aprenderaprogramar.com

Visual Basic y .NET: Procedimientos Sub y funciones Function. Parámetros. Organizar programas en módulos. (CU00338A)

Sección: Cursos

Categoría: Curso Visual Basic Nivel I

Fecha revisión: 2029

Autor: Mario R. Rancel

29

PROGRAMACIÓN DE MÓDULOS. TIPOS. PROCEDIMIENTOS SUB, FUNCIONES FUNCTION.

Visual Basic, como no podía ser de otra manera, está orientado a la programación modular. Ya hemos comentado que puede haber cierta confusión terminológica: el concepto de módulo que hemos usado en cursos de bases de programación de aprenderaprogramar.com no coincide con el concepto de módulo en la terminología de Visual Basic. Vamos a hacer una clasificación libre tratando de conectar el Visual Basic con el desarrollo que hemos realizado en otros cursos.

DESCRIPCIÓN	TERMINOLOGÍA DE VISUAL BASIC
Formulario: ventana de formulario, objetos sobre el formulario, código	Módulo de formulario (fichero *.frm ó *.vb, según versiones)
Código de declaraciones y definiciones de procedimientos, tipos y datos con carácter público para otros módulos	Módulo estándar (fichero *.bas ó *.vb, según versiones)
Código de definición de una clase de objeto	Módulo de clase (Fichero *.cls ó *.vb, según versiones)
Módulo de código que se ejecuta cuando tiene lugar un evento determinado	Procedimiento conducido por eventos (Sub)
Módulo de código que se ejecuta cuando es llamado desde algún punto del programa	Procedimientos generales (Sub)
Módulo de código que se ejecuta cuando es llamado desde algún punto del programa y devuelve un valor	Procedimiento función (Function)

Dado que el código contenido en un módulo estándar de Visual Basic es accesible desde distintos formularios del programa, será ventajoso colocar en este módulo todo lo que queramos disponer como "código compartido".

Buscando analogías con el desarrollo que hacemos en los cursos de pseudocódigo de aprenderaprogramar.com, usaremos el evento que se produce cuando ordenamos la ejecución de nuestros programas (carga del formulario o Form_Load) para disponer en él el código del "algoritmo principal" o guía del programa y el resto del código irá ordenado en procedimientos conducidos por eventos, procedimientos generales ó procedimientos función.

Habíamos dicho que un módulo no se ejecuta hasta que es llamado a ejecutarse desde el algoritmo principal de acuerdo con la sintaxis de pseudocódigo:

```
Llamar [Nombre del Módulo]
```

Sin embargo con programación guiada por eventos esto es sólo parcialmente cierto: un módulo puede ejecutarse por ser llamado desde algún punto del código pero también sin ser llamado desde el código, cuando tiene lugar un determinado evento.

La declaración de un **procedimiento general** se realizará mediante la sintaxis:

```
[Carácter Público o Privado] Sub [Nombre del procedimiento]([Parámetros])
```

EJEMPLO

```
Private Sub Calcular()
```

Todo procedimiento tiene un final indicado mediante End Sub, aunque se puede provocar una salida forzada usando la expresión Exit Sub.

El carácter público o privado se establece mediante las palabras clave Public o Private, que dan lugar a que dicho procedimiento pueda invocarse o no desde otros módulos. Sub indica que se está declarando un procedimiento Sub con un nombre determinado, y los paréntesis están destinados a contener parámetros requeridos por el procedimiento para ser invocado. El paso de parámetros lo veremos más adelante, por lo que consideraremos de momento procedimientos sin parámetro (paréntesis vacíos).

La declaración de un **procedimiento conducido por eventos** la realizaremos usando las listas desplegables de objetos y eventos que vimos cuando hablamos de botones (Buttons o Command Buttons). De esta manera, al seleccionar un objeto y un evento automáticamente nos aparecerá un código del tipo (según versiones habrá diferencias):

```
Private Sub Text1_Change()


End Sub
```

```
Private Sub TextBox1_TextChanged(ByVal sender As System.Object,
ByVal e As System.EventArgs) Handles TextBox1.TextChanged


End Sub
```

Este es un procedimiento conducido por eventos que se invoca cuando el texto contenido en el TextBox cambia. No prestaremos atención de momento a los parámetros que puedan aparecer automáticamente.

La declaración de una **función** se realizará mediante la sintaxis:

```
[Carácter Público o Privado] Function [Nombre de la función]([Parámetros]) As [Tipo de dato]
```

EJEMPLO

```
Private Function Hipotenusa() As Single
.
.
.
End Function
```

Como hemos dicho, una función devuelve un valor, de ahí que especifiquemos un tipo de dato para ella. En caso de no especificarse, el tipo será Variant u Object (dependiendo de la versión) por defecto.

