Visual Basic .NET
Sub Programs

By: Sayed Hassan Adelyar

پروگرامهاي فرعي

Sub Programs

- بسياري پروگرامهاي كمپيوتر داراي دستورهاي زياد بوده و تجارب نشان داده است كه بهتر است اين پروگرامها به **بخش هاي خورد و قابل اداره** تبديل شود. در ويژول بسيک دات نت اين بخش ها ميتواند به انواع مختلف به شمول **ماديول** و **كلاس** باشد. پروگرامر ميتواند **ماديولها** و **كلاسهاي جديد** را در آينده نيز به پروگرام اضافه نمايد. اين **كلاسها** و ماديولها داراي بخشهاي خورد بنام **Procedure** ميباشد. وقتيكه اين Procedure ها در داخل كلاس باشند بنام **Method** ياد ميشوند. **Procedure** ها به سه نوع ميباشند:
  - **Sub Procedures**
  - **Function procedures**
  - **Event procedures**
- اين **پروگرامهاي فرعي** بنام آنها اجرا ميشود. در اثنائي اجرا **نام** و **معلومات ضروري** براي اجراي وظيفه آنها (**پارامترها**) ضروري ميباشد. وقتيكه procedure وظيفه خودرا انجام دهد، **دوباره كنترول** را به **پروگرام صدا دهنده** ميدهد.

Module

Sub Programs

- The **module block** contain **any number** of **subs**, **functions**, and **variables**.
- The **block** is **delimited** by the **Module** and **End Module keywords**.

Sub Procedure

Sub Programs

- A **Sub procedure** is a series of **Visual Basic statements** enclosed by the **Sub** and **End Sub** statements. The **Sub procedure performs** a **task** and then **returns control** to the **calling code**, but it **does not return** a **value** to the calling code.
- **Each time** the procedure is called, its **statements** are **executed**, starting with the first executable statement after the **Sub** statement and ending with the first **End Sub**, or **Exit Sub** statement encountered.
- You **can define** a **Sub** procedure in **modules**, **classes**, and **structures**. By **default**, it is **Public**, which means you can **call** it from anywhere in your **application** that has access to the **module**, **class**, or **structure** in which you defined it. The term, **method**, describes a **Sub** or **Function** procedure that is **accessed from outside** its **defining module**, **class**, or **structure**.
- A **Sub** procedure can take **arguments**, such as **constants**, **variables**, or **expressions**, which are passed to it by the calling code.

Sub Programs

- Declares the name, parameters, and code that define a Sub procedure.

[accessmodifier] [proceduremodifiers]

Sub name [(parameterlist)] [Handles eventlist]

    [statements]

    [Exit Sub]

    [statements]

End Sub

Sub Programs

- ***accessmodifier***
  - **Public**
    - Specifies that one or more declared programming elements have **no access restrictions**.
  - **protected**
    - Specifies that one or more declared programming elements are **accessible** only from **within** their **own class** or **from** a **derived class**.
  - **Friend**
    - Specifies that one or more declared programming elements are accessible only from **within** the **assembly** that contains their declaration**.**

Sub Programs

- **Private**
  - Specifies that one or more declared programming elements are accessible only from within their **declaration context**, including from within any contained types.

Sub Programs

- **proceduremodifiers**
  - **Overloads**
    - Specifies that a **property** or **procedure re-declares** one or more **existing properties** or **procedures** with the **same name**.
  - **Overrides**
    - Specifies that a **property** or **procedure overrides** an **identically named property** or **procedure inherited** from a **base class**.
  - **Overridable**
    - Specifies that a **property** or **procedure** can be **overridden** by an **identically named property** or **procedure** in a **derived class**.

Sub Programs

- **Notoverridable**
  - Specifies that a **property** or **procedure cannot** be **overridden** in a **derived class**.
- **Mustoverrides**
  - Specifies that a **property** or **procedure** is not implemented in this class and must be **overridden** in a **derived class** before it can be **used**.

Sub Programs

- **Static**
- **Specifies** that one or more declared **local variables** are to **continue** to **exist** and **retain** their **latest** values after **termination** of the **procedure** in which they are **declared**.

