Visual Web Development

Terry Marris January 2009

13 Functions

We have looked at sequences, selections, iterations and testing. Now we look at functions and scope.

13.1 Functions

Factorials are used in probability and statistical calculations. Factorial 5, written as 5!, means

5 x 4 x 3 x 2 x 1 = 120

A function usually performs one small task but always provides a single result. We shall write a function that returns the factorial of an integer - if it can.

A function is contained within the keywords Function and End Function.

```
Function
...
End Function
```

A function has a name, a parameter list and a return type.

```
Function factorial(ByVal n As Integer) As Integer
```

Here, the function's name is factorial, its parameter list is (ByVal n As Integer) and its return type is also Integer.

A function's name is usually chosen to reflect the kind of result being returned. Here, it is the factorial of a number n that we are calculating and so the function's name is factorial.

A function may have zero, one or many parameters. In our example there is just one parameter, ByVal n As Integer. n acts as a local variable.

A function must return a result. The result is the function's value.

```
Return nFactorial
```

Return is a VB word. There must be at least one return statement that is not subject to an If statement.

Here is the entire factorial function.

```
Function factorial(ByVal n As Integer) As Integer
  Try
    Dim i As Integer
    Dim factor As Integer = 1
    Dim nFactorial As Integer = 0

    For i = 1 To n
      factor = i * factor
    Next

    nFactorial = factor

    If n >= 0 Then
      Return nFactorial
    End If

    lblError.Text = "factorial: argument cannot be negative"
    Return -1
    Catch ex As OverflowException
      lblError.Text = "factorial: overflow"
    End Try
End Function
```

Declare variables

Calculate factorial

Return the result

Deal with errors

13.2 Function Calls

To call a function, to say "Hey, function, do your job!" we provide the function name along with any argument values it requires, and a variable to receive the value it returns.

```
txtOutput.Text = factorial(n)
```

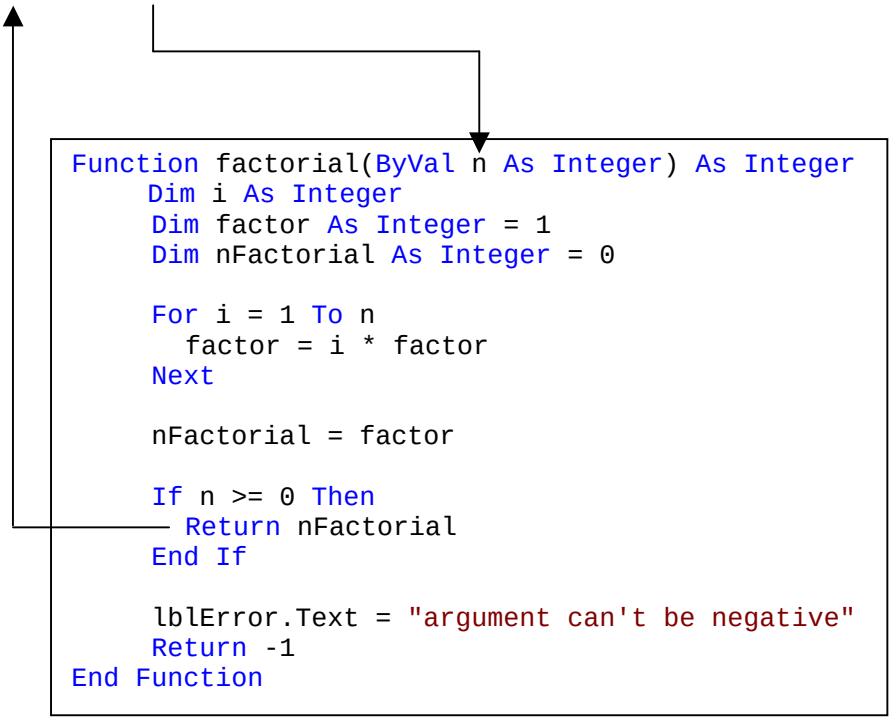
Here, n is the argument value provided by the user.

```
Dim n As Integer = Convert.ToInt32(txtInput.Text)
```

Argument-parameter pairs are the mechanism by which functions receive data to act upon.

