe dónde tiene que **venirse**).

El comando RETURN al final de la subrutina es un poco como la instrucción NEXT de un bucle FOR, pero mientras que NEXT te lleva al comienzo del bucle para ver si hay que dar OTRA ronda, el comando RETURN te hace VOLVER a la instrucción siguiente a la de GOSUB.

Un programa puede usar subrutinas para dibujar una casa de la misma manera que el constructor usa subcontratistas y **agentes** para efectuar algunas de las **tareas.** Oficios como carpinteros, albañiles y fontaneros, son **citados** para hacer labores específicas. Y estos especialistas pueden a su vez, subcontratar ciertas partes de sus tareas. Por ejemplo, un albañil, casi seguro que tendrá un peón para mezclar el cemento y los días en que él se hacía sus propios ladrillos ya se han pasado. Además, el mismo peón puede contribuir a muchas otras labores.

Tú puedes construir los programas de la misma manera. Una subrutina **citada** (reclamada) por un GOSUB, puede a su vez requerir otra subrutina para realizar parte de la tarea, y así sucesivamente. El CPC464 puede 'seguir la pista' a muchos desvíos a subrutinas, en diferentes puntos y siempre encontrará el camino para **venirse** hasta los puntos en que se le mandó desviarse. (Momento, en que -como buen 'Pulgarcito'- dejó sus anotaciones hechas para saber regresar).

Final de tarea

Las subrutinas se ponen siempre al final del programa. (Aunque el sitio mejor para las **citadas** más frecuentemente es ¡al principio! -ya que se tarda menos en **acudir** a ellas). ¿Pero, qué sucede cuando se ha ejecutado el último comando? Bien, si no tomas ninguna precaución, el programa seguirá su curso y se meterá en la primera subrutina, con resultados en ocasiones cómicos, pero habitualmente sin importancia. Una manera de evitar problemas es poner un comando END. Con lo simple que es, como puedes ver en MANSION:

```
880 END
```

Cuando el CPC464 alcanza esa línea, **'fina**liza' el seguimiento del programa y vuelve a READY.

Otro método es colocar un bucle tal y como:

```
880 GOTO 880
```

Lo que le mantiene dando vueltas sin hacer nada, hasta que pulses ESC, o hagas un **'restauro'** general de las condiciones iniciales. Es útil, si no quieres que READY aparezca en la pantalla.

Ejercicios

Intenta usar la subrutina encargada de dibujar las ventanas, para colocar un bonito dibujo de una enorme ventana en el lateral derecho de la casa para poder admirar el jardín.

Escribe una subrutina que dibuje chimeneas, y luego úsala para colocar chimeneas en la izquierda, la derecha o el centro del tejado.

END

Autoevaluación

Esta vez no sólo se te **control**ará sobre los temas tratados en este capítulo, sino que también se te dará una pequeña revisión sobre temas anteriores. Es bastante normal que hayas de repasar el libro para encontrar la respuesta a una pregunta. Nadie puede esperar que recuerdes la cantidad de información que has recibido en tan corto período de tiempo.

9 DISEÑO DEL PROGRAMA

No se puede poner demasiado énfasis en que se requiere tanta habilidad en programación como ciencia y práctica. Recordarás que ya hemos dicho más de una vez, que los ordenadores no piensan. La habilidad en programación de ordenadores es el análisis cuidadoso de un problema, -la labor a desarrollar- y la conversión de esa información objetiva en un programa coherente y lógico que no use demasiado espacio en memoria y que sea lo suficientemente rápido como para **ejecutar** apropiadamente esa labor.

Obviamente, no puedes creer que lograrás eso sin conocimiento y experiencia, pero este capítulo puede ayudarte a adquirir ciertos hábitos y técnicas de buena programación que te servirán muy bien en los años venideros.

Trabajo por objetivos

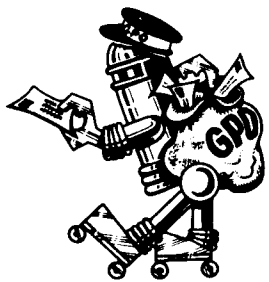
Para cualquier programa que tenga más de, digamos, 20 líneas, hay sólo una manera de abordar la faena. Tienes que **desglosarlo** en trozos manejables. No importa lo bueno que seas, aficionado o profesional, no es posible retener todas las facetas de un programa largo sólo en tu cabeza, y todo a la vez.

Por lo tanto, la primera cosa a hacer cuando se empieza un programa, es dividirlo en partes y detallar lo que se supone hace cada parte. Es conveniente abrir una 'carpeta de proyecto' para registrar todos estos parámetros del diseño para cada programa, y **sus subrutinas.** Si no lo haces así, puede que te encuentres a medio camino en la confección de un programa y te percates que has olvidado la razón para un requisito específico, o la especificación exacta de lo que una subrutina debe hacer.

A partir de la descripción global de lo que el programa hace, debes desglosarlo según una estructura ordenada y lógica de **tareas** concretas y precisas, 'encargables' en los momentos oportunos.

¿Y cómo se hace eso?, te preguntarás. Bien, hay varios métodos en uso, pero vamos a explicar sólo uno de ellos. Utiliza algo llamado "LENGUAJE EN EL DESARROLLO DE PROGRAMAS" -abreviado en inglés como PDL. Vamos a tomar un ejemplo familiar. Un cartero tiene que efectuar una serie de decisiones, acciones y operaciones, para hacer su trabajo. Imagina que tú tienes que programar un 'robot' cartero para hacer la misma cosa. Dado que esos **autómatas** están controlados por ordenadores tendrás que **detallarle** completamente lo que tiene que hacer en todas y cada una de las situaciones en que se pueda encontrar. Así que echa una mirada a lo siguiente:

Programa para un cartero automático




```
MEMOrandum: Ya está situado ante la primera casa de la calle
CON cada casa de la calle, debe hacer:
    SI hay correo para esa casa,
        ENTONCES, HASTA que no quede nada para esa casa:
            prosiga sacando correo de su bolsa.
        'EN DEMAS', váyase a 'otra casa' de la calle.
    SI hay cancela en la entrada del jardín,
        ENTONCES, ábrala; entre; ciérrela.
    MIENTRAS no haya llegado a la puerta de la casa:
        prosiga caminando hacia ella.
    SI hay algún paquete en el correo de esa casa:
        ENTONCES, llame al timbre;
            espere a que le contesten;
            CON cada paquete que haya, debe hacer:
                SEGUN la clase de paquete, elija acción:
                    "certificado", que le firmen la entrega;
                    "sin sellos",  que le paguen el dinero;
                    "voluminoso",  entréguelo en mano.
            OTRO paquete que hubiera, lo trataría igual.
        'EN DEMAS', eche el correo por el buzón
        Dése la vuelta preparándose para salir.
    MIENTRAS no haya llegado a la salida del jardín:
        prosiga caminando hacia ella.
    SI hay cancela en la entrada del jardín,
        ENTONCES, ábrala; salga; ciérrela.
OTRA casa que hubiera en la calle, la trataría igual.

VUELVA donde deba (para seguir 'trabajando')
```

Hemos escrito el plan del programa, usando español bastante ordinario y de una manera formal y organizada (y ¡suena rarillo!).

Hay ciertas cosas importantes que observar en el ejemplo anterior. La primera es que hay un pequeño número de **bloques constructivos** en el programa, algunos de los cuales reconocerán como comandos en "BASIC-ñol" y algunos no. Otra es que el texto está **"sangrado"** para mostrar las acciones que pertenecen a bucles o son cláusulas de instrucciones condicionales. Presta atención cuidadosa a esto cuando escribas tus propios programas, o ¡puede que se conviertan en pura confusión!

