SOBRECARGA DE PROCEDIMIENTOS

Si bien el uso de parámetros opcionales es un medio para ahorrar al programador el paso de los mismos en situaciones en las que no son necesarios, resulta una solución un tanto artificiosa, ya que lo que realmente hace es complicar más que facilitar la escritura de código.
VB.NET aporta al lenguaje una nueva técnica que permite obviar el uso de parámetros opcionales por una solución más elegante y flexible: los procedimientos sobrecargados.
Antes de explicar en qué consiste un procedimiento sobrecargado, situémonos en el siguiente escenario: Necesitamos mostrar los datos de un empleado de dos formas, en función del modo de consulta. Por un lado visualizaríamos su nombre, domicilio y localidad; y por otra parte su edad, DNI y fecha de alta en la empresa.
Con lo que sabemos hasta el momento, podríamos resolver este problema escribiendo un procedimiento con parámetros opcionales, y según pasáramos un valor u otro, mostrar la información correspondiente.
El Código fuente 131 muestra este modo de resolver el problema.

CONSOLE131

```
Module Module1
    Sub Main()
        ' mostrar datos del empleado
        ' en función del nombre
        VerDatosEmpleado("Pedro")
        ' mostrar datos del empleado
        ' en función de la edad
        VerDatosEmpleado(, 28)

        Console.ReadLine()
    End Sub
    Sub VerDatosEmpleado(Optional ByVal Nombre As String = "X", Optional ByVal Edad As Integer = 999)

        If Nombre <> "X" Then
            Console.WriteLine("Nombre del empleado: {0}", Nombre)
            Console.WriteLine("Domicilio: La Rioja 1450")
            Console.WriteLine("Localidad: Córdoba")
        End If

        If Edad <> 999 Then
            Console.WriteLine("Edad del empleado: {0}", Edad)
            Console.WriteLine("DNI:21555666")
            Console.WriteLine("Fecha de alta en la empresa: 10/4/1997")
        End If

        Console.WriteLine()
    End Sub
```

```
End Module
```

El uso de parámetros opcionales, como acabamos de constatar, resulta engorroso, ya que nos obliga a comprobar qué valor ha sido pasado y mostrar los datos correspondientes en consecuencia. Tenemos además, un inconveniente añadido, y es que podemos pasar los dos parámetros a la vez, con lo que se mostrarían todos los datos, cuando lo que queremos es visualizar un grupo u otro en cada llamada.
Una aproximación diferente al problema sería escribir dos procedimientos distintos, y llamar a uno u otro según los datos que necesitemos. Ver Código fuente 132.

CONSOLE132

```
Sub Main()
        ' mostrar datos del empleado según nombre
        VerEmpleNombre("Pedro")

        ' mostrar datos del empleado según edad
        VerEmpleNum(28)

        Console.ReadLine()
    End Sub

    Public Sub VerEmpleNombre(ByVal Nombre As String)
        Console.WriteLine("Datos empleado por nombre")
        Console.WriteLine("Nombre del empleado: {0}", Nombre)
        Console.WriteLine("Domicilio: La Rioja 1450")
        Console.WriteLine("Localidad: Córdoba")
        Console.WriteLine()
    End Sub

    Public Sub VerEmpleNum(ByVal Edad As Integer)
        Console.WriteLine("Datos empleado por edad")
        Console.WriteLine("Edad del empleado: {0}", Edad)
        Console.WriteLine("DNI:21555666")
        Console.WriteLine("Fecha de alta en la empresa: 10/4/1997")
        Console.WriteLine()
    End Sub

End Module
```

Sin embargo, esta solución nos obliga a tener que saber varios nombres de procedimiento, con lo que tampoco ayuda mucho a simplificar el código.
¿No sería ideal, disponer de un único nombre de procedimiento y que este fuera lo suficientemente inteligente para mostrar los datos adecuados en cada caso?, pues esta característica está implementada en VB.NET a través de la sobrecarga de procedimientos. La sobrecarga de procedimientos es una técnica que consiste en crear varias versiones de un mismo procedimiento, distinguiéndose entre sí por la lista de parámetros o protocolo de llamada del procedimiento.

CONSOLE134

```
Module Module1

    Sub Main()
        Dim Dias As Integer
        ' mostrar datos del empleado según nombre
        VerEmpleado("Pedro")

        ' mostrar datos del empleado según edad
        Dias = VerEmpleado(28)
        Console.WriteLine("Días libres del empleado: {0}", Dias)
        Console.WriteLine()

        ' mostrar salario pasando las horas trabajadas
        VerEmpleado(25, 80)

        Console.ReadLine()
    End Sub
    Sub VerEmpleado(ByVal Nombre As String)
        Console.WriteLine("Datos empleado por nombre")
        Console.WriteLine("Nombre del empleado: {0}", Nombre)
        Console.WriteLine("Domicilio: La Rioja 1450")
        Console.WriteLine("Localidad: Córdoba")
        Console.WriteLine()
    End Sub

    Function VerEmpleado(ByVal Edad As Integer) As Integer
        Dim DiasLibres As Integer
        Console.WriteLine("Datos empleado por edad")
        Console.WriteLine("Edad del empleado: {0}", Edad)
        Console.WriteLine("DNI:21555666")
        Console.WriteLine("Fecha de alta en la empresa: 10/4/1997")
        Console.WriteLine()

        DiasLibres = 5
        Return DiasLibres
    End Function

    Sub VerEmpleado(ByVal PrecioHora As Integer, ByVal HorasTrabajadas As Long)
        Dim Salario As Long

        Salario = PrecioHora * HorasTrabajadas
        Console.WriteLine("Salario según horas: {0}", Salario)
        Console.WriteLine()
    End Sub
End Module
```

En este código hemos creado tres versiones sobrecargadas del procedimiento VerEmpleado( ). En una mostramos los datos del empleado según el nombre; en otra también mostramos otro conjunto de datos según la edad y además, al ser una función, devolvemos el número de días libres del empleado; trabajadas, que pasamos al protocolo de llamada. Desde Main( ) por lo tanto, siempre llamamos al procedimiento VerEmpleado( ).