When a function's Return statement is executed control returns immediately to the statement where the function was called.

```
txtOutput.Text = factorial(5)
```



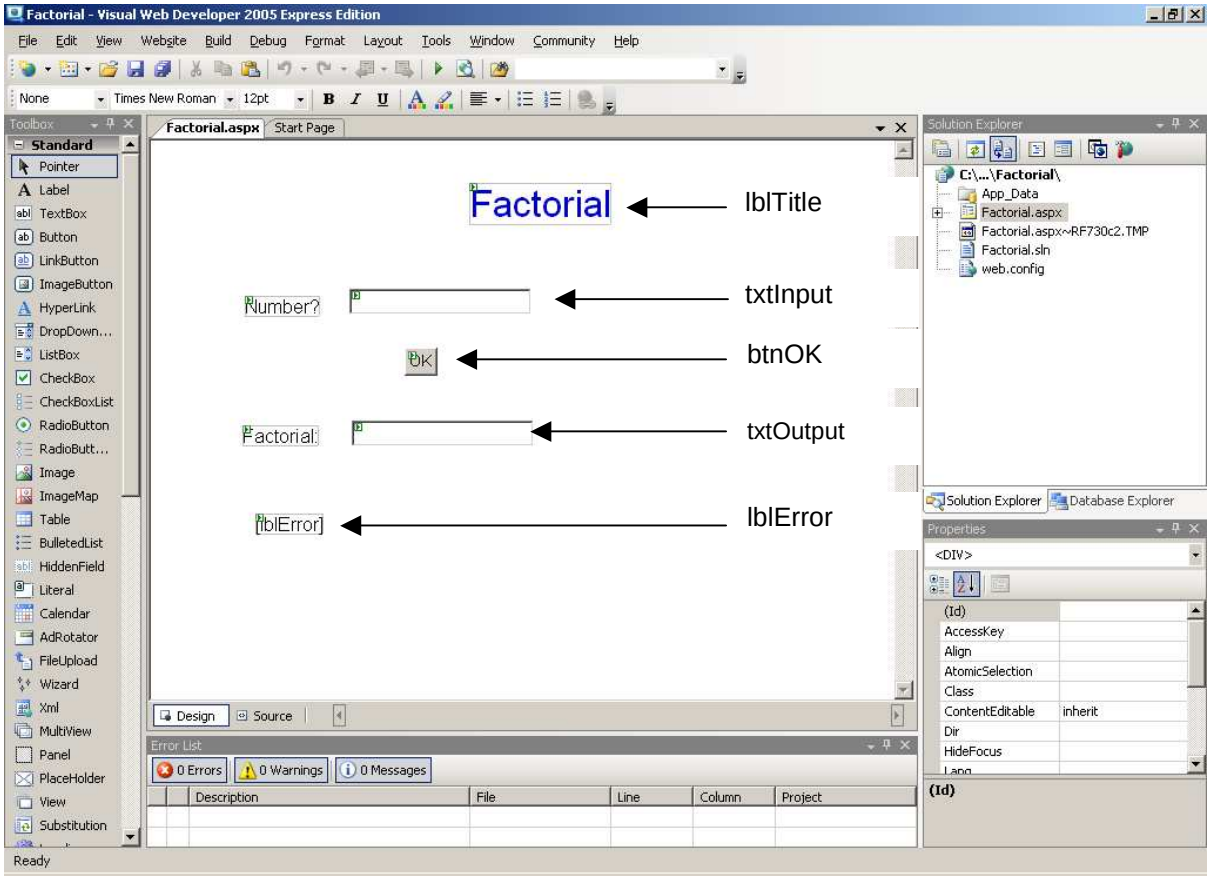
```
Function factorial(ByVal n As Integer) As Integer
    Dim i As Integer
    Dim factor As Integer = 1
    Dim nFactorial As Integer = 0

    For i = 1 To n
      factor = i * factor
    Next

    nFactorial = factor

    If n >= 0 Then
      Return nFactorial
    End If

    lblError.Text = "argument can't be negative"
    Return -1
End Function
```

In the example shown above, the argument, 5, is passed to the parameter, n. The function works out the factorial of 5. (5 x 4 x 3 x 2 x 1 = 180), and returns 180 in the function name, which is then assigned to txtOutput.Text.

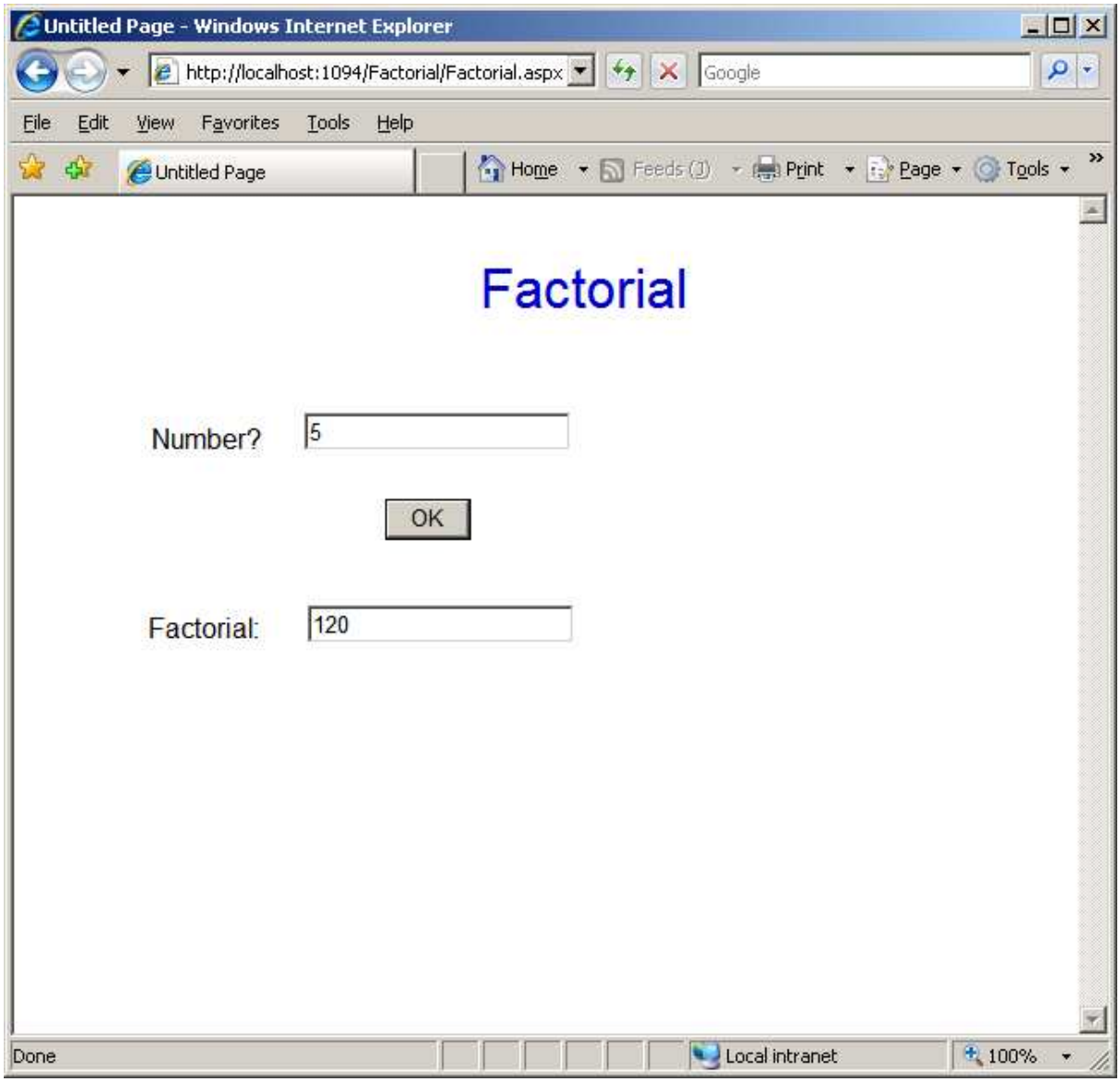
The value returned by a function may either be used by its caller, or ignored.

Here is the entire btnOK_Click procedure that obtains the input from the user, calls the factorial function and then displays the result.

```
Protected Sub btnOK_Click(ByVal sender As Object, ByVal e As
System.EventArgs) Handles btnOK.Click
Try
   Dim n As Integer = Convert.ToInt32(txtInput.Text)
  txtOutput.Text = factorial(n).ToString()
  Catch ex As FormatException
     lblError.Text = "Number format error"
  Catch ex As OverflowException
     lblError.Text = "Number input too large"
  End Try
End Sub
```



Form Layout


Untitled Page - Windows Internet Explorer
http://localhost:1094/Factorial/Factorial.aspx
Google
File Edit View Favorites Tools Help
Untitled Page
Home Feeds (J) Print Page Tools
Factorial
Number?
5
OK
Factorial:
120
Done
Local intranet
100%

Program Run

13.3 Scope

The variables defined in the factorial() function

```
Function factorial(ByVal n As Integer) As Integer
     Dim i As Integer
     Dim factor As Integer = 1
     Dim nFactorial As Integer = 0
     ...
End Function
```

are private or local to the function itself. Anything outside the function cannot directly access these variables. We say that the scope of these variables is the function. This means we can have two different variables, in two different functions, with the same name and without conflict.

```
Function f1() As Integer
     Dim i As Integer
     ...
     return i
End Function


Function f2() As Integer
     Dim i As Integer
     ...
     return i
End Function
```

The variable named i in f1() has no affect on the variable named i in f2(). And conversely.

The scope of global variables is the entire file in which they are defined.

```
Partial Class _Default
    Inherits System.Web.UI.Page
    ' GLOBAL VARIABLES
    Dim intCounter As Integer = 0
```

Any function within the file may update and access the values stored in such variables.

The scope of a variable is that part of the program that can access and update the variable.

13.4 Exercises

1. Try out a program that has a function to calculate the factorial of a none-negative integer as shown above.

2. Write and test a function that will return the sum of two integers

3. Write and test a function that will sum all the integers between zero and any integer.

13.5 Conclusion

We have looked at the fundamental building block of programs: functions. Next, we look at procedures.