

Programación en Q BASIC

TEMA 1: INTRODUCCIÓN

1. Introducción

Qbasic es un entorno de programación constituido por un *editor* que permite convertir el ordenador en una máquina de escribir sofisticada para construir programas fuente, *un gestor de archivos* (en el caso del PC, el propio sistema operativo), *un compilador de lenguaje* y *un depurador para corrección de errores*.

El editor interactivo es la pieza central de Qbasic. Es un editor que verifica la sintaxis de cada línea tan pronto se teclea. Si la sintaxis es correcta, se traduce la línea inmediatamente al código ejecutable; en caso contrario, aparece una descripción del error.

Como cada línea se traduce a código ejecutable (.BAS) en el momento de introducción, se puede inmediatamente corregir y capturar la mayoría de los errores. No se precisa esperar a la compilación (.EXE) después de que se haya terminado la edición el programa está preparado para su ejecución.

Qbasic es un lenguaje de alto nivel que incorpora un depurador interactivo. Se puede detener un programa en cualquier punto, editar el programa y reanudar la ejecución en el punto en que se detuvo. En Qbasic se puede crear de modo fácil y rápido versiones de programas (.EXE) que se ejecutan directamente desde el DOS.

2.- ¿ DÓNDE ENCONTRAR QBASIC ?

Si tienes un CD con Windows, desde Mi PC abres el disco y en las carpetas other → tools → oldmsdos encontrarás el programa Qbasic.

Copia la carpeta oldmsdos en tu disco duro, por ejemplo en Mis Documentos. Una vez copiada ábrela y arrastra el archivo Qbasic al escritorio para crear un acceso directo.

El programa Qbasic también lo tienen casi todos los ordenadores que tienen instalado el sistema operativo Windows. Para encontrarlo solamente tienes que seguir los siguientes pasos: pon en inicio → buscar → archivos o carpetas → qbasic.exe .

3.- COMANDOS DE QBASIC

Los Comandos se encuentran en la "Barra de Menú" de la pantalla de QBasic.

Un programa se compone de grupos de comandos (Instrucciones) que les damos al computador, los comandos en QBASIC se escriben en líneas (líneas de programa) que pueden ir precedidas o no de un número que indica su ubicación (número de línea).

Ejemplo **sin número de ubicación** de las líneas de programa:

```
SCREEN 12
CLS
INPUT "¿Cuántos años tienes?:\0 \0 ", edad
IF edad < 18 THEN
PRINT "Eres menor de edad"
ELSE
PRINT "Eres mayor de edad"
END IF
```

En los programas muy largos y laboriosos se facilita su desarrollo y seguimiento si numeramos las líneas de programa.

Ejemplo **con número de ubicación** de las líneas de programa:

```
10 SCREEN 12
20 CLS
30 INPUT "¿Cuántos años tienes?:\0 \0 ", edad
40 IF edad < 18 THEN
50 PRINT "Eres menor de edad"
55 ELSE
60 PRINT "Eres mayor de edad"
70 END IF
```

Es conveniente numerar las líneas de 10 en 10. De este modo, si durante la elaboración del programa se nos olvida introducir una línea, la podremos introducir posteriormente con su correspondiente número. Observa en el ejemplo que del número de línea 50 se pasa al 55.

Una sola línea de programa puede albergar varias instrucciones siempre que éstas estén separadas por dos puntos (:) , aunque desaconsejo que los principiantes utilicen esta forma de estructurar un programa:

```
SCREEN 12 : CLS : INPUT "¿Cuántos años tienes?:\0 \0 ", edad
IF edad < 18 THEN : PRINT "Eres menor de edad" : ELSE
PRINT "Eres mayor de edad" : END IF
```

Una vez realizado el programa éste se ejecuta pulsando la tecla F5

Las Teclas : ESC , ALT + Q y ctrl. + "BREAK" detienen la ejecución del programa.

Si estando en la pantalla de Qbasic deseas obtener información adicional sobre algún comando, sitúa el cursor de Qbasic sobre dicho comando y pulsa el botón derecho del ratón o F1, se activará el menú de ayuda conteniendo información sobre esa sentencia en particular.

TEMA 2: INSTRUCCIONES GRÁFICAS Y OTRAS

1.- RESOLUCIÓN DE LA PANTALLA

QBasic puede trabajar con diferentes modos de pantalla. Cuando digo modos de pantalla estoy hablando de diferentes tamaños y colores. Aquí tienes una pequeña lista de diferentes modos en Qbasic:

Los siguientes modos de pantalla solamente se utilizan con ordenadores muy antiguos:

- SCREEN 0: Sólo modo de texto → Tinta en color solamente la línea de texto
- SCREEN 1: 320 * 200 gráficos → Cambia los colores
- SCREEN 2: 640 * 200 gráficos → No admite color de fondo
- SCREEN 4: 640 * 480 gráficos → No admite color de fondo

Con los ordenadores actuales se pueden utilizar los siguientes modos de pantalla:

- SCREEN 7: 320 * 200 gráficos → La podemos utilizar. Tamaño de letra grande
- SCREEN 8: 640 * 200 gráficos → La podemos utilizar. Tamaño de letra mediano
- SCREEN 9: 640 * 350 gráficos → La podemos utilizar. Tamaño de letra pequeño
- SCREEN 10: 640 * 350 gráficos, sólo monitor monocromo → Blanco y negro
- SCREEN 11: 640 * 480 gráficos → La podemos utilizar.
- SCREEN 12: 640 * 480 gráficos → La podemos utilizar. Tamaño de letra pequeña
- SCREEN 13: 320 * 200 gráficos → La podemos utilizar. Tamaño de letra grande

Los modos de pantalla son muy importantes en los programas. Por ejemplo, si tu quieres dibujar líneas, cubos y círculos en la pantalla con alguna clase de método gráfico debes usar un modo de pantalla que te admita el dibujo de gráficos.

Para más información de los modos de pantalla escribe "SCREEN" en la pantalla de Qbasic, pon el cursor sobre esta palabra y presiona el botón derecho del ratón o **F1**.

Modos de pantalla más utilizados:

- ♦ **SCREEN 12:** Tiene una resolución de 640 x 480 píxeles. Admite 16 colores elegidos de una paleta de 256.
- ♦ **SCREEN 13:** Tiene una resolución de 320 x 200 píxeles. Admite 256 colores.

