INSTITUTO DE EMPRENDEDORES

Curso	HERRAMIENTAS DE DESARROLLO DE SOFTWARE
Docente	Ing. Eric Gustavo Coronel Castillo
Tema	Estructuras Repetitivas y Programación Orientada a Objetos

Objetivos

- Aplicar estructuras repetitivas en la solución de problemas informáticos.
- Aplicar la programación en capas.
- Aplicar la programación orientada a objetos.
- Creación de GUI

Fundamentos

Estructura: Do - Loop

Sintaxis:

Opción 1	Opción 2
Do While | Until <condición> (Instrucciones) [Exit Do] Loop	Do (Instrucciones) [Exit Do] Loop While | Until <condición>

`While`|`Until` son las palabras clave que se utilizan para repetir el bucle. Solamente se puede utilizar una de las dos. Si utiliza `While` el bucle se repetirá hasta que la condición se falsa. Por el contrario si utiliza `Until`, el bucle se repetirá hasta que la condición resulte verdadera.

La instrucción `Exit Do` se utiliza para salir del bucle `Do`.

Si utilizas la **opción 1**, primero se va a evaluar la condición y, depende de la instrucción que usted utilice, se ejecutará el lazo si la condición es verdadera (si utiliza `While`) o si la condición es falsa (si utiliza `Until`). Mientras que con la **opción 2**, el bucle se ejecutará por lo menos una vez.

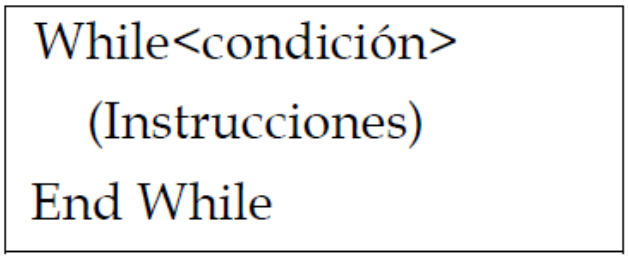
Ejemplo 1

Que valores imprime el siguiente programa:

```
Module Module1
    Sub main()
        Dim cont As Integer
        cont = 0
        Do While (cont < 10)
            cont = cont + 1
            If cont = 5 Or cont = 8 Then
                Continue Do
            End If
            Console.WriteLine(cont)
        Loop
        Console.ReadLine()
    End Sub
End Module
```

Estructura: While - End While

Sintaxis:



La estructura `While` se utiliza para repetir un conjunto de acciones cuando se verifica una condición.

Ejemplo 2

Que imprime el siguiente programa:

```
Module Module2
    Sub main()
        Dim n As Integer
        Dim valor As Integer
        n = 1
        valor = 0
        While n <= 10
            valor = valor + n * n
            n = n + 1
        End While
        Console.WriteLine("valor = " & valor.ToString())
        Console.ReadLine()
    End Sub
End Module
```

Estructura: For - Next

Sintaxis:

```
For contador=<valor inicial> To <valor Final> [Incremento]
   (Instrucciones)
   [Exit For]
  Next [Contador]
```

La estructura `For - Next` se utiliza para repetir un conjunto de instrucciones, un número dado de veces.

Dónde:

- Contador: cualquier variable numérica
- Valor Inicial: Es el valor inicial del contador
- Valor Final: Es el valor final del contador
- Instrucciones: Son las instrucciones que se repetirán su ejecución.
- `Exit For`: Es opcional y se utiliza para salir del bucle `For`.
- `Next`: Marca el final de la instrucción `For`. Tan pronto como el programa se tope con la instrucción `Next`, se agregará el valor de incremento al contador y la siguiente iteración del bucle tendrá lugar. Una buena práctica de programación consiste en especificar el nombre del contador en la instrucción `Next`, pero no es obligación.

Ejemplo 3

Que imprime el siguiente programa:

```
Module Module3
    Sub main()
        Dim f As Integer = 1
        For n = 2 To 6
            f *= n
        Next
        Console.WriteLine("Resultado: " & f.ToString())
        Console.ReadLine()
    End Sub
End Module
```

Proyectos a Desarrollar

En los siguientes proyectos debe aplicar la programación en capas y la programación orientada a objetos.

Proyecto 1: Analizar Número

Desarrollar un proyecto que permita analizar un número con respecto a las siguientes características:

- Si es primo o no.
- Si es par o impar.
- Si es capicúa o no.

Se recomienda implementar la siguiente clase:

```
Public Class AnalizaNumero

    Function esPrimo(ByVal n) As Boolean
        Dim primo As Boolean = True
        For i = 2 To n - 1
            If n Mod i = 0 Then
                primo = False
                Exit For
            End If
        Next
        Return primo
    End Function

    Function esPar(ByVal n) As Boolean
        Dim par As Boolean = True
        If n Mod 2 = 1 Then
            par = False
        End If
        Return par
    End Function

    Function esCapicua(ByVal n) As Boolean
        Dim capicua As Boolean = True
        ' Falta implementar
        Return capicua
    End Function

End Class
```

Proyecto 2: MCD y MCM de Dos Números

Desarrollar un proyecto que permita calcular el MCD y MCM de dos números.

Se recomienda implementar la siguiente clase:

```
Public Class Mate

    Function mcm(ByVal n1 As Integer, ByVal n2 As Integer) As Integer
        ' Falta implementar
        Return 0
    End Function

    Function mcd(ByVal n1 As Integer, ByVal n2 As Integer) As Integer
        ' Falta implementar
        Return 0
    End Function

End Class
```

Proyecto 3: Encontrar Suma

Desarrollar un proyecto que permita encontrar la suma de los números comprendidos entre n1 y n2, siendo n1 < n2.

Se recomienda implementar la siguiente clase:

```
Public Class Suma

    Function calcular(ByVal n1 As Integer, ByVal n2 As Integer) As Integer
        ' Falta implementar
        Return 0
    End Function

End Class
```