Quizás el término **rutina** necesita ser más explicado. Una serie de acciones (o comandos) que efectúa una **tarea** determinada, se colocan separadamente para formar un **párrafo** una **rutina.** Veamos la segunda línea del ejemplo anterior:

```
CON cada casa de la calle, debe hacer:
```

Ese es el **com**ienzo de una labor que será repetida para cada casa de esa calle, ya que las acciones a realizar son iguales para todas las casas.

Dentro de esa labor, hay varias tareas **rutina**rias (sí, ese es el significado 'real' del término). La siguientes es de las típicas:

```
MIENTRAS no haya llegado a la puerta de la casa,
    prosiga caminando hacia ella
```

Puedes ver que esto representa una tarea repetitiva condicionada: Si ha llegado a la puerta debe pararse; si no dará otro paso y volverá a mirar si ha llegado.

La consideración más importante sobre nuestro ejemplo es que no hay que poner demasiada atención en detallar cada una de las acciones en esta etapa.

Si continúas **desglosando** este programa, encontrarás que la acción previa de 'situarlo ante la primera casa' representa montones de instrucciones (incluso sin el problema de ¡cómo hacer que mantenga el equilibrio al andar!). Pero a este nivel, es bastante con una descripción general y simple.

Cuando se desglosa en tareas y acciones sucesivas toda una labor, se llega finalmente a una situación en que cada acción del programa corresponde con una acción individual que concierne a los mecanismos **sensores** o **motores** de nuestro robot.

Si lo anterior te cuesta trabajo imaginártelo, lo siguiente es un pensamiento consolador. A un equipo medio de programadores le tomaría muchos años de duro trabajo, escribir realmente el programa anterior; así que probablemente es buena idea quitarlo de nuestra lista de proyectos. La razón de escoger este ejemplo, es mostrar cómo incluso las tareas más complejas pueden expresarse simplemente con frases 'casi' normales.

Ejercicios

Hay varios **parámetros** de diseño que faltan en el programa anterior. Mira a ver si puedes diseñar las subrutinas que se preocupan de las acciones que debe realizar, cuando

- Nadie responde al timbre
- No hay buzón para cartas en la puerta
- Manos con muchos paquetes y no puede abrir la puerta
- La calle sólo tiene nombres de casas, no números que la identifiquen.

Y probablemente puedas pensar muchas otras **circunstancias.** No te desanimes si no puedes resolver muchas de ellas. Las cosas que para nosotros -seres **humanos**- constituyen una sola 'zancada', puede ser una serie increíble de complicados pasos para esos 'seres **mecanos**'.

Los bloques constructivos

Aquí hay algunos de los bloques que constituyen nuestro programa del cartero, con ejemplos de cómo pudieran traducirse al Amstrad BASIC, con lo que hasta ahora conoces.

CON cada **casa** de la calle, debe hacer:
OTRA **casa** que hubiera en la calle, la trataría igual.

```
800 FOR casa=prime TO ulti
.................
870 NEXT casa
```

SI...
ENTONCES, **hasta** que no quede **nada** para esa casa:
prosiga sacando **correo** de su bolsa.

```
1000 IF... THEN GOSUB 12000:REM a sacar un bulto de la bolsa
1010 IF ultibulto=0 THEN GOTO 1000
```

Suponiendo que la subrutina de la línea 12,000 realiza la tarea de coger un bulto de la saca del correo y asignar a la variable 'ultibulto' el valor 1 cuando ya no hay más para una casa.

```
MIENTRAS no haya llegado a la puerta de la casa:
    prosiga caminando hacia ella.

    2000 IF AnteLaPuerta=1 THEN GOTO 2040
    2010 GOSUB 1300:REM a dar un paso adelante
    2030 IF AnteLaPuerta=0 THEN GOTO 2010
    2040 REM fin de bucle 'andando hasta la puerta'
```

La subrutina de la línea 13000 asignará el valor de la variable 'AnteLaPuerta' según corresponda:

1='Sí' ha llegado; 0='No' ha llegado.

```
SEGUN la clase de paquete, elija acción:
    "certificado", que le firmen la entrega;
    "sin sellos",  que le paguen el dinero;
    "voluminoso",  entréguelo en mano.

    3000 REM ya ha examinado la clase de paquete
    3010 IF clase=1 THEN GOSUB 16000:REM a firmar certificado
    3020 IF clase=2 THEN GOSUB 17000:REM a cobrar sellos
    3030 IF clase=3 THEN GOSUB 18000:REM a entregar en mano
    3040 REM fin de selección de tarea adecuada
```

En una subrutina previamente ejecutada -v.g.: la de sacar un 'bulto' de la bolsa- se habrá visto si el correo es un paquete o no, y la clase de paquete que le corresponde:

1="Certificado"; 2="Sin sellos";
3=Voluminoso

SI hay algún **paquete** en el **correo** de esa casa,
    ENTONCES, llame al timbre;
    EN DEMAS, eche el correo por el buzón.

Y esta sería una posible traducción al BASIC, si en una rutina previamente ejecutada -v.g.: la de sacar un 'bulto' de la bolsa- se ha hecho por ejemplo, que la variable literal **correo$** tenga como valor "paquete" cuando el 'bulto' extraído no sea una carta normal.

```
4000 IF correo$="paquete" THEN GOSUB 20000 ELSE GOSUB 21000
20000 REM llamar al timbre, etc.
.....
20990 RETURN
21000 REM echar por el buzón
.....
21090 RETURN
```

O puede que desees mantener las costumbres adquiridas de hacer con saltos el comando de elección entre alternativas "Si... entonces... en demás" (IF... THEN... ELSE...) en cuyo caso la traducción pudiera ser (aunque es menos aconsejable):

```
4ØØØ IF correo$="paquete" THEN GOTO 41ØØ
.... 
....     Serie de acciones para los paquetes
....
4Ø9Ø GOTO 42ØØ
41ØØ
....     Serie de acciones para los 'no paquetes'
....
42ØØ REM fin de decisión entre alternativa
```

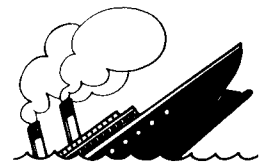
Con labores RUTINArias

Aquí hay una comprobación de lo que deberías preparar antes de escribir una rutina:

- Nombre de la rutina (para tu propio uso, no para el CPC464)
- Nombres de las variables que deben prefijarse antes de **'recurrir'** a la rutina.
- Efecto de la rutina sobre las variables.
- Cualquier efecto **colateral** de la rutina.

Cuando diseñas programas, es buena idea usar hojas separadas para cada tarea, que posteriormente pueden ser encuadernadas en carpetas de anillas formando tu proyecto. Cada hoja debiera tener escrita la información anterior en lugar visible. Eso tiene dos ventajas:

1. Puedes ver de una ojeada la tarea que intentas efectuar con la subrutina.
2. Es más fácil de identificar dos subrutinas muy similares que pudieran combinarse en una sola y ahorrarte el hacer dos veces la misma cosa.



¿Recuerdas las rutinas que dibujaban las ventanas de la casa? Comenzaba con una REM, para **'memo**randum':

REM subrutina para cuadrado

La única variable que debe estipularse antes de **citar** la subrutina era 'lado', y la subrutina simplemente usaba la variable, pero no la alteraba en nada. Finalmente, la rutina no tiene ningún otro efecto sobre ninguna otra parte del programa ni sobre el propio CPC464.

Diseñando las rutinas de esa manera, hacemos que sean **'estancas'**. Sabe exactamente qué hacer para usarlas, sabes la tarea que realizan, y sabes dónde pueden afectar a algo.

No solamente los programas diseñados de esta manera son mucho más fáciles de **depurar;** sino que tales diseños evitan tener resultados inesperados desparramados por todas partes, hasta que todo el programa se hunda más rápidamente, aunque no sea tan catastrófico generalmente como lo del 'Titanic'.