Ubicación: la instrucción SCREEN 12 se coloca generalmente en la cabecera del programa

Cualquier píxel se puede ubicar en la pantalla mediante un sistema de coordenadas cartesianas que tiene como (0, 0) la esquina superior izquierda de la pantalla

Para una resolución SCREEN 12 : $X < 640$ e $Y < 480$,

2.- COLORES

- | | | |
|-------------|------------------|-----------------------|
| 0 → Negro | 6 → Amarillo | 12 → Rojo claro |
| 1 → Azul | 7 → Blanco mate | 13 → Magenta claro |
| 2 → Verde | 8 → Gris | 14 → Amarillo claro |
| 3 → Cian | 9 → Azul claro | 15 → Blanco brillante |
| 4 → Rojo | 10 → Verde claro | |
| 5 → Magenta | 11 → Cian claro | |

3.- PUNTO

Sintaxis: **PSET (X, Y), número del color**

- ♦ (X, Y) → coordenadas del punto en la pantalla
- ♦ $X < 640$ e $Y < 480$, para una resolución SCREEN 12
- ♦ Color: puede tomar valores de 0 a 15. Por omisión dibuja un píxel blanco.

Ejemplo: PSET (20, 30), 14

4.- LÍNEA

Sintaxis: **LINE (X1, Y1) – (X2, Y2), número del color**

- ♦ (X1, Y1) → punto de comienzo de la línea.
- ♦ (X2, Y2) → fin de la línea.
- ♦ Color: puede tomar valores de 0 a 15. Por omisión dibuja una línea blanca.

Ejemplo: LINE (5, 8) - (105, 123), 14

5.- CUADRADO - RECTÁNGULO

Sintaxis: **LINE (X1, Y1) – (X2, Y2), número del color, B**

- ♦ B → indica cuadrado o rectángulo
- ♦ (X1, Y1) – (X2, Y2) → punto de comienzo y fin de la diagonal del cuadrado o rectángulo

Ejemplo: LINE (5, 5) - (105, 105), 14, B

A partir de la diagonal especificada el programa dibuja un cuadrado de 100 x 100 con línea de color amarillo claro.

6.- CUADRADO - RECTÁNGULO RELLENO DE COLOR

Sintaxis: **LINE (X1, Y1) – (X2, Y2), número del color, BF**

- ♦ BF → indica cuadrado o rectángulo relleno
- ♦ (X1, Y1) – (X2, Y2) → punto de comienzo y fin de la diagonal del cuadrado o rectángulo
- ♦ Ejemplo: LINE (5, 5) - (105, 105), 14, BF

A partir de la diagonal especificada el programa dibuja un cuadrado de 100 x 100 relleno de color amarillo claro.

7.- CÍRCULO

Sintaxis: **CIRCLE (X, Y), valor del radio, número del color del perímetro**

- ♦ (X, Y) → coordenadas del centro del círculo en la pantalla
- ♦ Color del perímetro: puede tomar valores de 0 a 15. Por omisión dibuja una línea blanca.

Ejemplo: CIRCLE (100, 100), 30, 2

8.- CAMBIO DEL COLOR DE FONDO DE LA PANTALLA

Sintaxis: **PAINT (X, Y), color de relleno**

- ♦ (X, Y) → coordenadas de un punto interior de la pantalla, por ejemplo el (0, 0).
- ♦ color de relleno → puede tomar valores de 0 a 15.

9.- COLOREADO DE UNA ZONA

Sintaxis: **PAINT (X, Y), color de relleno, color del límite o borde**

- ♦ (X, Y) → coordenadas de un punto interior de la figura que queremos rellenar
- ♦ color de relleno → puede tomar valores de 0 a 15.
- ♦ Color del límite → debemos poner el color que posee el perímetro de la figura que queremos rellenar. OJO, si no se hace así se colorea toda la pantalla.

El comando PAINT pinta un color hasta encontrarse con un borde de un color específico. Esto significa que si no se aplica sobre algo cerrado pinta toda la pantalla. Esto ocurre también si encuentra un borde que nos el color que debería ser

Ejemplo: CIRCLE (100, 100), 30, 2
PAINT (95, 93), 4, 2

El programa dibuja, en las coordenadas 100 x 100, un círculo de radio 30 con línea de perímetro de color verde y relleno de color rojo.

Las coordenadas (95, 93) marcan un punto interior del círculo. Podríamos haber puesto cualquier otro punto interior, el más cómodo es el del centro del círculo (100, 100)

10.- MULTICOMANDO DE DIBUJO

Sintaxis: **DRAW “{parámetros}”**

Es un multicomando que tiene muchas funciones dentro de él. Con este comando podemos realizar un dibujo conforme a los parámetros que le vayamos dando.

Parámetros:

Cx → Color x, indica que se va a empezar a dibujar con el color x.

Dx → Down x, dibuja x píxeles hacia abajo

Ux → Up x, dibuja x píxeles hacia arriba

Lx → Left x, dibuja x píxeles hacia la izquierda

Rx → Right x, dibuja x píxeles hacia la derecha

Ex → dibuja x píxeles hacia arriba-derecha

Fx → dibuja x píxeles hacia abajo-derecha

Gx → dibuja x píxeles hacia abajo-izquierda

Hx → dibuja x píxeles hacia arriba-izquierda

M x,y → Move x,y , se mueve al punto x, y dibujando píxeles.

Hace una línea desde donde esté al punto x, y

BM x,y → se mueve al punto x, y sin dibujar.

Ax → Gira una figura hecha con DRAW:

- ♦ x = 1 → gira 90°
- ♦ x = 2 → gira 180°
- ♦ x = 3 → gira 270°

Ejemplo:

```
SCREEN 12
DRAW " C10 BM 150,140 D20 L30 U40 R60"
```

Comentarios a la línea anterior:

C10 → dibuja con color verde.
 BM 150,140 → desplázate a la posición 150, 140 de la pantalla
 D20 → dibuja una línea de 20 píxeles hacia abajo
 L30 → dibuja 30 píxeles hacia la izquierda.
 U40 → dibuja 40 píxeles hacia arriba.
 R60 → Dibuja 60 píxeles hacia la derecha.

11.- INSTRUCCIONES VARIAS**1) PAUSA**

Sintaxis: **SLEEP t**

Para la ejecución del programa durante un número de segundos igual al indicado con el número " t ".

Ejemplo: SLEEP 10 → La ejecución del programa se detiene durante 10 segundos.

2) BORRADO DE PANTALLA

Sintaxis: **CLS** → Se coloca en la cabecera del programa para borrar la pantalla realizada en un programa anterior

3) INTRODUCCIÓN DE INFORMACIÓN SOBRE EL PROGRAMA

Sintaxis: **REM: " Información que deseemos introducir "**

4) SONIDO

Sintaxis: **BEEP**

5) DETECCIÓN DE PULSACIÓN SOBRE EL TECLADO

Sintaxis: **a\$ = INKEY \$** → Detecta si se pulsa una tecla y la almacena en la variable a\$

6) CLEAR

Sintaxis: **CLEAR**

Hace que todas las variables de un programa queden a cero. Este comando es opcional y solamente se pone por seguridad.

