Learning VB.Net

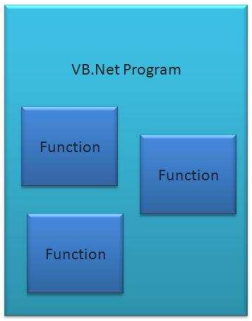
Tutorial 11 – Functions

Hello everyone… welcome to vb.net tutorials. These are going to be very basic tutorials about using the language to create simple applications, hope you enjoy it. If you have any notes about it, please send them to **notes@mka-soft.com** I will be happy to answer them. Finally if you find these tutorials are useful, it would be nice from you to send a small donation via PayPal to **donation@mka-soft.com**.

Tutorial posted on 2010-Feb-14.

Function

Up to this point, when you want to write an application you would simply write all your statements into one single block and try to solve all the problems in that simple block. However there is a better approach to write applications by dividing the application into a number of blocks that form together one application. These what functions are. Think about functions as small or mini-programs that each is designed to address a simple problem.



Functions make your applications easier to write and easier to understand. Now let us consider this example: you need to find the factorial for three variables, A,B and C. (The factorial of 4 is 4x3x2x1, and factorial of 7 is 7x6x5x4x3x2x1).

Usually to solve this issue you write something like:

```
A = 10
B = 5
C = 4

FA = 1
For I = 1 To A
    FA = FA * I
Next

FB = 1
For I = 1 To B
    FB = FB * I
Next

FC = 1
For I = 1 To C
    FC = FC * I
Next
```

So if you have for example 30 variables, you might need to rewrite the code 30 times (assuming you are not using arrays). However if you think about this in a different way, that is: **Could I have a statement that get me the factorial?** Then the code would be something like:

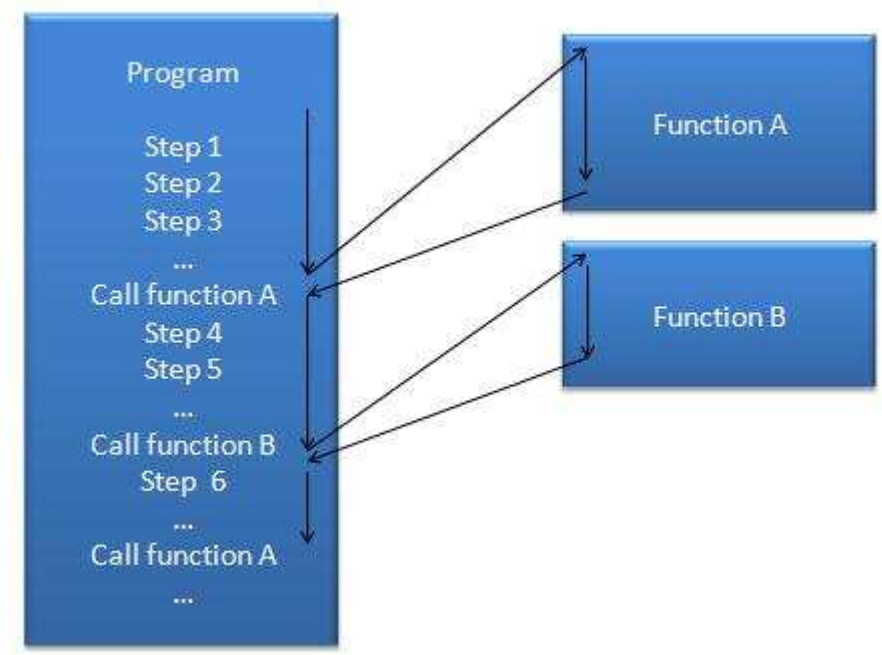
```
A = 10
B = 5
C = 4

FA = Factorial(A)
FB = Factorial(B)
FC = Factorial(C)
```

Which is very easy to write and understand. All you need is to tell the computer what Factorial really is:

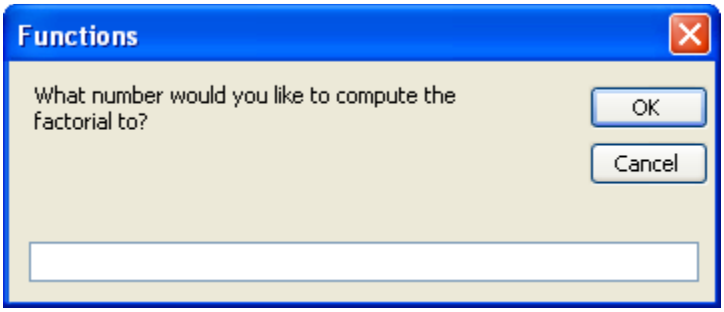
```
Function Factorial(ByVal N As Integer) As Double
    Dim F As Double         ' the factorial total
    Dim I As Integer        ' the counter
    F = 1                   ' the initial value of F
    For I = 1 To N          ' this loop to calculate the factorial
        F = F * I
    Next
    Return F                ' return the result
End Function
```

Now no matter how many times you need to calculate the factorial, you can just call it whenever you need. You don't have to worry about the loop or initializing F or anything else. You do it only once and you can use it anywhere.