Diseñando programas usando este método de 'rutina por rutina', significa que en el momento que has acabado, terminas con una gavilla de hojas que te dice exactamente lo que debes escribir.

Documentación

Como programador aficionado, probablemente trabajarás sólo para ti mismo. Así, cualquier programa que escribas tenderá a ser una orgullosa posesión que no estés dispuesto a revelar al mundo exterior, ni aceptes verlo sujeto a críticas.

Los programadores profesionales no trabajan así. La mayoría de los profesionales trabajan en equipo sobre el mismo programa, o **cadenas** de programas, y por lo tanto deben ser capaces de leer y comprender lo que cualquier otro ha escrito, sin mucha dificultad.

BASIC es un lenguaje accesible y tolerante que no insiste en ninguna estructura rígida o manera de programar. Todo eso está muy bien cuando eres un principiante, pero en cuanto las tareas se hacen un poco más complejas, es necesario comenzar a confeccionar los programas para que sean claros y fáciles de comprender. **En** los **demás** casos, te encontrarás que

estás volviendo a reescribir los programas que terminaste hace un mes, simplemente porque ya no puedes sacar provecho de ellos, ni entiendes como pudiste hacer eso, y menos cambiarlos.

El secreto del éxito es programación estructurada y REM. Y REM y REM otra vez.

Cuando tienes un salto (GOTO...) o un **desvío** (que es un salto de ida y vuelta: GOSUB) en tu programa, simplemente pon una REM para decir por qué y adónde vas. Poniendo GOSUB 5000, por ejemplo, es aceptable hasta que haya 50 subrutinas diferentes. Cuando así es el caso, necesitas escribir (también por ejemplo):

GOSUB 5000:REM A situar cartero

Hemos hablado bastante claramente, definiendo lo que una rutina tiene que hacer y cuáles son las variables **-argumentos, parámetros y resultados-** involucradas. La manera de hacerlo dentro de un programa es poner toda la información apropiada en REM, al comienzo de cada rutina. Esto no tiene mucho efecto sobre la velocidad de ejecución, y tiene efectos mágicos en la legibilidad del programa, tanto por ti mismo como por otros programadores.

10 FANTASTICOS SONIDOS

Cuando hemos dicho antes que los programas en BASIC del CPC464 pudieran no funcionar directamente en otros ordenadores, fue especialmente teniendo en cuenta los comandos de sonido. El surtido y la potencia de los comandos de sonido del BASIC CPC464 son tales que muy pocos otros ordenadores pueden igualarlos.

Pero eso también crea un problema. Sería muy posible escribir un libro separado sólo sobre los comandos de sonido.

Así que, manteniéndonos en la intención de aprender los primeros pasos en Amstrad BASIC, hemos dejado fuera de este capítulo un montón de información.

Habiendo dicho eso, carga el siguiente programa del cassette. Se llama "SONIDOS", y te dará una idea de la 'zapatiesta' que puedes conseguir con estos comandos de sonido. Un listado de este programa se da al final de este capítulo, así que puedes copiar los trozos interesantes e incluirlos en tus propios programas.

Afinándolo

Como puedes haber adivinado, el comando que hace que **suene** es SOUND. Teclea lo siguiente:

sound 1,478 ENTER

Eso es la C media. El número 478 se llama el **período**, y determina el **tono** de la nota que sonará. Una lista completa de las notas y de sus períodos correspondientes, se da en la Guía del Usuario Amstrad CPC464. El 1 es el **canal** de sonido que se va a usar. La Parte 2 de este curso describe cómo usar los otros dos canales.

Ahora teclea lo siguiente:

sound 1,478,200 ENTER

Cuando esta vez pulses la tecla ENTER, el sonido se oirá durante precisamente 2 segundos. Esto está gobernado por el valor 200 que fija la **duración** del sonido en centésimas de segundo.

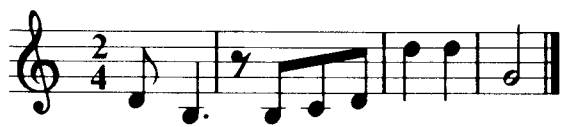
Si omites este parámetro del comando, el CPC464 supone que la duración que deseas es 20. Puedes ahora escribir un programa para hacer sonar una tonada familiar. Por ejemplo:

SOUND

```
10 rem Una Tonada Familiar
20 sound 1,213
30 sound 1,253,60
40 sound 1,0,40
50 sound 1,253
60 sound 1,239
70 sound 1,213
80 sound 1,127,40
90 sound 1,0,1
100 sound 1,127,40
110 sound 1,159,60
120 end
```

Las líneas 40 y 90 son interesantes. Para dar un 'silencio' (**pausa** en terminología musical) entre dos notas, debes poner un comando cuyo período es 0. En la línea 40 tenemos una pausa por dos "semitiempos"; en la línea 90 una pausa extremádamente breve nos asegura que la nota repetida no se confunda con la primera.

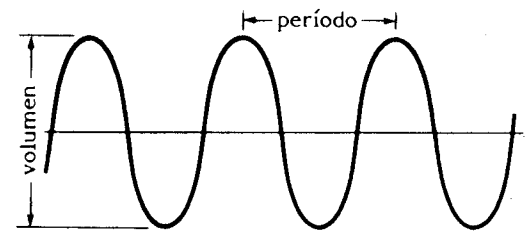
Una Tonada Familiar



Para aquéllos de vosotros que estáis familiarizados con la música, la misma tonada la mostramos arriba en notación normal para que podáis compararla con el programa.

Sonidos BASIC

Antes de que prosigamos, es necesario echar una mirada a cómo están construidos los sonidos. En su forma pura, un sonido puede representarse gráficamente por una onda sinusoidal, como la siguiente:

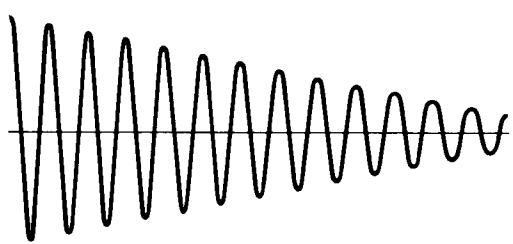


Quizás ahora sea patente lo que el parámetro de **período** en el comando SOUND, significa realmente. Es el tiempo que transcurre entre dos oscilaciones consecutivas, medidos en intervalos de 8 **microsegundos** (millonésimas de segundo). Cuanto más corto sea el tiempo entre las oscilaciones, más agudo será el sonido; y cuanto más largo sea el tiempo, más grave será el sonido.

La altura de la onda se conoce como **volumen,** y podemos especificarlo también en el comando SOUND. Intenta teclear de nuevo la C media, sólo que esta vez pon un 2 al final, en la forma siguiente:

sound 1,478,2ØØ,2 ENTER

Eso lo hace muy quedo, ¿verdad? El volumen puede especificarse entre 0 y 7, con el 0 significando ningún volumen en absoluto (y ya veremos que puede ser útil) y 7 indicando máximo volumen. El sonido que hemos examinado comienza y termina con el mismo volumen. En el mundo real las cosas no son habitulamente así. Es bastante normal tener un sonido cuyo volumen cambia desde que arranca hasta que termina.

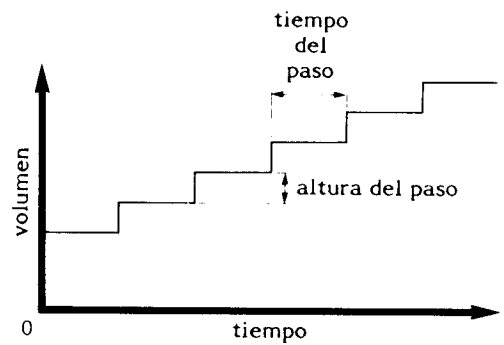


Hacemos eso en el CPC464 con el comando ENV, que te da una **envolvente** que permite variar el volumen del sonido mientras dura la nota. Teclea las dos instrucciones siguientes y luego ejecútalas:

ENV

```
1Ø env 1,6,1,3Ø
3Ø sound 1,478,18Ø,1,1
```

Probablemente necesitarás estudiar minuciosamente el siguiente diagrama para comprender cómo funciona esto. No te preocupes sobre la línea 20 que falta; ya la veremos a su hora.