7) IMPRESIÓN EN PANTALLA

Sintaxis: **PRINT " texto que se desea imprimir "**

Ejemplo: CLS
 PRINT " Hola, Kyle Roberts, ¿cómo estás? "
 END

EJERCICIOS PROPUESTOS:

Ejercicio 2.1. Dibuja un punto blanco brillante sobre fondo negro en las coordenadas 420 x 200

Ejercicio 2.2. Dibuja una línea de color amarillo que parta de las coordenadas (23, 45) y finaliza en las coordenadas (395, 280)

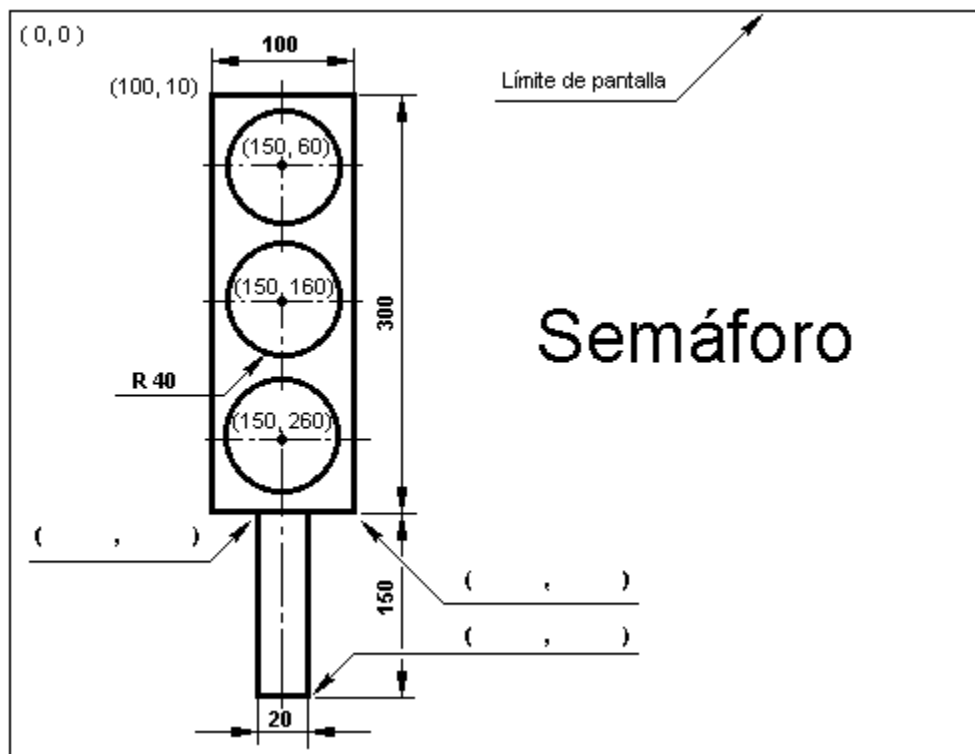
Ejercicio 2.3. Dibuja un rectángulo de 200 x 100 con línea de color verde de forma que su esquina superior izquierda esté en las coordenadas (225, 115).

Ejercicio 2.4 . Modifica el programa del ejercicio anterior para que el rectángulo esté relleno de color rojo.

Ejercicio 2.5. En las coordenadas (320, 240) dibuja un círculo de radio 100, con perímetro de color gris, color de relleno azul y fondo de pantalla blanco.

Ejercicio 2.6. Aplicando los conocimientos adquiridos realiza un programa que dibuje el semáforo de la figura de forma que cumpla las siguientes condiciones:

- ♦ Pantalla color blanco
- ♦ Perímetros de figuras color gris
- ♦ Todas las lámparas apagadas: círculos rellenos de color negro
- ♦ Relleno rectángulo grande color azul claro
- ♦ Relleno rectángulo pequeño color cian



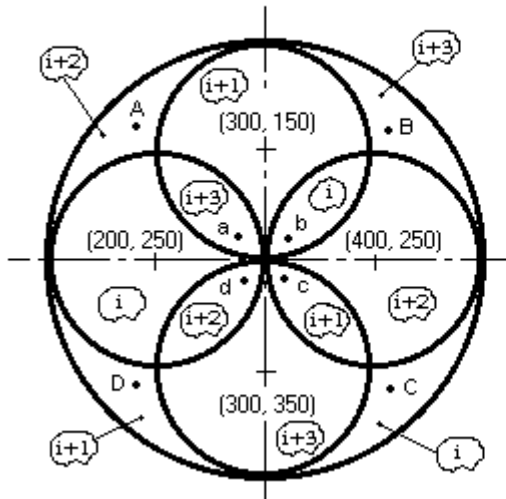
Ejercicio 2.7. Modifica el programa del ejercicio 6 para que cumpla con las siguientes condiciones:

- Tras aparecer en pantalla del ordenador el semáforo con todas las lámparas apagadas, a los dos segundos se debe encender la lámpara verde.
- La lámpara verde ha de permanecer encendida 4 segundos.
- A los dos segundos de encenderse la lámpara verde se ha de encender la lámpara amarilla, que se apagará junto con la lámpara verde.
- Simultáneamente al apagado de las lámparas verdes y amarilla se ha de encender la lámpara roja que permanecerá 4 segundos encendida y se apagará.

Ejercicio 2.8. Realiza un programa que coloree las distintas superficies de las que consta la figura “Margarita” y que cumpla con las siguientes condiciones:

- Aplica una gama de colores del 1 al 14.
- Dos superficies contiguas no pueden tener el mismo color.

MARGARITA



PUNTO	COORDENADAS	COLOR
a	(290, 240)	i + 3
b	(310, 240)	i
c	(310, 260)	i + 1
d	(290, 260)	i + 2
A	(175, 125)	i + 2
B	(425, 125)	i + 3
C	(425, 375)	i
D	(175, 375)	i + 1
Central	(300, 250)	—

TEMA 3: BUCLES

Un bucle es un rizo consistente en dar una vuelta sobre sí mismo. En programación es algo parecido; se trata de la *repetición sucesiva de varias sentencias*.

1.- BUCLES CON CONTADOR: CICLOS FOR / NEXT

1.1.- BUCLES CRECIENTES

Sintaxis:

FOR i = inicio TO fin

Instrucciones
que queremos
repetir

NEXT i

También podemos poner:

FOR i % = inicio TO fin

Instrucciones
que queremos
repetir

NEXT i

Ejemplo: Imprimir en la pantalla números del 1 al 20 con una cadencia de 1 segundo.

```
CLS
FOR I = 1 TO 20
PRINT I
SLEEP 1
NEXT I
END
```

Con las instrucciones FOR / NEXT evitamos tener que repetir 20 veces todas las instrucciones de impresión.