El flujo para una función sigue las mismas reglas que para un procedimiento general o conducido por eventos: al llegar a End Function el control vuelve a la sentencia inmediatamente posterior a la llamada efectuada. Se puede provocar la salida forzada de una función utilizando la expresión Exit Function.

Los procedimientos pueden insertarse en el programa en cualquier orden, aunque siempre será recomendable tratar de disponerlos en el mismo orden que está previsto que se ejecuten.

La llamada de un procedimiento general o conducido por eventos se realiza, cuando no hay parámetros que pasar, simplemente escribiendo su nombre, o bien usando Call [Nombre]. La llamada a una función se hará normalmente para obtener un valor o asignar un valor a una variable, en expresiones del tipo:

```
Label1.Text = Label1.Text & [Nombre de la función]()
Variable = [Nombre de la función]()
If [Nombre de la función]() > [Variable]
```

Hay que recordar siempre que una "función" ejecuta un código y devuelve un valor: podríamos decir que es un híbrido entre una variable y un procedimiento.

La llamada a un procedimiento desde sí mismo es posible, dando lugar a un anidamiento o recursión. Habrá de existir una condición que evolucione para dar lugar a la salida de la recursión, regresando el control del flujo a la instrucción posterior desde la que se autollamó el módulo. No vamos a desarrollar contenidos relativos a la recursión, pero si tienes interés en profundizar en esta técnica de programación te remitimos a profundizar en esta materia mediante otros cursos que se ofrecen en aprenderaprogramar.com.

Una llamada del algoritmo principal a sí mismo, que en su momento escribimos como Llamar Inicio, sería posible con Visual Basic, pero no vamos a entrar a detallar este tipo de cuestiones que raramente se usarán.

EJERCICIO

Transformar el siguiente programa "Comunicado" en código de Visual Basic.

1. **Inicio [PROGRAMA Comunicado curso Visual Basic aprenderaprogramar.com]**
   2. Mostrar “Comunicado de la empresa”
   3. Llamar Saludo
   4. Llamar Comunicado
   5. Llamar Despedida
6. **Fin**

Módulo Saludo

1. Mostrar “Con motivo de la celebración el próximo día 5 del Día Mundial del Medioambiente la empresa saluda a todos los empleados y les agradece el compromiso con el cuidado de la naturaleza”

FinMódulo

Módulo Comunicado

1. Mostrar “Con motivo de dicha conmemoración está previsto realizar un acto de plantación de árboles en los jardines del edificio central el próximo día 5 a las 12 del mediodía al que están todos invitados”

FinMódulo

Módulo Despedida

1. Mostrar “La empresa agradece su participación y les invita a sumarse al programa <<Empleados por una ciudad sostenible>>. Atentamente, El Director General”

FinMódulo

SOLUCIÓN

Código versiones menos recientes de VB:

```
'Curso VB aprenderaprogramar.com
Option Explicit
'Algoritmo principal

Private Sub Form_Load()
MsgBox("Comunicado de la empresa")
Saludo  'Invocación del procedimiento Saludo
Comunicado  'Invocación del procedimiento Comunicado
Despedida  'Invocación del procedimiento Despedida
End Sub

Private Sub Saludo()
MsgBox("Con motivo de la celebración el próximo día 5 del Día Mundial del Medioambiente")
End Sub

Private Sub Comunicado()
MsgBox("Se invita a todos los empleados a un acto de plantación de árboles a las 5 de la tarde en los jardines del edificio central")
End Sub

Private Sub Despedida()
MsgBox("La empresa agradece su participación")
End Sub
```

Código versiones más recientes de VB:

```
REM Curso Visual Basic aprenderaprogramar.com
Option Explicit On
Public Class Form1

    Private Sub Form1_Load(ByVal sender As
System.Object, ByVal e As System.EventArgs)
Handles MyBase.Load
MsgBox("Comunicado de la empresa")
Saludo()  'Invocación del procedimiento Saludo
Comunicado()  'Invocación del procedimiento
Comunicado
Despedida()  'Invocación del procedimiento
Despedida
    End Sub

    Private Sub Saludo()
        MsgBox("Con motivo de la celebración el
próximo día 5 del Día Mundial del
Medioambiente")
    End Sub

    Private Sub Comunicado()
        MsgBox("Se invita a todos los empleados
a un acto de plantación de árboles a las 5 de la
tarde en los jardines del edificio central")
    End Sub

    Private Sub Despedida()
        MsgBox("La empresa agradece su
participación")
    End Sub
End Class
```

El único interés de este ejercicio es reforzar la idea de estructura basada en algoritmo principal e invocación de módulos a la hora de construir nuestros programas.

Próxima entrega: CU00339A

Acceso al curso completo en aprenderaprogramar.com -- > Cursos, o en la dirección siguiente:
http://www.aprenderaprogramar.com/index.php?option=com_content&view=category&id=37&Itemid=61