Puedes tener hasta 15 diferentes envolventes de volumen en un programa, y el primer 1 de la instrucción ENV anterior, simplemente la identifica como la primera. El siguiente número, el 6, significa que quieres que el volumen cambie en 6 **pasos** iguales. El siguiente número, el 1, da la cantidad que debe crecer el volumen al principio de cada uno de los pasos. Si fuera -1, el volumen decrecería.

Y finalmente, el 30, da la duración del paso en centésimas de segundo.

Ten cuidado sobre el volumen especificado en el comando SOUND. El CPC464 puede realmente fijar el volumen a 16 niveles diferentes, cuando se especifica envolvente de volumen. La gama entonces es de 0 a 15, siendo los niveles 0, 2, 4, 6, 8, 10, 12 y 14, los correspondientes a la gama 0 a 7 que puedes conseguir cuando no se especifica envolvente de volumen.

ENT

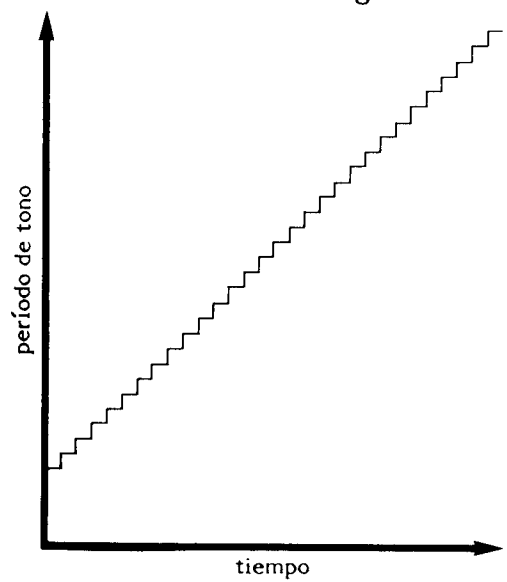
Analicemos ahora cómo funcionan conjuntamente SOUND y ENV en nuestro ejemplo. El volumen se fija a 1 en el comando SOUND, y ENV 1 exige 6 pasos de volumen creciente en un nivel, así que el volumen final será 1+6=7. Nota que el primer paso se efectúa inmediatamente que se ejecuta el comando SOUND, así que el volumen **inicial** es 1+1=2. Observa también que los 6 pasos de la envolvente multiplicadas por la duración del paso, no deberá ser mayor que la duración especificada en el comando SOUND, pues en caso contrario, se perdería el final de la envolvente.

Veamos ahora la línea número 20 que faltaba. De la misma manera que los niveles de volumen en un comando SOUND se modifican por una envolvente de volumen, la frecuencia del sonido puede modificarse por una **envolvente de tono.** El comando usado es ENT. Así que pongamos la línea 20 en su lugar:

```
10 env 1,6,1,30
20 ent 1,180,1,1
30 sound 1,478,180,1,1,1
```

Como puedes ver, nuestro comando SOUND ha recibido otro 1 más para indicar que se va a usar la ENT 1.

La estructura del comando ENT es muy similar a ENV, como puedes ver del diagrama siguiente. Pero esta vez, sin embargo, es el **período de la nota** la que aumenta en 1 cada vez, siendo 180 el número de pasos, cada uno con una duración de una centésima de segundo.



Si todavía no lo has hecho, ejecuta el programa anterior varias veces. Luego cambia los valores dados al paso en los comandos ENV y ENT, a -1 en lugar de 1; cambia a 7 el volumen inicial fijado en el comando SOUND, y vuélvelo a ejecutar.

Ruidos

El CPC464 puede añadir un poco de **ruido** a cualquier sonido, dándole interés adicional.

Puedes pensar que el comando SOUND ya se parece a un árbol de Navidad, dado que le hemos añadido los números para ENV y ENT. Esta es realmente la última cosa que puedes empalmarle: el ruido. Corrige la línea 30 de nuestro programa para que sea:

```
30 sound 1,478,180,1,1,1,5
```

Si ejecutas ahora el programa, observarás bastante diferencia.

Intenta experimentar con las 31 diferentes clases de ruido que pueden especificar poniendo el número apropiado en esa parte del comando. Dejándolo sin poner, o poniendo 0, significa ruido con cero volumen.

Ejercicios

Añade envolventes de tono y volumen a la 'tonada familiar' y luego cámbialos para conseguir diferentes sonidos. Luego añade diferentes clases de ruido a algunos de los comandos de sonido.

Escribe tu propio programa para un pequeño pasaje de una melodía familiar. Incluso aunque no sepas mucho sobre música, es posible hacer bastante trabajo mediante **'ensayo y error'**. Si no sabes sobre música, no seas demasiado ambicioso al principio. El ritmo de una "nana" es suficiente para empezar.

Hora de tocar

Bien, no es realmente eso. El siguiente programa en la datocassette se llama ORGANO. No solamente puedes interpretar directamente melodías con el teclado, sino que tienes la oportunidad de analizar el programa para ver cómo puede hacerse. Deberás comenzar por comprender lo suficiente de BASIC para listar y examinar lo que se supone que el programa hace, aunque este programa en particular contiene una cierta cantidad de **programación avanzada.** No temas ensayar cambiando cosas para ver lo que sucede.

Como prometimos, aquí está el listado del programa SONIDOS:

```
10 REM Zappow
20 REM by Dave Atherton
30 MODE 1
35 INK 0,1:INK 1,25
40 LOCATE 12,4:PRINT"SONIDOS DEMO"
50 LOCATE 10,7:PRINT"1. Explosion"
60 LOCATE 10,8:PRINT"2. Ladridos"
70 LOCATE 10,9:PRINT"3. Sirena"
80 LOCATE 10,10:PRINT"4. Cisterna"
90 LOCATE 10,11:PRINT"5. Cuco"
100 LOCATE 10,12:PRINT"6. Ametralladora"
110 LOCATE 10,13:PRINT"7. Invasores del espacio"
120 LOCATE 8,17:PRINT"Elige un sonido de 1 a 7"
130 IF INKEY$>"" THEN 130
140 a$=INKEY$:IF a$="" THEN 140
150 IF a$<"1" OR a$>"7" THEN 140
160 a=VAL(a$)
170 LOCATE 20,19:PRINT a$
180 IF a=1 THEN GOSUB 280
190 IF a=2 THEN GOSUB 330
200 IF a=3 THEN GOSUB 380
210 IF a=4 THEN GOSUB 430
220 IF a=5 THEN GOSUB 480
230 IF a=6 THEN GOSUB 530
240 IF a=7 THEN GOSUB 580
250 FOR j=0 TO 1000:NEXT
260 LOCATE 20,19:PRINT " "
```

```
270 GOTO 130
280 REM Explosion
290 ENV 1,11,-1,25
300 ENT 1,9,49,5,9,-10,15
310 SOUND 1,145,255,0,1,1,12
320 RETURN
330 REM Dog Bark
340 ENV 1,4,7,10
350 ENT 1,7,-8,3,6,24,2
360 SOUND 1,120,33,8,1,1,3
370 RETURN
380 REM Siren
390 ENV 1,2,9,45
400 ENT 1,2,9,45
410 SOUND 1,150,90,6,1,1
420 RETURN
430 REM Toilet Flush
440 ENV 1,3,-2,85
450 ENT 1,5,-1,51
460 SOUND 1,150,254,11,1,1,8
470 RETURN
480 REM Cuckoo
490 ENV 1,4,12,11
500 ENT 1,5,12,8
510 SOUND 1,165,40,13,1,1
520 RETURN
530 REM Machine Gun
```

```
540 ENV 1,21,-5,4
550 ENT 1
560 SOUND 1,162,82,15,1,1,11
570 RETURN
580 REM Space Invader
590 ENV 1,4,30,19
600 ENT 1,9,49,5,1,-10,26
610 SOUND 1,136,68,15,1,1,0
620 RETURN
```

Autoevaluación

Sería bastante sorprendente que no tuvieras que volver a repasar concienzudamente este capítulo para contestar las preguntas en TAE10. No dejes que eso te preocupe. Si ya supieras todo sobre los comandos de SONIDO del CPC464, no estarías leyendo este libro.