Al comenzar el programa la variable `I` tomará valor 1. Cada vez que la instrucción `NEXT I` es alcanzada el programa comprobará si `I` es 20. Siempre que `I` no sea 20, el programa le sumará uno al valor que tenga `I` y regresará a la línea donde está la instrucción `FOR I = 1 TO 20`. Cuando `I` alcance el valor 20 el programa continuará ejecutando la línea siguiente a `NEXT I`.

`I` es una variable que al ejecutarse el bucle va incrementando su valor de 1 en 1. Si a continuación de `FOR I = 1 TO 20` ponemos `STEP 2` su valor se incrementará de 2 en 2. Si es `STEP 3` su valor se incrementará de 3 en 3, y así sucesivamente.

Ejemplo: Imprimir en la pantalla números impares del 1 al 20 con una cadencia de 1 segundo.

```
CLS
FOR I = 1 TO 20 STEP 2
PRINT I
SLEEP 1
NEXT I
END
```

Ejemplo: Imprimir en la pantalla números pares del 1 al 20 con una cadencia de 1 segundo.

```
CLS
FOR I = 2 TO 20 STEP 2
PRINT I
SLEEP 1
NEXT I
END
```

Ejemplo: Realiza un programa que dibuje 150 circunferencias concéntricas de varios colores

```
REM "Circunferencias concéntricas"
SCREEN 12 ; CLS
FOR I = 1 TO 150 STEP 1
C = I MOD 3
REM: con la línea anterior hacemos que C tome alternativamente los valores 0, 1, 2
CIRCLE (320, 240), I, C
REM: al poner de radio la variable I, cada vez que se ejecute el bucle el radio irá aumentando
SLEEP 1
NEXT I
END
```

1.2.- BUCLES DECRECIENTES

Sintaxis:

FOR I = fin TO inicio STEP -1

Instrucciones que queremos repetir

NEXT I

Ejemplo: Cuenta atrás

```
REM: Cuenta atrás
CLS
FOR I = 20 TO 0 STEP -1
LOCATE 10,10 ; REM: indica fila y columna de comienzo de impresión
PRINT I
SLEEP 1
NEXT I
END
```

2.- BUCLES CONDICIONALES

2.1.- Comandos DO, LOOP, UNTIL {condición}

Sintaxis:

DO

Instrucciones que queremos repetir

LOOP UNTIL "Condición"

También podemos poner:

DO UNTIL "Condición"

Instrucciones que queremos repetir

LOOP

Con este bucle las instrucciones se ejecutan continuamente hasta que se cumpla una determinada condición.

Ejemplo:

```
DO
PRINT "Hola, Ramón Menor"
a$ = INKEY$
LOOP UNTIL a$ <> ""
```

El bucle se estará ejecutando hasta que a\$ sea distinto de nada (a\$ <> ""), es decir, hasta que se pulse una tecla.

Ejemplo:

```
DO
PRINT "Hola, Laura Delima"
a$ = INKEY$
LOOP UNTIL a$ = "A "
```

El bucle se ejecutará hasta que pulsemos la tecla A

2.2.- Comandos WHILE {condición}, WEND

El programa ejecutará repetidamente las instrucciones contenidas entre WHILE y WEND, mientras se cumpla la condición especificada en WHILE. La condición ha de ser una expresión numérica que Qbasic evaluará como verdadera o falsa . Si la condición es verdadera el programa continuará ejecutándose y si es falsa se detendrá.

WHILE {condición}

Instrucciones que queremos repetir

WEND

Ejemplo:

```
CLS
WHILE i < 10
i = i + 1
PRINT "¿ Qué tal , Marina Corchado? "
SLEEP 1
WEND
END
```

La variable i inicialmente toma el valor 0. Como en la línea WHILE i < 10 se cumple la condición, se ejecutarán las instrucciones siguientes. Al leer la instrucción i = i + 1 la variable i se incrementará en una unidad. Al llegar a WEND el programa regresará de nuevo a WHILE y volverá a comprobar de nuevo la condición y así sucesivamente hasta que no se cumpla la condición i < 10 .

2.3.- Comandos DO, LOOP, WILE {condición}

El comando WHILE se puede utilizar tanto con el comando DO (DO WHILE) como con el comando LOOP (LOOP WHILE)

La sintaxis puede ser de 2 formas:

1ª Forma:

DO WHILE {condición}

Instrucciones
que queremos
repetir

LOOP

2ª Forma:

DO

Instrucciones
que queremos
repetir

LOOP WHILE {condición }

Con este bucle las instrucciones se ejecutan repetidamente mientras que se cumpla una determinada condición. El comando WHILE es el que permite comprobar dicha condición.

Ejemplo 1:

```
CLS
DO WHILE i < 10
i = i + 1
PRINT "¿Cómo estás, Agustín?"
SLEEP 1
LOOP
END
```

En este ejemplo el programa comprobará al comienzo del bucle si i ha alcanzado el valor diez:

- ♦ Si no lo ha alcanzado se repetirá el bucle.
- ♦ Si lo ha alcanzado continuará la ejecución desde el comando LOOP.

Ejemplo 2: el siguiente programa tiene el mismo efecto que el del ejemplo anterior.

```
CLS
DO
I = I + 1
PRINT "Lara, porfa, cállate un poquito"
SLEEP 1
LOOP WHILE i < 10
END
```

EJERCICIOS PROPUESTOS:

Ejercicio 3.1. Utilizando los comandos FOR, TO y NEXT realiza un programa que presente en la pantalla los numero del 1 al 15 en diferentes colores.

Ejercicio 3.2. Utilizando los comandos FOR, TO y NEXT realiza un programa que presente en la pantalla para los diferentes colores “ Este es el color N° ___”.

Ejercicio 3.3. Utilizando los comandos WHILE y WEND , realiza un programa que presente los numero del 1 al 10, en diferentes colores, en la fila 15 y columna 40 de la pantalla.

Ejercicio 3.4. Utilizando los comandos DO, LOOP y UNTIL , realiza un programa que presente los numero del 1 al 10, en diferentes colores, en la fila 15 y columna 40 de la pantalla.

Ejercicio 3.5. Utilizando los comandos DO, LOOP y WHILE , realiza un programa que presente los numero del 1 al 10, en diferentes colores, en la fila 15 y columna 40 de la pantalla.