Sub Programs

- پروگرام ذيل مثال پروگرام فرعي Sub را در كلاس فورم نشان ميدهد:

```
Public Class Form1
Private Sub Button1_Click(ByVal sender As System.Object, ByVal e
    As System.EventArgs) Handles Button1.Click
        showresult(10, 20)
        showresult(40, 50)
        showresult(12, 8)
    End Sub
    Sub showresult(ByVal hours As Integer, ByVal wage As Integer)
        Dim cost As Double
        cost = hours * wage
        MessageBox.Show(cost)
    End Sub
End Class
```

Sub Programs

مثال ذيل، پروگرام فرعي **Sub** در **Module** را نشان ميدهد:

```
Module Module1
    Public c As Integer
    Sub twonum(ByVal a As Integer, ByVal b As Integer)
        c = a + b
    End Sub
End Module

Public Class Form1
Private Sub Button2_Click(ByVal sender As System.Object, ByVal e As
     System.EventArgs) Handles Button2.Click
        Call twonum(12, 7)
        TextBox1.Text = c
    End Sub
End Class
```

Sub Programs

- مثال دوم:

```
Module Payment
Sub PrintPay(ByVal hours As Double, ByVal wage As Decimal)
      TextBox1.Text = "The payment is:  " &  hours * wage
   End Sub
End Module

Public Class Form1
Private Sub Button3_Click(ByVal sender As System.Object, ByVal e As
     System.EventArgs) Handles Button3.Click
      PrintPay(40, 10.5)
      PrintPay(38, 21.75)
      PrintPay(20, 13)
      PrintPay(50, 14)
   End Sub
End Class
```

Sub Programs

- مثال سوم:

```
Module Module1
Sub printpay(ByVal a As Integer, ByVal b As Integer)
        Dim c As Integer
        c = a + b
        MessageBox.Show("Sum is " & c)
        Form1.TextBox1.Text = c
End Sub
End Module
Public Class Form1
Private Sub Button1_Click(ByVal sender As System.Object, ByVal e As
    System.EventArgs) Handles Button1.Click
        printpay(2, 6)
    End Sub
End Class
```

Function

Sub Programs

- A **Function** procedure is a **series of Visual Basic statements** enclosed by the **Function** and **End Function** statements. The **Function** procedure performs a task and then returns control to the calling code. When it returns control, it also **returns** a **value** to the **calling code**.
- **Each time** the procedure is called, its **statements run**, **starting** with the **first executable** statement after the **Function** statement and ending with the first **End Function**, **Exit Function**, or **Return** statement encountered.
- You can define a **Function** procedure in a **module**, **class**, or **structure**. It is **Public** by **default**, which means you can call it from **anywhere** in your **application** that has access to the **module**, **class**, or **structure** in which you defined it.
- A **Function** procedure can take **arguments**, such as **constants**, **variables**, or **expressions**, which are passed to it by the **calling code**.

Sub Programs

[accessmodifier] [proceduremodifiers]

Function name [(parameterlist)] [As returntype] [Handles eventlist]

[statements]

[Exit Function]

[statements]

End Function

Sub Programs

```
Public Class Form1
    Dim starray As Integer()
    Private Sub Button1_Click(ByVal sender As System.Object, ByVal e As
     System.EventArgs) Handles Button1.Click
        TextBox2.Text = getGrade(TextBox1.Text)
    End Sub

    Public Function getGrade(ByVal score As Integer) As String
        If score >= 90 Then Return "Excellent"
        If score >= 80 Then Return "Very Good "
        If score >= 70 Then Return "Good"
        If score >= 60 Then Return "Pass"
        Return "Fail"
    End Function
End Class
```

Sub Programs

```
Public Class Form1

    Private Sub Button1_Click(ByVal sender As System.Object, ByVal e
     As System.EventArgs) Handles Button1.Click
        TextBox3.Text = updateSum(TextBox1.Text)
    End Sub

    Function updateSum(ByVal newsub As Decimal) As Decimal
        Static total As Decimal = 0
        total += newsub
        Return total
    End Function

End Class
```

Sub Programs

- مثال ذيل نمونه پروگرام فرعي Function را در Module را نشان ميدهد:

```
Module Module1
    Function addnumbers(ByVal x As Integer, ByVal y As
    Integer)
        Return x + y
    End Function
End Module

Public Class Form1
Private Sub Button1_Click(ByVal sender As System.Object, ByVal e
    As System.EventArgs) Handles Button1.Click
      TextBox1.Text = addnumbers(5, 8)
    End Sub
End Class
```

Sub Programs

- پروگرام فرعي Function يک **قيمت** را به پروگرام صدا کننده برگشت ميدهد. پروگرام ذيل استفاده از تابع را نشان ميدهد:

```
Public Class Form1
Private Sub Button1_Click(ByVal sender As System.Object,
    ByVal e As System.EventArgs) Handles Button1.Click
        MessageBox.Show(showresult(10, 20))
End Sub
Function showresult(ByVal hours As Integer, ByVal wage As
    Integer)
        Return hours * wage
End Function
End Class
```

Sub Programs

- **فرستادن پارامترها به قیمت (By Value) و ادرس (By Reference)**
- **پارامتر ها** به يكي از **دو روش** به **پروگرامهاي فرعي** فرستاده شده ميتواند. فرستادن **قيمت** پارامتر، و فرستادن **آدرس پارامتر**. اگر پارامترها به اساس **قيمت** فرستاده شود پروگرام يک **كاپي قيمت** آنرا به **پروگرام فرعي** صدا شده ميفرستد. در اين صورت **تغير** در **پروگرام صدا شده** اصل قيمت را تغير نميدهد. ولي اگر **پارامترها** به اساس **آدرس فرستاده** شود پروگرام فرعي صدا شده اصل قيمت را تغير ميدهد. چون پروگرام فرعي صدا شده به آدرس حافظه آن دسترسي دارد.