11 TRABAJANDO CON NUMEROS

Como ya hemos dicho varias veces, el CPC464 como cualquier otro ordenador, es bueno ante **tareas repetitivas** y aburridas. También es bueno para tratar con números. De hecho, es la única cosa en la que **es** bueno. Todo lo que hemos visto se consigue mediante el procesado interno en el CPC464 de montones de números a velocidad increíble. Incluso las funciones de gráficos y sonidos funcionan de esa manera.

Los números son el alimento de los ordenadores. Si no eres especialmente bueno con las matemáticas, este capítulo no será fácil. Pero en todo caso dále una pasada. La aritmética simple puede ser interesante, y BASIC es bastante directo en la manera en que la presenta. En cualquier caso, las materias realmente difíciles, no se comentan en esta parte del curso.

Aritmética en BASIC

El CPC464 hace sus operaciones usando las cuatro funciones aritméticas familiares, pero dos de ellas tienen diferente símbolo:

Operación	Significado	BASIC
+	sumar	+
-	restar	-
x	multiplicar	*
÷	dividir	/

Por tanto, las operaciones escritas en BASIC parecen algo diferentes de lo que estamos acostumbrados.

En el capítulo 4, tecleamos:

print 2+2

Puedes hacer la misma cosa con las siguientes operaciones consultando la lista anterior, y usando los símbolos correctos en BASIC:

3 + 7	17 - 8
15 x 4	15 ÷ 3
8 ÷ 2	20 x 17

Para operaciones más complicadas, tenemos que decirle al CPC464 que ciertas operaciones ha de efectuarlas antes que otras. Por ejemplo, es perfectamente claro en la aritmética ordinaria lo siguiente:

$$\frac{15 \times 7}{7 - 2}$$

En BASIC eso **no** se escribe:

15*7/7-2 [=13 falso!]

Tenemos que indicar que la multiplicación 15*7 y la resta 7-2 deben realizarse antes de la división. Eso se hace encerrando las operaciones entre **paréntesis,** como sigue:

(15*7)/(7-2) [=21 cierto!]

Con eso ya serás capaz de hacer las operaciones siguientes:

$$\frac{30}{3 + 2} \qquad \frac{15 \times 6}{9}$$

$24 - (4 \times 3)$ $5 \times (2 + 8)$

Se aplica la misma regla a las **variables** que a las **constantes.** Teclea el siguiente programa y continúa cambiando la línea 40, de manera que puedas resolver las otras operaciones indicadas ulteriormente. No olvides usar los símbolos correctos y colocar los paréntesis en su sitio.

```
10 let a=2
20 let b=5
30 let c=10
40 print a+b+c
```

$a \times a$ $\qquad \frac{b \times c}{b - a}$

$c - (a \times b)$

$\frac{c}{b - a}$ $\qquad \frac{c - b}{c + b}$

Las dos primeras operaciones son lo suficientemente directas, pero ¿qué piensas de las respuestas a las otras tres?

Has tenido una simple perspectiva sobre la manera en que el CPC464 opera en matemáticas. Si cuentas las cifras antes y después del **punto decimal** (equivalente a la coma que tú usas) de las últimas tres respuestas, verás que siempre son 9. Esa es la manera en que el CPC464 normalmente muestra los números en la pantalla. La faena es que algunas veces una serie de cálculos dará un resultado ligeramente inexacto. En lugar de 5.0 conseguirías:

5.ØØØØØØØ1

o bien:

4.99999999

Curioso, ¿verdad? Afortunadamente no necesitamos a menudo tanta exactitud.

Algunas veces, de hecho, sólo queremos números enteros, dado que la diferencia entre 276.25 y 276, por ejemplo, no es a menudo muy significativa. La clave que podemos usar para dar un resultado como un número entero es ROUND, y lo REDONDEARA. La función típica sería:

x=round(26*17)/1.6

Ensáyalo. Si la parte anterior al punto decimal es menor de 0.5 el resultado es redondeado hacia **abajo** para conseguir el siguiente número entero; y si es 0.5 o más, se redondea hacia **arriba.** (i.e. da el Número Entero más próximo al resultado de la operación).

Otra cosa que debes recordar, es que aunque el CPC464 sólo muestra en la pantalla números con una **precisión** de 9 **cifras** en total, puede tener incluso más **"dígitos"** en la memoria. Por ejemplo, puedes terminar con un valor de:

5.ØØØØØØØØ1 (diez cifras)

El CPC464 insistirá en que eso es 5.0 cada vez que se lo preguntes! Pero, y eso es importante, si el siguiente comando es algo como:

IF a=5 THEN GOTO ....

¡GOTO nunca se ejecutará!

Una técnica estándar en programación para evitar tales problemas consiste en no usar nunca la línea anterior, sino más bien la siguiente:

vepsilon=0.000001
IF ABS(a-5)<vepsilon THEN GOTO ....

No te preocupes sobre la clave ABS. Simplemente da el valor **absoluto** de la diferencia entre 'a' y 5, como veremos en la parte 2.

El siguiente programa en la cassette A se llama "TABMULT", y es una buena **"chuleta"** para las tablas de multiplicar así como para ejemplos de matemáticas simples con el CPC464. ¡Pídele que te dé la tabla de multiplicar por 13.87! Después de ejecutar este programa, merecerá que pases un rato listándolo en la pantalla del monitor y viendo cómo ha sido confeccionado.

Elemental y lógico

Hay un montón de cosas que mencionamos en capítulos anteriores sin explicarlas en detalle, siendo la razón, simplemente, que era demasiado pronto para explicarlas en ese momento y que más bien te habrían confundido en lugar de ayudarte. Una de esas es el uso de **operadores lógicos.**

Pon tu mente de nuevo en nuestro estudio de **bucles.** Usábamos signos como '<' y '>'. Hay todo un conjunto completo de esos símbolos, que se llama 'operadores lógicos'.

<	menor que
>	mayor que
=	igual a
<>	distinto de
<=	menor que o igual a
>=	mayor que o igual a

Incluso aunque los términos no te sean familiares, está bastante claro lo que pueden significar.

Cuando necesitas **comparar** dos valores de manera que sepas cuándo terminar un bucle o decidir entre dos cursos de acción, estos operadores serán el medio para poder discriminar si la **condición** de comparación se cumple - es **cierta,** o bien no se cumple - es **falsa.** Los comandos IF estudiados en el Capítulo 7 dependen completamente de estos operadores.