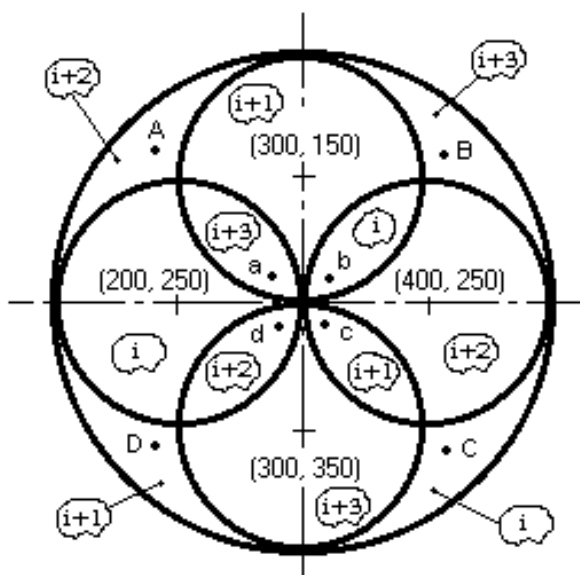
Ejercicio 3.6. Modifica el programa del **Ejercicio 2.7** para que ciclo de encendido y apagado del semáforo se repita 10 veces.

Ejercicio 3.7. Modifica el programa del **Ejercicio 2.7** de forma que la secuencia de encendido y apagado del semáforo se repita continuamente hasta que pulsemos una tecla.

Ejercicio 3.8. Realiza un programa que coloree las distintas superficies de las que consta la figura “ Margarita” (tema 2) y que cumpla las siguientes condiciones:

- Aplica una gama de colores del 1 al 14.
- Dos superficies contiguas no pueden tener el mismo color.
- Los colores de las distintas superficies deben cambiar con pausas de 1 segundo.
- Utiliza para su realización ciclos FOR / NEXT y los comandos DO / LOOP UNTIL
- El programa debe funcionar de forma continuada hasta que se pulse una tecla del teclado.

MARGARITA



PUNTO	COORDENADAS	COLOR
a	(290, 240)	i + 3
b	(310, 240)	i
c	(310, 260)	i + 1
d	(290, 260)	i + 2
A	(175, 125)	i + 2
B	(425, 125)	i + 3
C	(425, 375)	i
D	(175, 375)	i + 1
Central	(300, 250)	—

TEMA 4: IMPRESIÓN DE TEXTOS EN PANTALLA

1.- BORRADO DE PANTALLA

Sintaxis: **CLS**

El comando CLS permite limpiar la pantalla de los restos de programas anteriores.

2.- IMPRESIÓN EN PANTALLA

Sintaxis: **PRINT “ texto que se desea imprimir ”**

Ejemplo:

```
CLS
PRINT “ Hola, Jesús Ares, ¿cómo estás? ”
END
```

Ejemplo: escribe en la pantalla diez veces “Hola, ¿cómo estás Albertina?”

```
CLS
FOR I = 1 TO 10
PRINT “ Hola, ¿cómo estás Albertina? “
NEXT I
END
```

3.- FILA Y COLUMNA DEL COMIENZO DE IMPRESIÓN

Se utiliza para situar el texto en un determinado lugar de la pantalla

Sintaxis: **LOCATE fila, columna, cursor**

- Cursor: 1 → encendido ; 0 → apagado
- Valores máximos aproximados de filas y columnas:

	FILA	COLUMNA
SCREEN 12	29	75
SCREEN 13	23	40

Ejemplo:

```
REM "Prueba de impresión de texto"
SCREEN 12 ; CLS
LOCATE 1, 1
PRINT “ Fíjate bien, Armando, estamos en 1, 1”
SLEEP 4
PRINT “ Mira , Sergio, ahora hemos pasado a 20, 20”
END
```

Ejemplo: Números escalonados

```
REM "Números escalonados"
CLS
SCREEN 12
FOR i = 1 TO 20 STEP 1
  SLEEP 1
  LOCATE i, i
  PRINT i
NEXT i
END
```

4.- IMPRESIÓN EN PANTALLA DE LA FECHA ACTUAL

Sintaxis: **PRINT DATES**

Con este comando podemos presentar en pantalla la fecha actual

Ejemplo:

```
PRINT " La fecha de hoy es " ; DATES
```

5.- IMPRESIÓN EN PANTALLA DE LA HORA ACTUAL

Sintaxis: **PRINT TIMES**

Con este comando podemos presentar en pantalla la hora actual

Ejemplo:

```
PRINT " La hora actual es " ; TIMES
```

6.- COLORES DE LOS CARACTERES Y FONDO DE PANTALLA

Sintaxis: **COLOR , número color letra, número color fondo**

- ♦ número color letra: toma valores de 0 a 15
- ♦ número color fondo: toma valores de 0 a 15. Por omisión toma como color de fondo el negro

Ejemplo:

Realiza un programa que imprima el texto " Laura Terrón, esto es un ejemplo para probar el color de los caracteres ", con los 15 colores de SCREEN 12 (menos el negro) sobre fondo negro. Utiliza para ello un ciclo FOR/NEXT.

```
REM: Prueba de color
CLS
FOR I = 1 TO 15
  COLOR I, 0
  PRINT "Laura Terrón, esto es un ejemplo para probar el color de los caracteres"
NEXT I
END
```

Ejemplo : Prueba de impresión de texto con diferentes colores

```
REM "Prueba de color y texto"  
CLS  
SCREEN 12  
COLOR 4  
PRINT "SCREEN 12. Estamos usando una letra de color rojo sobre fondo negro"  
PRINT  
PRINT "SCREEN 12. La resolución de pantalla es de 640x480"  
PRINT  
SLEEP 3  
SCREEN 9  
COLOR 2, 1  
PRINT "SCREEN 9. Estamos usando una letra de color verde sobre un fondo azul"  
PRINT  
PRINT "SCREEN 9. La resolución de pantalla es de 640x350"  
PRINT  
SLEEP 3  
SCREEN 13  
PRINT "SCREEN 13. No admite color, presenta una letra de color blanco sobre fondo negro"  
END
```

EJERCICIO PROPUESTO:

Realiza un programa que nos muestre la pantalla de inicio de una aplicación para realizar las cuatro operaciones aritméticas básicas.

PROGRAMA DE CÁLCULO

¿Qué operación deseas realizar?:

- 1.- Suma
- 2.- Resta
- 3.- Multiplicación
- 4.- División

TEMA 5: VARIABLES E INTRODUCCIÓN DE DATOS

1.- INTRODUCCIÓN

Con las variables se puede guardar información relevante con la que se podrá operar más tarde. Una variable es un nombre que damos a un área de memoria en el cual guardamos un dato. Cuando necesitemos sacar esta parte del dato o modificar su valor, para localizar esa área de memoria pondremos el nombre de la variable.

