Programando con B4X

Tema 12 – Bucles

Version 1.0, abril 2021

Autor original: Prokopis Leon

Traducido al español por José Miguel López

Tema 12 – Bucles

⏱ 4h

Lo que los estudiantes aprenderán
• ¿Qué son los bucles? • Do While • Do Until • For – Next • Algoritmos con bucles

Las estructuras de repetición son las más importantes de un lenguaje de programación. Un bucle o lazo repite un conjunto de sentencias hasta que una condición se alcanza. En una estructura de repetición se establece una pregunta. Si la respuesta es correcta, se ejecuta el bucle. Esta misma pregunta se pregunta una y otra vez hasta que la respuesta sea falsa y entonces no se ejecutan las acciones de dentro del bucle. Cada vez que se realiza la pregunta se habla de "repetición" o "iteración".

Un programador que utiliza las mismas líneas de código múltiples veces en un programa puede usar un bucle para ahorrar tiempo, espacio y para facilitar la programación.

Recuerda

Los bucles deben terminar en algún momento, si no, se trata de un error de programación. Un bucle que nunca acaba se llama "bucle infinito". En ese caso el ordenador se queda atrapado en una repetición perpetua sin posibilidad de salir de ella y sin que sea consciente de lo que sucede.

En B4X hay 4 **sentencias de repetición**: For, Do-While, Do Until, For each.

La sentencia Do-While

Imagina que quieres mostrar en pantalla con la sentencia "Log" los números del 1 al 5. El código que deberías escribir sería el siguiente:

```
Log(1)
Log(2)
Log(3)
Log(4)
Log(5)
```

Como ves, si quisieras escribir más números tendrías que escribir tantas sentencias Log como números quieras mostrar.

La sentencia Do While se escribe de la siguiente forma:

Do While *Condición* Sentencias ... **Loop**	**Mientras** la condición sea cierta Ejecutar las sentencias ... **Volver a la condición.**

Imagen 1. Sentencia Do While

El ejemplo anterior para escribir 5 números sería así:

```
Private i as Int = 1

Do While i <= 5
   Log(i)
   i = i + 1
Loop
```

La variable **i** cuenta cuántas veces se ha ejecutado el bucle.

La condición comprueba si el contenido del contador excede el límite fijado por el programador (p. ej. 5) y, si sucede, el bucle finaliza.

Antes de que la iteración del bucle acabe, el contador se incrementa en 1. A este valor también se le llama "paso".

Este paso puede ser negativo o positivo, pero nunca puede ser 0, porque entonces el bucle nunca acabaría.

Recuerda

Cuando el contador comienza con un valor menor que el final, el **paso** debe ser **positivo**.

Cuando el contador empieza con un valor mayor que el final, el **paso** debe ser **negativo**.

Ejemplos de la sentencia Do While.

Ejemplo 1

Mostrar todos los enteros del 100 al 1

```
Private i as  Int  = 100

Do While i >= 1
  Log(i)
  i = i -  1
Loop
```

En este ejemplo, el contador comienza con el valor 100 y el bucle acaba cuando alcance el valor 0 (va de un valor mayor a uno menor). Fíjate que en este caso el **Paso** es **Negativo** (-1).

Ejemplo 2

Mostrar los números pares del 1 al 100

Aquí mostramos dos soluciones; la primera usa una sentencia "if" dentro del bucle para comprobar si el número es par y ejecutar así la sentencia "log" con el número. La otra solución usa como **valor inicial** en el **contador** "i" el número 2 y en cada **paso** suma 2; es un algoritmo más rápido.

```
Private i as  Int  = 1

Do While  i  <= 100
  If I  mod 2 = 0
then
    Log(i)
  End if
  i = i +  1
Loop
```

```
Private i as  Int  =  2

Do While  I <= 100
  Log(i)
  i = i + 2
Loop
```

Ejemplo 3 – Algoritmo de suma

Construye un programa que genere 10 números y calcule su suma. Los números deben estar en el intervalo -100 a 100

En este ejemplo usamos la función "Rnd" que se emplea así:

Recuerda

La función **Rnd (Primer Valor, Último Valor)** devuelve un número entre el primer y el último valor.