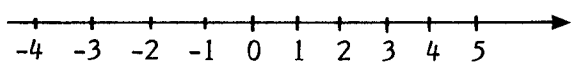
Una sutileza que debes recordar sobre los operadores lógicos, es que tienen en cuenta si un número es negativo o positivo; de manera que si a = 5 y b = 3:

a>b es cierto

Pero si a = -5 y b = 3, lo cierto es que:

a<b

ya que debes recordar en el "eje de números":



los negativos están **antes,** i.e.: son **menores** que los positivos, que están **después,** i.e.: son **mayores.**

Cadenas lógicas

¿Recuerdas que comenzamos diciendo que el CPC464 sólo sabía funcionar con números? En ese caso, te preguntarías cómo almacenar y procesar **caracteres** adosados unos a otros, en forma de **sartas** o **cadenas,** como aprendimos en el capítulo 6. Decimos que son **literales.**

La respuesta es que cada carácter está identificado por un número que se llama su **código del carácter,** de manera que las 'retahilas' de caracteres se tratan simplemente tomando las series de **códigos numéricos** equivalentes. Por lo tanto, los operadores lógicos pueden comprobar si dos valores **literales** están en orden alfabético, 'A' siendo menor que 'Z'; pero es debido al valor numérico de los correspondientes códigos de carácter. Si ensayamos el siguiente comando:

IF "naranjas" < "limones" THEN PRINT "pepinos" ELSE PRINT "mandarinas"

lo expuesto nunca sería 'pepinos' dado que 'naranjas' está detrás de 'limones' alfabéticamente.

Todas las letras mayúsculas son menores que cualquiera de las minúsculas, y un literal corto es menor que uno largo que comience de la misma manera. Por lo tanto:

"mano"<"mono"	es cierto
"CAJA"<"caja"	es cierto
"marzo"<"abril"	es falso
"Zoo">"Zoología"	es falso
"pera">"Peral"	es cierto

En este caso, el CPC464 está **cotejando** los valores 'numéricos' de cada letra de la palabra con los valores 'numéricos' de la letra que ocupa la misma posición en la palabra situada al otro lado del operador lógico. Realmente es como lo hace, pero coincide con el habitual **orden alfabético,** teniendo en cuenta que todas las **mayúsculas** van **antes** (i.e.: son **menores**) que todas las **minúsculas.**

Casa y jardín

La casa está ahora en buen estado. Tiene sus ventanas, su chimenea, y una valla que aleja al perro del vecino. Todo lo que necesita ahora es un poquito de trabajo en la huerta.

Siendo el CPC464 una máquina de las prácticas, no tiene mucha utilidad para planificar un seto con flores atractivas. Pero sí puede ayudarte a hacer los planes para tu cosecha de verduras. Así que carga el siguiente programa de la cassette A que hemos llamado eufemísticamente HUERTO.

El jardín tiene 6 metros de largo por 4 metros de ancho, y vamos a plantar las cosas en hileras de 4 metros. Las diferentes verduras necesitan esparcirse en hileras de diferente anchura, y cada una dará un diferente peso de producto por metro cuadrado cultivado. Cuando ejecutes el programa, te preguntará cuántas hileras quieres de cada verdura, y te dirá si te queda espacio sólo para otra hilera o para dos.

Suponiendo un buen suelo, y un clima razonable en ese año, puedes sacar un resumen que te muestre el peso de producto que puedes esperar de cada verdura. El listado para ese programa lo mostramos después. Como puedes ver, las matemáticas son bastante simples, y se basan en la siguiente tabla:

Vegetal	Anchura (metros)	Producto hilera (kg.)
Cebollas	0.30	9
Zanahorias	0.30	3
Patatas	1.0	50
Coles	0.60	8
Judías	1.0	30
Espinacas	0.50	11

¿No te gustan las espinacas? Intenta alterar este programa para incluir lo que te guste.

```
10 REM HUERTO
20 MODE 1
30 INK 0,0:BORDER 0
40 INK 1,26
50 length=6
60 CLS
70 PRINT"HUERTO"
80 PRINT"Longitud que queda :";length;"metros"
90 PRINT".Que verdura quieres cultivar..."
100 PRINT
110 PRINT"              anchura   produccion  filas"
120 PRINT"1. CEBOLLAS   0.3m        9Kg       ";onions
130 PRINT"2. ZANAHORIAS 0.3m        3Kg       ";carrots
140 PRINT"3. PATATAS      1m       50Kg       ";potatoes
150 PRINT"4. COLES      0.6m        8Kg       ";cabbages
160 PRINT"5. JUDIAS       1m       30Kg       ";beans
170 PRINT"6. ESPINACAS  0.5m         1Kg       ";parsnips
180 PRINT
190 PRINT"Elige un numero entre 1 y 6"
200 PRINT"o 7 para mostrar los totales de produc- cion"
210 INPUT veg
220 IF veg<=0 OR veg>7 THEN GOTO 190
230 IF veg=7 THEN GOTO 320
240 IF veg=1 THEN GOSUB 470
250 IF veg=2 THEN GOSUB 540
260 IF veg=3 THEN GOSUB 610
270 IF veg=4 THEN GOSUB 680
```

```
280 IF veg=5 THEN GOSUB 750
290 IF veg=6 THEN GOSUB 820
300 PRINT '---------------------------------------'
310 GOTO 60
320 REM Summary
330 CLS
340 PRINT'RESUMEN'
350 PRINT'PRODUCCION DEL HUERTO EN KILOS'
370 PRINT
380 PRINT'Cebollas  :';onions*9;'Kg'
390 PRINT'Zanahorias:';carrots*3;'Kg'
400 PRINT'Patatas   :';potatoes*50;'Kg'
410 PRINT'Coles     :';cabbages*8;'Kg'
420 PRINT'Judias    :';beans*30;'Kg'
430 PRINT'Espinacas :';parsnips*11;'Kg'
450 GOTO 450
460 :
470 REM Onions
480 PRINT'Cebollas'
490 rowwidth=0.3
500 GOSUB 890
510 onions=rows
520 RETURN
530 :
540 REM Carrots
550 PRINT'Zanahorias'
560 rowwidth=0.3:produce=3
```

```
570 GOSUB 890
580 carrots=rows
590 RETURN
600 :
610 REM Potatoes
620 PRINT"Patatas"
630 rowwidth=1
640 GOSUB 890
650 potatoes=rows
660 RETURN
670 :
680 REM Cabbages
690 PRINT"Coles"
700 rowwidth=0.6
710 GOSUB 890
720 cabbages=rows
730 RETURN
740 :
750 REM Beans
760 PRINT"Judias"
770 rowwidth=1
780 GOSUB 890
790 beans=rows
800 RETURN
810 :
820 REM Parsnips
830 PRINT"Espinacas"
```

```
840 rowwidth=0.5
850 GOSUB 890
860 parsnips=rows
870 RETURN
880 :
890 REM Details
900 INPUT".Cuantas filas quieres plantar";rows
910 testlength=length-rows*rowwidth
920 IF testlength<0 THEN PRINT"No hay espacio":GOTO 900
930 length=testlength
950 RETURN
```

Autoevaluación

Realmente debes trabajar con TAE11 para conseguir un montón de práctica en tratar con los **operadores matemáticos** y los **operadores** "lógicos" (de **relación)** del CPC464. La mayoría de los programas requieren una cierta cantidad de matemáticas elementales para que puedan ser de alguna utilidad, así que merece la pena emplear todo el tiempo que sea posible en comprender este aspecto del CPC464. No olvides que la mayoría de los programas de juegos que puedes comprar para el CPC464 dependen de algunas operaciones lógico-matemáticas a muy alta velocidad.

12 ACTIVIDADES RECREATIVAS

La mayoría de **los juegos**, y no sólo los juegos de ordenador, se basan en oponer tus habilidades físicas y mentales contra otros jugadores, contra sucesos de azar o contra el reloj. Algunas veces tienes que jugar contra los tres. Un programa de ajedrez o un juego de derribar muros te hará pasar muchas horas de diversión de esa manera, y no necesitas saber nada sobre la programación de juegos. Pero, con eso no basta.

Si tú quieres **diseñar** juegos, es necesario, sin embargo una palabra de aviso.

Vuelve a **estudiar bien** tus libros de matemáticas.

No puedes esperar poder programar el rebote de una pelota en una pared, si no recuerdas las fórmulas para calcular el ángulo de rebote, la velocidad y la trayectoria. A propósito, no vamos a cubrir ese nivel de detalle en esta etapa del curso, pero debes darte cuenta que los dos juegos dados en este capítulo, cubren sólo una diminuta fracción de lo que el CPC464 puede realmente hacer.

RND

Suerte y azar

Cuando jugaste a BOMBARDERO en el Capítulo 4, la nave extraterrestre siempre aparecía a la vista en lugares diferentes. Eso se hizo mediante la función cuya clave es RND (abreviatura de **Ran**dom=Aleatorio), que es útil para incluir al **azar** dentro de un programa de ordenador.

Lo siguiente es un ejemplo de la manera en que puede usarse la función RND:

```
move rnd*639,rnd*399
```

RND te da un **número fraccionario** entre 0 y 1, de manera que tienes que multiplicarlo por el máximo número que requieres en el comando o en la rutina. En el caso anterior, el cursor de gráficos terminará en una posición aleatoria de la pantalla, dado que hemos multiplicado RND por las **máximas** coordenadas 'x' e 'y'.

Intenta escribir un breve programa para colocar un pequeño cuadrado en una posición de la pantalla determinada por la 'suerte'.

Pasó la hora

Otra función útil en el CPC464, y no sólo para juegos, es TIME. Desde el momento en que se pone en marcha el CPC464 o se **restaura**, empieza a contar el TIEMPO transcurrido en intervalos de tres centésimas de segundo, y deposita la cuenta en una variable llamada TIME. Esta cuenta sólo se suspende mientras se carga un programa o se guarda en cassette. El siguiente ejemplo muestra cómo puede usarse:

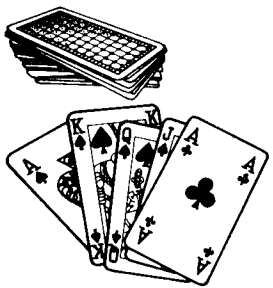
```
10 print "Pulsa cualquier tecla"
20 if inkey$="" goto 20
30 for t=1 to rnd*5000:next
40 a=time
50 print "pulsa OTRA vez"
60 if inkey$="" goto 60
70 b=time
80 print "Tiempo de reacción=";
   (b-a)/300;" segundos"
90 end
```

Lo que estamos haciendo aquí es examinar el valor de TIME antes y después de que la frase "pulsa OTRA vez" aparezca en la pantalla; siendo el tiempo de reacción la diferencia entre los dos valores registrados.

Habrás observado también otra nueva palabra clave en las líneas 20 y 60, INKEY$. Es similar a INPUT exceptuando que es una **función (no** un **comando)** y que sólo te entrega el carácter que se haya pulsado en el teclado, sin necesidad de que vaya seguido de la pulsación de ENTER. Aquí lo estamos usando para (INdagar) si se ha pulsado **cualquier tecla** ("key"=tecla, "in"=hacia **dentro**), entendiendo por cualquier tecla todas exceptuando SHIFT, CTRL, CAPS LOCK y ESC.

INKEY$

TIME



BLACKJACK (A las 21)

Jugar a las cartas con el CPC464 es más divertido que jugar solitarios contigo mismo. El siguiente programa en la cassette A es BLACKJACK. En caso de que no sepas el juego, la idea es continuar pidiendo cartas hasta que su valor sea lo más cercano posible a 21. Pero si sacas más de 21, te 'has pasado' y pierdes esa mano. Si no, el CPC464 saca a continuación una mano e intenta conseguir una puntuación mayor que la tuya sin pasarse.

Otras reglas son:

- "Cinco" cartas que sumen 21 o menos son superiores a cualquier baza
- Los ases pueden valer bien 1 o bien 11
- Las cartas con figura cuentan como 10

Aunque hay varios comandos que no estudiaremos hasta la Parte 2 de este curso, el listado del BLACKJACK se muestra a continuación para que lo puedas empezar a investigar por ti mismo.

```
10 REM ** BLACKJACK **
20 REM
30 REM ** inicio **
40 MODE 1:BORDER 4
50 INK 0,17:INK 1,0
60 LOCATE 16,5:PRINT "BLACKJACK"
70 LOCATE 7,12
80 PRINT"Pulsa una tecla para empezar"
90 suit$=CHR$(226)+CHR$(227)+CHR$(228)+CHR$(229)
100 card$="A23456789TJQK"
110 C5$="Truco de las 5 cartas - Gano"
120 myace=0:youace=0
130 mygames=0:yourgames=0
```

```
140 WHILE INKEY$="":WEND
150 CLS
160 REM ** turno HUMANO **
170 yourcards=0:yourace=0
180 yourhand=0
190 LOCATE 18,1
200 PRINT"Partidas Yo:";mygames:
210 PRINT "Tu:";yourgames
220 y=20:x=5
230 yourcards=0
240 GOSUB 770
250 yourcards=yourcards+1
260 IF value=1 THEN yourace=yourace+1
270 yourhand=yourhand+value
280 GOSUB 830:x=x+5
290 IF yourhand>21 THEN GOTO 690
300 GOSUB 930
310 oneacehand=yourhand
320 IF yourace>=1 THEN oneacehand=yourhand+10
350 IF yourcards=5 THEN 440
360 IF yourhand=21 THEN 440
370 IF oneacehand=21  THEN 420
380 IF yourace=0 AND yourhand<=11 THEN 240
390 IF yourcards=1 THEN 240
400 INPUT".Quieres otra carta (S/N)";q$
410 IF UPPER$(q$)="S" THEN GOTO 240
420 IF oneacehand<=21 THEN yourhand=oneacehand
```

```
440 REM ** turno MECANO **
450 y=10:x=5
460 myhand=0:mycards=0:myace=0
470 GOSUB 770
480 mycards=mycards+1
490 IF value=1 THEN myace=myace+1
500 myhand=myhand+value
510 GOSUB 830:x=x+5
520 FOR delay=0 TO 1000:NEXT
530 IF myhand>21 GOTO 720
540 IF mycards=5 THEN GOSUB 930:PRINT C5$:GOTO 710
550 IF yourcards=5 THEN 470
560 mineA=myhand
570 IF myace>=1 AND myhand<12 THEN mineA=myhand+10
600 IF myhand>=yourhand THEN 640
610 IF mineA>=yourhand THEN myhand=mineA:GOTO 640
630 GOTO 470
640 REM ** compara cartas **
650 GOSUB 930
660 PRINT"Yo tengo";myhand;
670 PRINT"y tu";yourhand
680 IF myhand<yourhand GOTO 730 ELSE GOTO 700
690 GOSUB 930:PRINT"/Te has pasado!"
700 PRINT"Gano"
710 mygames=mygames+1:GOTO 140
720 GOSUB 930:PRINT"/Me he pasado!"
730 PRINT"Ganas"
```

```
740 yourgames=yourgames+1:GOTO 140
750 END
760 REM ** extrae carta AL AZAR **
770 card=INT(RND*13)+1
780 suit=INT(RND*4)+1
790 value=card
800 IF value>10 THEN value=10
810 RETURN
820 REM ** expone carta sacada **
830 LOCATE x,y
840 PRINT CHR$(24);"    ";CHR$(24)
850 LOCATE x,y+1
860 PRINT CHR$(24);" ";
870 PRINT MID$(card$,card,1);
880 PRINT MID$(suit$,suit,1);
890 PRINT" ";CHR$(24)
900 LOCATE x,y+2
910 PRINT CHR$(24);"    ";CHR$(24)
920 RETURN
930 LOCATE 1,24:PRINT SPACE$(40)
940 LOCATE 1,24:RETURN
```

SIMON (fue un mago)

CHR$

Como puedes ver del listado siguiente, ¡no es tan simple como todo eso! De nuevo, aquí hay algunos comandos para los que tendremos que esperar a la Parte 2 de este curso. Hay sin embargo una palabra clave, que puede que te guste conocer. Es CHR$ (abreviatura de character), y es una **función (no** un **comando).**

Retrocediendo al Capítulo 3, vimos que podemos conseguir algunos caracteres especialmente curiosos en la pantalla que no aparecen rotulados en las teclas.