Ejemplo: en los ciclos FOR / NEXT la variable es “ I “, en principio toma el valor 1 y luego la vamos incrementando de 1 en 1, de 2 en 2, etc.

2.- NOMBRE DE LAS VARIABLES

El nombre de las variables es libre , pero siempre conviene hacerlo descriptivo. Por ejemplo en la variable TELEFONO es obvio que su contenido se ha de referir a un N° de teléfono. Si a los N° de teléfonos los llamamos, por ejemplo, COCINA el programa también funcionará, pero en programas largos se dificultará su localización.

3.- SINTAXIS DE LAS VARIABLES

Las variables pueden tener cualquier forma y tamaño respetando las siguientes reglas:

- 1) El primer carácter debe ser siempre una letra: a, b, c.....A, B, C,....., no importa si el nombre está en mayúsculas o minúsculas.
- 2) Después de la primera letra podemos poner letras , números o subrayados (_)
- 3) El carácter final:
Para variable numéricas → puede ser : %, ! , # , \$, o nada.
Para variables de cadena → ha de ser : \$
- 4) No puede ser una palabra reservada para un comando de Q Basic, por ejemplo PRINT.

Ejemplo:

```
REM: "Coste"
CLS
Coste = 825
PRINT Coste
SLEEP 1
Coste = 36
PRINT Coste
END
```

Ejemplo

```
REM "Calculo de PVP"
CLS
COSTE = 1000
PVP = COSTE * 1.16
PRINT PVP
END
```

El programa guarda dos valores diferentes para la variable “Coste”. En principio se mostrará el primer valor, 825, correrá el programa y “Coste” pasará a tomar el valor 36, mostrándose finalmente el segundo valor, 36.

4.- VARIABLES DE CADENA

Una cadena es una forma de guardar información en Q BASIC

Ejemplo:

```
CLS
nombre1$ = " Mª Dolores Villalobos "
PRINT nombre1$
nombreuno$ = "Daniel Ventura "
PRINT nombreuno$
END
```

Ojo, SIEMPRE el símbolo \$ debe ir al final del nombre de la variable de cadena y su contenido debe de ir entre comillas. El símbolo \$ da a entender que es una variable de cadena (de cadena solamente).

Las variables de cadena pueden tener caracteres numéricos, pero estos no tienen valor numérico.

Ejemplo: nombre1\$ = "3215" → 3215 no tiene valor numérico

Q Basic sólo permite nombres de variables numéricas no superiores a 40 caracteres. En el caso de las variables de cadena se pueden introducir hasta 32767 caracteres (la única vez que podremos excedernos de este valor es si estamos escribiendo en un editor de texto).

Ejemplo:

```
SCREEN 12 ; CLS
manzanas = 2
ciruelas = 5
naranjas = 3
LOCATE 2, 1
PRINT "En el almacén tenemos: " ; manzanas ; "Manzanas," ; ciruelas ; Ciruelas y ;
naranjas ; "Naranjas"
SLEEP 3 ; LOCATE 6, 8
PRINT " Ahora hemos vendido des ciruelas y una naranja "
Ciruelas = ciruelas - 2
naranjas = naranjas - 1
LOCATE 10, 1
PRINT "En el almacén tenemos: " ; manzanas ; "Manzanas," ; ciruelas ; Ciruelas y ;
naranjas ; "Naranjas"
END
```

5.- PARTE ENTERA DE UN NÚMERO

Sintaxis: **INT { Variable numérica }** ; Ejemplo: B = INT A

Con este comando obtenemos la parte entera de un número: Si A = 3,46 B tomará el valor 3

6.- PARTE DECIMAL DE UN NÚMERO

Sintaxis: **FRAC { Variable numérica }**

Ejemplo: B = FRAC A

Con este comando obtenemos la parte decimal de un N° : Si A = 3,46 B tomará el valor 0,46

7.- GENERACIÓN DE NÚMEROS ALEATORIOS

1) Comando RND

Cada vez que este comando es leído por el programa se genera un número **cuasialeatorio** diferente, con un valor comprendido entre cero y uno.

Ejemplo: genera 25 números aleatorios comprendidos entre cero y uno

```
REM: números aleatorios comprendidos entre cero y uno
SCREEN 12
CLS
FOR I = 1 TO 25
X = RND
PRINT X
NEXT I
END
```

Ejemplo: genera 25 números aleatorios comprendidos entre tres y siete

```
REM: números aleatorios comprendidos entre tres y siete
SCREEN 12
CLS
FOR I = 1 TO 25
X = 3 + 4 * RND
PRINT X
NEXT I
END
```

Ejemplo: genera 25 números aleatorios enteros comprendidos entre tres y siete

```
REM: números aleatorios enteros
SCREEN 12
CLS
FOR I = 1 TO 25
X = INT (3 + 5 * RND)
PRINT X
NEXT I
END
```

2) Comando RANDOMIZE

Los números obtenidos con la sentencia RND no son verdaderamente aleatorios, ya que son generados utilizando un procedimiento computacional fijo; sin embargo, estos números parece que fueran aleatorios y tienen las mismas propiedades estadísticas que los números realmente aleatorios.

Cada vez que es leído el comando RND se genera la misma secuencia de números. Si previamente introducimos el comando RANDOMIZE, le estamos suministrando al generador de números aleatorios un punto de comienzo diferente.

La instrucción RANDOMIZE debe preceder, por tanto, a la instrucción RND

Si no indicamos nada más, el programa parará su ejecución para pedirnos que introduzcamos un punto de comienzo comprendido entre -32.768 y 32.767 para poder continuar.

3) Comando RANDOMIZE TIMER

Con esta instrucción el programa ejecuta cada vez un punto de comienzo que también es aleatorio.

```
REM: "Ejercicio INT / RANDOMIZER TIMER"
CLS
FOR i = 1 TO 25
RANDOMIZER TIMER
x = INT (3 + 5 * RND)
PRINT x
NEXT i
END
```

8.- INTRODUCCIÓN DE DATOS MEDIANTE EL TECLADO

Sintaxis: **INPUT ; { nombre de la variable numérica o de cadena }**

El comando INPUT permite leer caracteres del teclado y de archivos. Los datos introducidos mediante el teclado se guardan en la memoria con un nombre determinado.

Ejemplo:

```
CLS
INPUT "Tu nombre es:"; nombre$
INPUT "Tu edad es es:"; edad
PRINT "Hola, Ø"; nombre$ ; "Øtienes Ø"; edad; "Øaños"
END
```

- ♦ nombre\$ → variable de cadena
- ♦ edad → variable numérica
- ♦ Ø → espacio en blanco

En la sentencia { INPUT "Tu nombre es:" ; nombre\$ } si ponemos (;) después de la impresión de "Tu nombre es" aparece el signo ? y además deja un espacio a la derecha de éste.