Ej.: A = Rnd(1, 10) guarda en la variable "A" un número entre 1 y 10

```
Private I  As  Int  =  1
Private A  as Int
Private intSuma  as Int  = 0

Do While  i  <= 10
  A = Rnd(-100, 100)
  intSuma =  intSuma  +  A

  i = i +  1
Loop
```

Ejemplo 4 – Algoritmo para contar

Construye un programa que genere 10 números entre -1000 y 1000 y cuente cuántos negativos hay.

```
Private I  As  Int  =  1
Private A  as Int
Private intContador  as Int  = 0

Do While  i  <= 10
  A = Rnd(-100, 100)
  If A < 0  then
    intContador = intContador + 1
  End If
  i = i +  1
Loop
```

Ejemplo 5 – Algoritmo Máximo-Mínimo

Construye un programa que genere 10 número entre -1000 y 1000 y que calcule el mayor y el menor.

```
Private I  As  Int  =  1
Private A  as Int
Private intMax,  intMin  as
Int

A = Rnd(-100, 100)    1
intMax = A          2
intMin = A
Do While  i  <=  9
   A = Rnd(-100, 100)

   If intMax  < A  then
     intMax = A
   End If                    3

   If intMin  > A  then
     intMax = A
   End If

   i = i +  1
Loop
```

1. Primero creamos un número fuera del bucle while.
2. Fijamos un valor inicial para intMax e intMin igual al primer número generado, ya que no hay ninguno con el que comparar de momento.
3. Para cada nuevo número, comprobamos si es menor que intMin (si es así, reemplazamos intMin por el contenido de "A"). Si es mayor que intMax, reemplazamos intMax por A.

Bucles con un número de repeticiones desconocidas

A menudo en programación no se sabe inicialmente cuántas veces se repetirá un bucle. Esto suele ser así cuando leemos un valor introducido por el usuario o bien cuando realizamos algún cálculo dentro del bucle.

Ejemplo 6 – Número de repeticiones indeterminado

Crea un programa que continuamente lea números y cuente cuántos hay pares. El programa acabará cuando se introduzca un número menor que 0.

```
Private A  as Int
Private intContador  as Int  = 0

A = Rnd(-100, 100)    1
Do While A  >  0
                    2
   If A mod 2 = 0  then
     intContador =  intContador  + 1
   End If

   A = Rnd(-100, 100)    3
Loop
```

1. Generamos un número fuera del bucle.
2. La condición del Do-While comprueba si el número A está en el rango correcto.
3. Se lee un nuevo número antes de finalizar la iteración del bucle.

Ejemplo 7 – Número de repeticiones indeterminado

Crea un programa que lea número y calcule su suma. El programa acabará cuando la suma supere 200.

```
Private A  as Int
Private intSuma  as Int  = 0

Do While intSuma <= 200
   A = Rnd(-100, 100)
   intSuma = intSuma + A
Loop
```

1. La condición comprueba que la suma no supere 200.
2. Se lee un número y el programa lo añade a la suma. La condición se comprueba de nuevo en el bucle Do While.

La sentencia Do Until

El funcionamiento del bucle Do Until es similar al Do While. La única diferencia es que la condición para finalizar el bucle es la contraria que en el bucle Do-While. La comprobación se realiza en ambos casos al principio del bucle. Las sentencias de dentro del bucle no se ejecutarán si la condición es Falsa.

Recuerda

La condición de la sentencia Do Until es la negación de Do While.

Ejemplo 1

Muestra todos los números del 100 al 1

```
Private i as  Int  = 100

Do Until  i  <  1
  Log(i)
  i = i -  1
Loop
```

Ejemplo 2

Mostrar todos los números pares del 1 al 100.

```
Private i as  Int  = 1

Do Until  I  > 100
  If I  mod 2 = 0  then
    Log(i)
  End if
  i = i +  1
Loop
```

```
Private i as  Int  =  2

Do Until  i  >  100
  Log(i)
  i = i + 2
Loop
```

Ejemplo 3 – Algoritmo para sumar

Construye un programa que genere 10 números y calcule su suma. Los números deben estar en el intervalo -100 a 100

```
Private I  As  Int  =  1
Private A  as Int
Private intSuma  as Int  = 0

Do Until  I  >  10
  A = Rnd(-100, 100)
  intSuma =  intSuma  +  A

  i = i +  1
Loop
```

Ejemplo 4 – Algoritmo para contar

Construye un programa que genere 10 números entre -1000 y 1000 y cuente cuántos negativos hay.

```
Private i  as  Int  =  1
Private A  as Int
Private intCounter  as Int  = 0

Do Until  i  >  10
  A = Rnd(-100, 100)
  If A < 0  then
    intCounter = intCounter + 1
  End If
  i = i +  1
Loop
```