Consider the above figure. It shows how the program behaves when it encounters a function call. The program executes normally, until it reaches a function call. At that time it will transfer its execution to function A. It will execute the function, get the result, and return back to the main program. The execution continues from the function call, and again, when there is another function call the process in repeated.

The functions you use could be built in functions, which means they are already written and available in the compiler, or user defined functions which are the ones you write. Built in functions are used to do general operations needed in most programs, and to make life easier for programmers. Examples of these are the **InputBox** function which shows a small window to read values from the display.



Other examples are math functions like **`Math.Pow`** - which finds the power of a number - and **`Math.Abs`** - which finds the absolute value of a number - and string functions like UCase and LCase.

As for user defined functions, you can define a function as follows:

```
' the factorial function
Function Factorial(ByVal N As Integer) As Double
    Dim F As Double         ' the factorial total
    Dim I As Integer        ' the counter
    F = 1                   ' the initial value of F

    For I = 1 To N          ' this loop is used to calculate the factorial
        F = F * I
    Next

    Return F                ' return the result
End Function
```

`Function`, `End Function` parts define where the code of the function starts and when it ends.

`Factorial` is the name of the function. You can choose any name however you might want to choose a meaningful name to remind you what this function does.

`ByVal N As Integer` is the parameter of the function. For now ignore the ByVal part. The parameter is a value passed from your program to the function so that the function could work on it. You define all the parameters between the brackets.

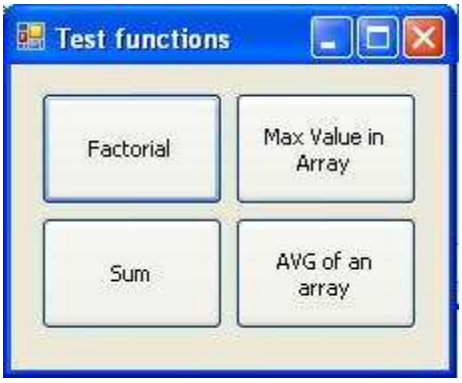
`As Double` which comes after the brackets tells the compiler what data type this function returns. When a function finish its processing it should return the result in the specified data type.

`Return` is used to tell the compiler what is the result of executing this function.

The following is an example of a function:

```
' the sum function
Function Sum(ByVal V1 As Integer, ByVal V2 As Integer, ByVal V3 As Integer) As Integer
    Return V1 + V2 + V3 ' calculate the sum and return the result in one single step
End Function
```

Now let us check the application included with the tutorial and see how it works:



It allows you to compute the factorial of a number, find the sum of 3 numbers, read and find max value or average of an array.

The factorial function is:

```
' the factorial function
Function Factorial(ByVal N As Integer) As Double
    Dim F As Double          ' the factorial total
    Dim I As Integer         ' the counter
    F = 1                    ' the initial value of F

    For I = 1 To N           ' this loop is used to calculate the factorial
        F = F * I
    Next

    Return F                 ' return the result
End Function
```

and the code of the factorial button is:

```
' read the number from screen
Dim MyNumb As Integer
Dim R As Double
MyNumb = InputBox("What number would you like to compute the factorial to?")
R = Factorial(MyNumb)
MsgBox("the result is:" & R)
```

Check out how to call the function, you use its name and pass the parameter value. You can even call the function in the following ways:

```
R = Factorial(MyNumb + 2)
R = Factorial(6)
R = Factorial(77/2)
R = Factorial(A - B)
```

and the value of the expression between the brackets is evaluated and passed to the function and placed in the variable N.

The rest of the code is straight forward. However I would like to highlight the functions that process functions:

```
' the max value in an array
Function GetMax(ByVal A() As Integer) As Integer
    Dim I As Integer
    Dim Max As Integer

    Max = A(0)                      ' assume the max value is the first value

    For I = 1 To A.Length - 1       ' loop over all other values in an array
        If Max < A(I) Then          ' if we find another value larger than max then
            Max = A(I)              '    correct max
        End If
    Next
    Return Max                      ' return the result
End Function

' the avg value of an array
Function GetAVG(ByVal A() As Integer) As Integer
    Dim I As Integer
    Dim sum As Integer

    sum = 0                         ' assume the sum value is 0

    For I = 0 To A.Length - 1       ' loop over all other values in an array
        sum = sum + A(I)
    Next

    Return sum / A.Length           ' return the result
End Function

' read array
Function ReadArray() As Integer()
    ' the counter
    Dim I As Integer

    ' the number of elements
    Dim N As Integer

    ' read the number of elements from the display
    N = InputBox("how many elements in the array")

    ' create the array
    Dim A(0 To N - 1) As Integer

    ' read the elements of the array
    For I = 0 To N - 1
        A(I) = InputBox("enter the element:" & I.ToString)
    Next

    ' return the result
    Return A
End Function
```

When you pass array to a function you use brackets () to tell the function that it is going to receive the array. In ReadArray() When the function returns an array, you add brackets after the return data type.

So this is a simple introduction to functions. If you find the code difficult or unclear leave me a comment, or send email to notes@mka-soft.com.

Next tutorial we discuss functions more. If you have any notes or questions send them to notes@mka-soft.com.

Thank you.

mkaatr.