Si en vez de (;) ponemos (,) no aparecerá ? y no dejará el espacio a la derecha con lo que al introducir el nombre lo escribirá unido a la pregunta. Para solucionarlo podemos hacer lo siguiente:

```
INPUT "Tu nombre es: Ø", nombre$
```

En la sentencia { PRINT "Hola, Ø"; nombre\$; "Øtienes Ø"; edad; "Øaños"} si en vez de (;) ponemos (,) se separará más las partes de la que consta la sentencia.

Otra forma de utilizar el comando INPUT con las variables de cadena es la siguiente:

Sintaxis: **Variable\$ = INPUT\$ (cantidad de teclas a recibir)**

Ejemplo: nombre\$ = INPUT\$ (6)

El llegar a esta línea el programa se detiene a la espera de que introduzcamos la información. El (6) indica el número de teclas que debemos pulsar

Ejemplo de programa:

```
CLS
nombre$ = INPUT$ (6)
PRINT "Tu nombre es: Ø"; nombre$
END
```

Ejemplo de programa:

```
REM "Interrogatorio"
CLS
SCREEN 12
COLOR 1
INPUT "Cual es tu nombre "; nombre$
COLOR 2
INPUT "Qué, edad tienes "; edad
COLOR 6
INPUT "Cual es tu número de teléfono "; telefono
COLOR 7
PRINT "Hola, tu nombre es "; nombre$; " ,tienes "; edad; " años y tu número de
teléfono es "; telefono
END
```

EJERCICIOS PROPUESTOS

Ejercicio 5.1 : Realiza un programa que nos permita obtener de forma aleatoria los 3 signos de la quiniela de fútbol (1 - x - 2)

Ejercicio 5.2 : Realiza un programa que nos permita obtener de forma aleatoria los 6 números de la lotería primitiva (1 → 49) de forma aleatoria

TEMA 6: TOMA DE DECISIONES

1.- NOTACIONES

- = → es igual a
- <> → no es igual a
- > → es mayor que
- < → es menor que
- >= → es mayor o igual que
- <= → es menor o igual que

2.- COMANDOS PARA LA TOMA DE DECISIONES

Comandos: IF.....THEN.....ELSE.....END IF

- ♦ IF → si, se cumple una condición
- ♦ THEN → entonces
- ♦ ELSE → en caso contrario (todo lo demás). Si no se cumple la sentencia continua la ejecución del programa
- ♦ END IF → fin de la toma de decisión

Sintaxis: **IF { condición } THEN { ejecuta la próxima sentencia } END IF**

Ejemplo:

Realiza un programa que establezca si una persona es mayor o menor de edad, en función del dato numérico introducido cuando se nos pregunte por nuestra edad.

Primera forma:

```
CLS
REM "Mayoría de edad"
INPUT "¿Cuántos años tienes?:", edad
IF edad < 18 THEN
PRINT "Eres menor de edad"
ELSE
PRINT "Eres mayor de edad"
END IF
```

Segunda forma:

```
CLS
INPUT "¿Cuántos años tienes?:", edad
IF edad < 18 THEN
PRINT "Eres menor de edad"
IF edad > 18 THEN
PRINT "Eres mayor de edad"
END IF
```

Ejemplo:

Realiza un programa que nos pregunte “ Qué día de la semana es” y , en función del día introducido, nos diga cuántos días faltan para terminar la semana.

```
CLS
REM "Días de la semana"
INPUT "¿ Qué, día de la semana es "; dia$
IF dia$ = "lunes" THEN
PRINT "Faltan 6 días para que acabe la semana"
END IF
IF dia$ = "martes" THEN
PRINT "Faltan 5 días para que acabe la semana"
END IF
IF dia$ = "miercoles" THEN
PRINT "Faltan 4 días para que acabe la semana"
END IF
IF dia$ = "jueves" THEN
PRINT "Faltan 3 días para que acabe la semana"
END IF
IF dia$ = "viernes" THEN
PRINT "Faltan 2 días para que acabe la semana"
END IF
IF dia$ = "sabado" THEN
PRINT "Falta 1 día para que acabe la semana"
END IF
IF dia$ = "domingo" THEN
PRINT "Acabó la semana"
END IF
```

EJERCICIOS PROPUESTOS

Ejercicio 6.1: Realiza un programa que te pregunte tu nombre, domicilio, numero de teléfono y edad y que posteriormente lo imprima todo junto.

Ejercicio 6.2: Realiza las primeras líneas de un programa que te pida una contraseña o PIN para ser ejecutado (por ejemplo: Juan, José, 1485, etc.) y que en caso de ser incorrecta te diga “acceso denegado”.

Ejercicio 6.3: Utilizando los comandos Do / LOOP UNTIL, RANDOMIZE TIMER, RND e IF / THEN realiza el programa de un juego de adivinanzas que te pida un número entero del uno al diez y que cumpla las siguientes condiciones:

- ♦ Si el número introducido es mayor que el generado por el ordenador, nos debe responder “ Es alto”
- ♦ Si el número introducido es menor, nos debe responder “ Es bajo”
- ♦ Cuando lo aciertes responderá “ Has acertado ”

Ejercicio 6.4: Realiza modificaciones al ejercicio anterior para introducir un contador y que cuando aciertes te responda “ Enhorabuena, has acertado al “x” intento”

TEMA 7: SALTOS

I.- SALTOS INCONDICIONALES

1.- COMANDO “GOTO”

Salta incondicionalmente al destino de salto especificado.

Sintaxis: **GOTO { Etiqueta de destino o N° de línea de destino }**

Ejemplo: GOTO 40

Según versiones de Q BASIC la etiqueta puede ser un número o una cadena de caracteres alfanuméricos

Ejemplo:

```
5 REM: Ejemplo de GOTO
10 SCREEN 12 ; CLS
20 PRINT " La línea 40 no se va a ejecutar"
30 GOTO 50
40 PRINT " Esta línea no se ejecuta"
50 PRINT ; PRINT "Hasta aquí el programa ejecutó las líneas 10, 20, 30 y 50"
60 END
```

2.- SUBROUTINAS

Una subrutina es una parte del programa que se ha de repetir en muchas ocasiones.

Para ahorrar trabajo las subrutinas se colocan después del fin (END) del programa principal y se accede a ellos mediante un comando GOSUB.

Las subrutinas:

- ♦ Necesitan de una etiqueta (nombre) que las identifique (ejemplo: “ PALOMA”, “450”, etc.)
- ♦ Finalizan siempre con el comando RETURN

3.- GOSUB

El comando GOSUB es una sentencia de control que hace que el programa salte a la subrutina especificada mediante la etiqueta. Este comando admite como etiqueta tanto números como cadena de caracteres alfanuméricos.