CHR$ nos permite que los usemos mediante su **código de carácter.** Por ejemplo, para poner un pequeño barco en la pantalla, el comando sería:

print chr$(239)

En la Guía de Usuario del Amstrad CPC464 está la lista completa de ellos.

```
10 cr$=CHR$(13)
20 REM Simon
30 REM **** INSTRUCCIONES ****
40 MODE 1:BORDER 20:INK 0,20:INK 1,1
50 LOCATE 15,2:PRINT CHR$(24);"Simon";CHR$(24)
60 PRINT
70 PRINT"En este juego, tienes que fijarte en"
80 PRINT"los circulos parpadeantes y recordar"
90 PRINT"las figuras. Cuando se termine la se-"
100 PRINT"cuencia debes hacer lo mismo con"
110 PRINT"las teclas del cursor. La secuencia"
```

```
120 PRINT"se  incrementa en uno despues de cada"
125 PRINT"acierto.":PRINT
130 PRINT"Por ejemplo,  un  circulo en la parte"
140 PRINT"superior  de la pantalla se indicaria"
150 PRINT"por la flecha hacia arriba.  Las  te-"
160 PRINT"clas del cursor  estan encima del te-"
170 PRINT"clado  numerico  y estan marcadas co-"
175 PRINT"mo sigue:"
180 PRINT TAB(17);CHR$(240)
190 PRINT TAB(16);CHR$(242);" ";CHR$(243)
200 PRINT TAB(17);CHR$(241)
210 LOCATE 7,22:PRINT"Pulsa ENTER para continuar"
220 LOCATE 8,24:PRINT "/habra antes una pausa!"
230 WHILE INKEY$<>cr$:WEND
240 REM **** PREPARACION ****
250 MODE 0
260 WINDOW 7,14,10,16
270 b=17:f=3:REM F O N D O / F R O N T E
280 BORDER b
290 INK 0,17
300 FOR i=1 TO 15:INK i,b:NEXT
310 x=320:y=70:c=2:GOSUB 940
320 y=330:c=1:GOSUB 940
330 x=120:y=200:c=3:GOSUB 940
340 x=520:c=4:GOSUB 940
350 INK 5,f:PEN 5
360 RANDOMIZE TIME
```

```
370 a$=""
380 REM **** EXhibe SECUENCIA ****
390 a$=a$+CHR$(RND*3+1)
400 FOR i=1 TO LEN(a$)
410 FOR j=1 TO 200:NEXT
420 x=ASC(MID$(a$,i,1))
430 INK x,2*x+1
440 SOUND 1,10+x*100
450 FOR j=1 TO 200:NEXT
460 INK x,b
470 NEXT
480 FOR i=1 TO 100:NEXT
490 REM **** reCOGE respuesta ****
500 FOR i=1 TO LEN(a$)
510 WHILE k$>"":k$=INKEY$:WEND
520 FOR L=1 TO 2000:k$=INKEY$
530 IF k$>"" THEN 560
540 NEXT
550 k$=" "
560 k=ASC(k$)-239:IF k<1 OR k>4 THEN 520
570 x=ASC(MID$(a$,i,1))
580 IF k<>x GOTO 730:REM @TRATAR fallo
590 INK x,2*x+1
600 SOUND 1,10+x*100
610 FOR j=1 TO 80:NEXT
620 INK x,b
630 FOR j=1 TO 20:NEXT
```

```
640 NEXT
650 REM **** L O G R A D O ****
660 CLS:PRINT" /BIEN!"
670 PRINT:PRINT:PRINT" PUNTOS:"
680 PRINT:PRINT"  ";LEN(A$)
690 FOR j=1 TO 600:NEXT
700 LOCATE 1,1:PRINT"       "
710 GOTO 390
720 REM **** F A L L A D O ****
730 SOUND 1,2000
740 CLS:PRINT"  Mal"
750 FOR j=1 TO 300:NEXT
760 PRINT:PRINT"   La":PRINT"secuen- cia era:"
770 FOR i=1 TO LEN(a$)
780 x=ASC(MID$(a$,i,1))
790 INK x,2*x+1
800 SOUND 1,10+x*100
810 FOR j=1 TO 200:NEXT
820 INK x,b
830 FOR j=1 TO 200:NEXT
840 NEXT
850 REM **** R E I N I C I O ****
860 CLS
870 PRINT"  Has"
880 PRINT"acertado"
890 PRINT"  ";LEN(a$)-1
900 PRINT:PRINT" PULSA"
```

```
910 PRINT" ENTER"
920 WHILE INKEY$<>CR$:WEND
930 GOTO 360
940 REM 'Encargado_de_curvas'
950 r=60
960 FOR i=-r TO r STEP 2
970 h=SQR(r*r-i*i)
980 MOVE x-h,i+y:DRAW x+h,i+y,c
990 NEXT
1000 RETURN
```

Autoevaluación

Antes de que pases a la Parte 2 de este curso, **Más sobre BASIC,** practica todo lo que sea posible en escribir tus programas, y pasa a través de la última de nuestras comprobaciones TAE12. Te presentará algunas cuestiones sobre **todos** los capítulos de esta parte del curso y te mostrará si necesitas volver a repasar alguno de los temas.

¡Buena suerte!

LISTA DE CLAVES

A continuación, hay una lista, capítulo por capítulo, de todas las claves del BASIC Amstrad tratadas en este libro. No todas las variaciones o ampliaciones han sido comentadas, dado que después de todo, este es un libro para principiantes. La Parte 2 de este curso, **Más sobre BASIC**, cubre claves adicionales y técnicas de programación más avanzadas.

Una descripción de todas las palabras clave, se puede encontrar en la Guía de Usuario Amstrad CPC464.

Capítulo 2

RUN
LOAD

Capítulo 3

CLS
RUN

Capítulo 4

BORDER
MODE
CAT

Capítulo 5

CLG
INK
DRAW
LIST
MOVE
NEW
PAPER
PLOT
REM

Capítulo 6

INPUT
LET
LOCATE
PRINT
SAVE

Capítulo 7

CONT
EDIT
ELSE
GOTO
IF
STOP
THEN

Capítulo 8

DRAWR
END
FOR
GOSUB
MOVER
NEXT
PLOTR
RETURN
STEP

Capítulo 10

ENT
ENV
SOUND

Capítulo 11

ROUND

Capítulo 12

CHR$
INKEY$
RND
TIME

LISTA DE PROGRAMAS

La cassette A contiene los siguientes programas en el mismo orden en que se mencionan en este libro. La cassette B contiene las pruebas de autoevaluación (TAEs) que el lector deberá completar al final de cada capítulo, exceptuando los capítulos 1 y 9.

Capítulo 2

HOLA
SIMON (Ver también Capítulo 12)

Capítulo 3

LETRAS
REPITE
TECLADO
AHORCADO

Capítulo 4

DIBUJAR
COORGEOM
BOMBARDERO

Capítulo 5

CASA

Capítulo 6

BLOQUES
GENERADOR DE FRASES

Capítulo 7

DECO

Capítulo 8

MANSION

Capítulo 10

SONIDOS
ORGANO

Capítulo 11

TABMULT
HUERTO

Capítulo 12

BLACKJACK
SIMON

INDICE


AMSTRAD