Sub Programs

مثال:

```
Public Class Form1
Private Sub Button1_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles
    Button1.Click
    Dim number1 As Integer = 2
    Label1.Text = "Passing a value-type argument by value:"
    TextBox1.Text = "Before calling SquareByValue, " & _
      "number1 is:  " & number1)
    SquareByValue(number1)
    TextBox2.Text = "After returning from SquareByValue, " & _
      "number1 is: " & vbCrLf & number1
    Dim number2 As Integer = 2
    Label2.Text = "Passing a value-type argument" & _
      " by reference:"
    TextBox3.Text = "Before calling SquareByReference, " & _
      "number2 is: " & number2
    SquareByReference(number2)
    TextBox4.Text =  "After returning from " & _
      "SquareByReference, number2 is: " & vbCrLf & number2
  End Sub
```

Sub Programs

```
Sub SquareByValue(ByVal number As Integer)
    TextBox1.Text =  "After entering SquareByValue, " & _
      "number is: " & number
    number *= number
    TextBox2.Text = "Before exiting SquareByValue, " & _
      "number is: " & number)
  End Sub
  Sub SquareByReference(ByRef number As Integer)
    TextBox1.Text =  "After entering SquareByReference" & _
      ", number is: " & number
    number *= number
    Console.WriteLine("Before exiting SquareByReference" & _
      ", number is: " & number)
  End Sub
End Class
```

Sub Programs

- **وقت متحول (Duration of Identifier)**
- **مدت زمان** كه يک **متحول** در **حافظه كمپيوتر** قرار دارد بنام **عمر** يا **وقت متحول** ياد ميشود. بعضي متحولين براي **مدت كوتاه** و برخي ديگر آن در طول اجراي پروگرام در حافظه موجود است. اين خاصيت را بنام **ساحه تعريف متحول** (Identifier Scope)نيزياد ميكند. ساحه **تعريف متحول** ميتواند **كلاس، ماديول، Namespace**، و يا **Block** باشد. پروگرام ذيل مسائل ساحه متحول را نشان ميدهد:

Sub Programs

```
Public Class FrmScoping
  Inherits System.Windows.Forms.Form
  Dim value As Integer = 1
  Private Sub FrmScoping_Load(ByVal sender As System.Object, _
    ByVal e As System.EventArgs) Handles MyBase.Load
    Dim value As Integer = 5
    lblOutput.Text = "local variable value in" & _
      " FrmScoping_Load is " & value
    MethodA()
    MethodB()
    MethodA()
    MethodB()
    lblOutput.Text &= vbCrLf & vbCrLf & "local variable " & _
      "value in FrmScoping_Load is " & value
  End Sub
```

Sub Programs

```
Sub MethodA()
    Dim value As Integer = 25
    lblOutput.Text &= vbCrLf & vbCrLf & "local variable " & _
      "value in MethodA is " & value & " after entering MethodA"
    value += 1
    lblOutput.Text &= vbCrLf & "local variable " & _
      "value in MethodA is " & value & " before exiting MethodA"
  End Sub
  Sub MethodB()
    lblOutput.Text &= vbCrLf & vbCrLf & "instance variable" & _
      " value is " & value & " after entering MethodB"
    value *= 10
    lblOutput.Text &= vbCrLf & "instance variable " & _
      "value is " & value & " before exiting MethodB"
  End Sub
End Class
```

Main Procedure

Sub Programs

- The **starting point** and **overall control** for your **application**.
- The .**NET Framework** calls your **Main procedure** when it has loaded your **application** and is ready to **pass control** to it.
- Unless you are creating a **Windows Forms application**, you must write the **Main procedure** for **applications** that **run** on **their own**.
- **Main contains** the code that **runs first**.
- In Main, you can determine which form is to be loaded first.

Sub Programs

```
Module Module1
    Sub main()
        Form2.show()
        MessageBox.Show("Hello")
    End Sub
End Module
```

- For this example to work:
  - Project, Properties.
  - In the Application Type, select Windows service or Console application.
  - In the Startup object, select Sub Main.

Sub Programs