Sintaxis: **GOSUB { etiqueta }**

4.- RETURN

El comando RETURN retorna la ejecución del programa desde una subrutina al programa principal

Sintaxis: **RETURN**

Ejemplo:

```

CLS
.....
..... Líneas del programa principal
.....
GOSUB PALOMA ; REM: con esta línea el programa salta a la subrutina etiquetada
..... con el nombre PALOMA
..... Líneas del programa principal
.....
END

PALOMA ; REM: a partir de aquí comienza la subrutina
.....
..... Líneas de la subrutina
.....
RETURN

```

Ejemplo GOSUB:

```

REM: ejemplo de GOSUB
CLS
PRINT " Esto es un ejemplo de GOSUB"
PRINT ; REM : esta línea es para separar una fila la impresión anterior de la que
..... sigue.
PRINT " Voy a saltar a un subprograma y escribir algo"
GOSUB subprograma 1
PRINT
PRINT " Hemos regresado al programa principal"
END

REM: en la línea siguiente comenzamos el subprograma poniendo su etiqueta
Subprograma 1
SLEEP 3
CLS
PRINT "Ya estamos en el subprograma"
PRINT
PRINT " Espero 3 segundos y regresamos al programa principal"
SLEEP 3
RETURN

```

II.- SALTOS CONDINCIONALES

1.- Comandos IF / THEN

Ejemplo: IF A = 20 THEN GOTO 35

Ejemplo: IF A > 33 THEN GOTO 20

Ejemplo: IF A < 56 THEN GOTO 20

Ejemplo: IF A >= 61 THEN GOTO 100

Ejemplo: IF A\$ = "Paloma" THEN GOTO 35

Ejemplo: IF A = 20 THEN GOSUB Canario

Ejemplo: IF A > 33 THEN GOSUB Telefono

Ejemplo: IF A < 56 THEN GOSUB 20

Ejemplo: IF A >= 61 THEN GOSUB Andres

Ejemplo: IF A\$ = "Paloma" THEN GOSUB 35

Ejemplo: vamos a realizar un programa que nos permita seleccionar entre tres opciones 1, 2 y 3. Dependiendo de la tecla que sea presionada lo encadenara a una etiqueta de una subrutina

```

REM: ejemplo GOSUB e IF / THEN
regresar ; REM: esto es una etiqueta
CLS
PRINT " Menú del programa "
PRINT "-----"
PRINT
PRINT " 1. Abrir Archivo"
PRINT " 2. Salvar Archivo"
PRINT " 3. Guardar Programa"
PRINT
LOCATE 8, 10
INPUT " Selecciona lo que deseas hacer " ; número
IF número = 1 THEN GOSUB opcion1
IF número = 2 THEN GOSUB opcion2
IF número = 3 THEN GOSUB opcion3
ELSE PRINT " Debes seleccionar un numero del 1 al 3 "
SLEEP 2
GOSUB regresar
END

opcion1
LOCATE 10,10
PRINT " Opción 1 seleccionada"
RETURN

Opcion2
LOCATE 10,10
PRINT " Opción 2 seleccionada"
RETURN

Opcion3
LOCATE 10,10
PRINT "Has elegido finalizar el programa"
RETURN

```

2.- BIFURCACIÓN MÚLTIPLE

1) Comando ON-GOTO

Sintaxis: **ON { Variable } GOTO , dato1, dato2, dato3,**

Ejemplo: ON A GOTO 15, 40, 25, 40, 60

Si la variable A = 1 entonces el control se transfiere a la etiqueta 15, si A = 2 a la etiqueta 40, si A = 3 a la etiqueta 25, etc.

2) Comando ON-GOSUB

Sintaxis: **ON { Variable } GOSUB , dato1, dato2, dato3,**

Ejemplo 1: ON A GOSUB 15, 40, 25, 40, 60

Si la variable A = 1 entonces el control se transfiere a la etiqueta 15, si A = 2 a la etiqueta 40, si A = 3 a la etiqueta 25, etc.

Ejemplo 2: ON A GOSUB Paloma, Canario, Lechuza, Loro, Águila

Si la variable A = 1 entonces el control se transfiere a la etiqueta Paloma, si A = 2 a la etiqueta Canario, si A = 3 a la etiqueta Lechuza, etc.

3.- SALTO SELECTIVO

Comandos: **SELECT CASE número ; CASE ELSE ; END SELECT**

Este comando nos permite ejecutar una sola sentencia de los múltiples sentencias que puede tener un determinado bloque del programa.

Ejemplo: desarrolla un programa que realice las cuatro operaciones matemáticas básicas

```
CLS ; SCREEN 12
PRINT " MENU PRINCIPAL "
PRINT " 1. SUMA "
PRINT " 2. RESTA "
PRINT " 3 MULTIPLICACIÓN "
PRINT " 4. DIVISIÓN "
REM: aquí comienza el bloque select-case
INPUT " INTRODUCER NÚMERO ( 1 - 4 ):" ; número
SELECT CASE número
CASE 1
PRINT " Has seleccionado el número 1"
CASE 2
PRINT " Has seleccionado el número 2"
CASE 3
PRINT " Has seleccionado el número 3"
CASE 4
PRINT " Has seleccionado el número 4"
CASE ELSE
PRINT " El número que has seleccionado no era de 1 a 4"
END SELECT
```

Previamente al comando SELECT CASE se nos pregunta, mediante el comando INPUT, por el valor de una variable llamada "número".

A continuación de "SELECT CASE número" viene un bloque constituido por CASE 1, CASE 2, CASE 3, CASE 4 y CASE 5. El programa ejecutará una sola línea de este bloque en función del número introducido. Si introducimos, por ejemplo un 2, el programa saltará a la línea CASE 2 y ejecutará todas las líneas que están a continuación antes de que aparezca CASE 3. Cuando alcance CASE 3 el programa saltará a END SELECT

CASE ELSE solamente es leído en el caso de que no introduzcamos un número comprendido entre 1 y 4.

Ejemplo:

```
CLS
INPUT " Introduce número de 1 a 8 " ; número
SELECT CASE número
CASE 1 TO 2
PRINT " Has seleccionado el número 1 ó 2 "
CASE 3 TO 4
PRINT " Has seleccionado el número 3 ó 4 "
CASE 5 TO 6
PRINT " Has seleccionado el número 5 ó 6 "
CASE 7 TO 8
PRINT " Has seleccionado el número 7 u 8 "
END SELECT
```

El nuevo comando **TO** que hemos agregado nos permite introducir un rango dentro de la sentencia **SELECT CASE**.