Ejemplo 5 – Algoritmo Máximo-Mínimo

Construye un programa que genere 10 número entre -1000 y 1000 y que calcule el mayor y el menor.

```
Private I  As  Int  =  1
Private A  as Int
Private intMax,  intMin  as Int

A = Rnd(-100, 100)
intMax = A
intMin = A
Do Until  I  >  9
  A = Rnd(-100, 100)

  If intMax  < A  then
    intMax = A
  End If

  If intMin  > A  then
    intMax = A
  End If

  i = i +  1
Loop
```

Ejemplo 6 – Número de repeticiones indeterminado

Crea un programa que continuamente lea números y cuente cuántos hay pares. El programa acabará cuando se introduzca un número menor que 0.

```
Private A  as Int
Private intContador  as Int  = 0

A = Rnd(-100, 100)
Do Until  A  <  0

  If A mod 2 = 0  then
    intContador =  intContador  + 1
  End If

  A = Rnd(-100, 100)
Loop
```

Ejemplo 7 – Número de repeticiones indeterminado

Crea un programa que lea número y calcule su suma. El programa acabará cuando la suma supere 200.

```
Private A  as Int
Private intSuma  as Int  = 0

Do Until  intSuma  >  200
   A = Rnd(-100, 100)
   intSuma = intSuma + A
Loop
```

Bucle For

El bucle For es seguramente la sentencia de repetición más simple:

```
for i  = n1  to  n2  Step  n3
        Sentencias
        ...
Next
```

Donde:

i es la variable contador

n1 el valor inicial del contador

n2 el valor final del contador

n3 el paso tras cada iteración

- Al inicio del bucle, la variable "i" contendrá el valor "n1".
- Se ejecutan las sentencias dentro del bucle.
- Al final de la iteración, el valor de "i" se incrementa o decrementa en "n3"
- Se comprueba si "i" ha superado el valor "n2":
  - Si el paso es positivo y "i" es menos o igual al valor final, entonces el bucle se ejecuta de nuevo
  - Si el paso es negativo y "i" es mayor o igual al valor final, entonces el bucle se ejecuta de nuevo.

Recuerda

Comparado con los bucles **Do While** y **Do Until,** el bucle **For**:

- No necesita inicializar el contador.
- No necesita efectuar ninguna operación con el paso.
- Siempre sabe el número de repeticiones que hará (aunque se puede usar la sentencia "exit" para acabar el bucle cuando quieras).

Ejemplos de bucle For

Ejemplo 1

Mostrar los números del 100 al 1

```
Private I As Int

For i = 100 to 1 step -1
  Log(i)
Next
```

Ejemplo 2

Mostrar los números pares del 1 al 100

```
Private I  As  Int

For i = 1  to  100
  If i  mod 2 = 0 then
    Log(i)
  End If
Next
```

Ejercicios

1. Escribe un programa que lea un número entero K entre 1 y 200 y calcule la suma de 1+2+3+4+…+K.
2. Escribe un programa donde el usuario introduzca números y calcule la media de los números. El programa finaliza cuando se introduce el 0.
3. Un cliente quiere comprar juguetes para regalar a 10 niños. Desea que el precio total no supere 200€. Escribe un programa que:
    a. Genere un precio para un juego (número aleatorio entre 10 y 50, usando la función Rnd(10, 50)).
    b. Calcule el precio total de los juegos comprados.
    c. Compruebe si el dinero se ha acabado.
    d. Muestre al final el precio total de los juegos y el número de juegos que se han comprado.
4. Un año bisiesto es aquel al que se le añade un día adicional (en un calendario lunar, se añade un mes) para mantener el año sincronizado con el año astronómico o año solar. Para comprobar si un año es bisiesto:
    - Si el año no es divisible por 4, NO es bisiesto
    - Si no, si el año no es divisible por 100, entonces el año es bisiesto.
    - Si no, si el año no es divisible por 400, entonces es un año normal.
    - Si no, es un año bisiesto.

    Escribe un programa que muestre los años bisiestos desde 1900 a 2100.

(https://es.wikipedia.org/wiki/Año_bisiesto)

5. Construye un programa que con la ayuda de la Tortuga genere polígonos con un número de ángulos introducido por el usuario en el campo adecuado. El programa incluirá una interfaz donde haya un botón de inicio y un botón para borrar la pantalla que borrará la pantalla cuando se pulse y colocará a la tortuga en el centro de la pantalla.
6. Dibuja las siguientes formas con la Tortuga: