Tema 4: Procedimientos y funciones.

- Definición y tipos.
- Procedimientos Sub.
- Procedimientos Function.
- Accesibilidad.
- Paso de argumentos.
- Procedimientos de eventos.

Definición y tipos.

- Bloque de código que comienza por una declaración de procedimiento y termina por la palabra End correspondiente
  - Todo el código ejecutable se encuentra dentro de un procedimiento.
- Tipos de declaraciones de procedimiento.
  - Procedimientos Sub.
    - ✓ Ejecutan acciones pero no devuelve un valor al código de llamada.
  - Procedimientos Function.
    - ✓ Devuelven un valor al código de llamada.
  - Procedimientos de evento.
    - ✓ Procedimientos Sub que se ejecutan en respuesta a un evento producido por una acción del usuario en la interfaz o desencadenado por el programa.
  - Procedimientos Property Get y Property Set
    - ✓ Devuelven y asignan valores de propiedades en clases y módulos.
  - Procedimientos Operator.
    - ✓ Se utilizan para sobrecargar los operadores.

Procedimientos Sub.

- Declaración.

[*accesibilidad*]Sub *nombreProc* [(*listaArgumentos*)]
    [*bloque de instrucciones*]
[Exit Sub | Return]
    [*bloque de instrucciones*]
End Sub

- Llamada.

[Call] *nombreProc* [(*listaArgumentos*)]

Procedimientos Function.

- Declaración.

  [*accesibilidad*]Function *nombreFunc* [(*listaArgumentos*)][As *tipoDato*]
  [*bloque de instrucciones*]
  [Exit Function]
  [*bloque de instrucciones*]
  End Function

- Valores de retorno.
  - El valor de retorno de la función se puede hacer mediante:

    Return expresión
    nombreFunción = expresión

  - Si el código no pasa por alguna de estas instrucciones, devuelve el valor por omisión correspondiente al tipo de la función.
    - ✓ El tipo de dato que devuelve por omisión es de tipo Object.
      - Si Option Strict está puesto a On, genera un error.

Accesibilidad

- Los distintos tipos de accesibilidad son:
  - Public (tipo por omisión).
    - ✓ El método puede utilizarse en cualquier parte del proyecto, desde otro proyecto o desde un ensamblado generado por el proyecto.
  - Private.
    - ✓ El método puede utilizarse sólo dentro del ámbito donde ha sido declarado (módulo, función, etc.).
  - Friend.
    - ✓ El método puede utilizarse desde el ámbito donde ha sido declarado y en cualquier sección del mismo ensamblado.
  - Protected.
    - ✓ El método sólo puede utilizarse dentro de la clase donde ha sido declarado y en las clases derivadas de ésta.
  - Protected Friend.
    - ✓ Una combinación de las dos anteriores

Paso de argumentos

- Declaración de cada argumento.

  [Optional] [{ ByVal | ByRef }] [ParamArray]
  nombreArgumento[( )] [As *tipoDato*] [= *valorDefecto*]

- Tipos de los argumentos.
  - Por omisión son argumentos de tipo Object.

- Paso por valor y paso por referencia.
  - Por omisión el paso se hace por valor.
  - Las constantes, los literales, las expresiones y las enumeraciones se pueden pasar por referencia, aunque su valor no cambiará en el programa llamador.
  - Los tipos de referencia (clases, arrays,…) se pueden pasar por valor.
    - ✓ El procedimiento no puede cambiar la referencia, pero si puede cambiar los miembros de la instancia que señala.

Paso de argumentos (II)

- Argumentos opcionales.
  - Se señalan mediante la cláusula Optional.
  - Se pueden omitir en la llamada.
  - Deben tener un valor por omisión y ser los últimos argumentos de la lista.
  - Para determinar si un argumento está o no presente en la llamada se puede utilizar un valor centinela.

- Argumentos ParamArray.
  - Se utilizan para pasar un número indeterminado de argumentos.
  - Sólo puede haber un argumento ParamArray y éste debe ser el último de la lista y pasarse por valor.

Paso de argumentos (III)

```
Module Module1
Class clase
        Friend valor As Integer
End Class

Sub main()
        'Ejemplo de uso de parámetros opcionales y ParamArray
        System.Console.WriteLine(Sumar(1, 2, 3, 4))   'Devuelve 10
        System.Console.WriteLine(Sumar(2, 3, 4))      'Devuelve 9
        Dim i As Integer = 1
        System.Console.WriteLine(contador(i))         'Devuelve 2
        System.Console.WriteLine(contador(i, 3))      'Devuelve 4

        'Ejemplo de paso por referencia en tipos de referencia
        Dim cls As New clase()
        cls.valor = 2
        prueba(cls)
        System.Console.WriteLine(cls.valor)           'Devuelve 3
        System.Console.ReadLine()
End Sub
```

Paso de argumentos (IV)

```
'Ejemplo de uso de ParamArray
Function Sumar(ByVal ParamArray elem()) As Integer
        Dim suma As Integer = 0
        Dim i As Integer
        For i = elem.GetLowerBound(0) To elem.GetUpperBound(0)
                suma += elem(i)
        Next
        Return suma
End Function

'Ejemplo de uso de argumentos opcionales
Function contador(ByVal x As Integer, Optional ByVal conta As Integer= Integer.MinValue) As integer
        If conta = Integer.MinValue Then
                Return x + 1
        Else
                Return x + conta
        End If
End Function

'Ejemplo de paso por valor en tipos de referencia
Sub prueba(ByVal cls As MiClase)
        cls.valor = contador(cls.valor)
End Sub

End Module
```

Procedimientos de evento.

- Procedimientos que se ejecutan en respuesta a un evento desencadenado por una acción del usuario o un suceso del programa.
  - No se les suele llamar por el método normal.