Software Design and Development

7 Functions

Go there. Do it. Then come back with a result.

Terry Marris July 2009

Previously we looked at selections and repetitions. We continue our study of program constructs by looking at functions.

7.1 Functions

As our programs get larger and more complicated, we need strategies for managing size and complexity. We need functions.

A function is a piece of programming code. It has a name. It does just one small job. It delivers just one result. It can be used time and time and time again. Here is a simple example.

```
Function Add() As Integer
    Dim intSum As Integer = 2 + 3
    Return intSum
End Function
```

A function begins with the VB keyword *Function*.

Then comes its descriptive name: *Add* in our example.

Then comes a pair of brackets. Our brackets have nothing in them. But we shall soon fix that.

Then comes *As Integer*. This is the data type of the value being returned. It is the return type for the function.

Then comes a line of VB code that does the function's job: adds two numbers.

Dim intSum As Integer = 2 + 3

Then comes the *Return* statement. Every function must have at least one *Return* statement. This is the value the function gives to its caller. Its caller?

The caller is a line of VB code that calls upon the function to do its job. It says to the function: "Hey. Do your job. Give me the result."

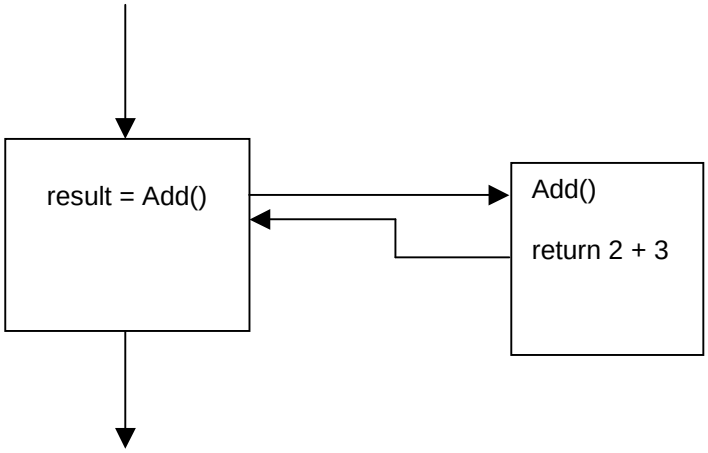
Dim intResult As Integer = Add()

Here, *intResult* receives the value returned by the function named Add().

It is a bit like having a partner make your tea in the morning. You say: "Hey, partner, make tea". Your partner goes and makes the tea and then returns with a hot steaming sweet cup of chai, which you gratefully receive and slurp.

Here, *intResult = Add()* says: Hey, *Add()*, do your job and give *intResult* the result. *Add()* goes and does its job and then returns with the result of two numbers added together, which it gives to *intResult* for it to do with as it wishes.

For those who prefer program flowcharts:



For those who prefer structured English:

```
Add()
  return 2 + 3
EndAdd

....
result = Add()
...
```

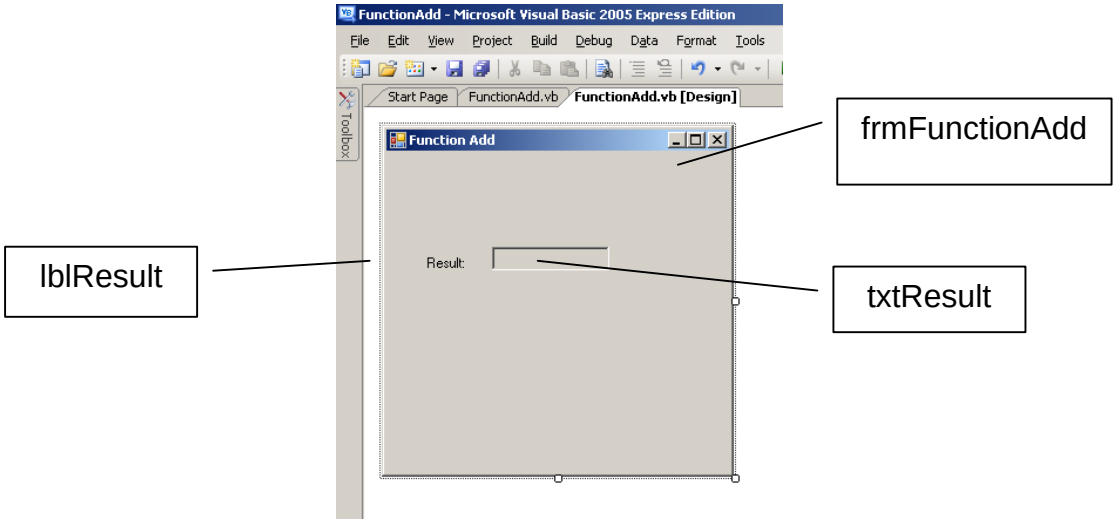
We first write the function. Then, when we use the function, we underline it to show that it is a function call we are making. As a general rule, we write it before we use it.

User Interface

1. Start a new project and name it *FunctionAdd*

2. Name the VB file *FunctionAdd.vb*

3. Set
   - **a.** Form
     - **i.** Name = *frmFunctionAdd*
     - **ii.** Text = *Function Add*
   - **b.** Label
     - **i.** Name = *lblResult*
     - **ii.** Text = *Result:*
   - **c.** TextBox
     - **i.** Name = *txtResult*
     - **ii.** ReadOnly = *True*



Programmer's Code

4. Rapid double click on a blank part of the form - this sets up a Sub routine named *frmFunctionAdd_Load(...)*

5. Above *Private Sub frmFunctionAdd_Load* write

```
Function Add() As Integer
    Dim intSum As Integer = 2 + 3
    Return intSum
End Function
```

6. Between *Private Sub frmFunctionAdd_Load(...)* and *End Sub* write

Dim intResult As Integer = Add()
txtResult.Text = intResult.ToString()

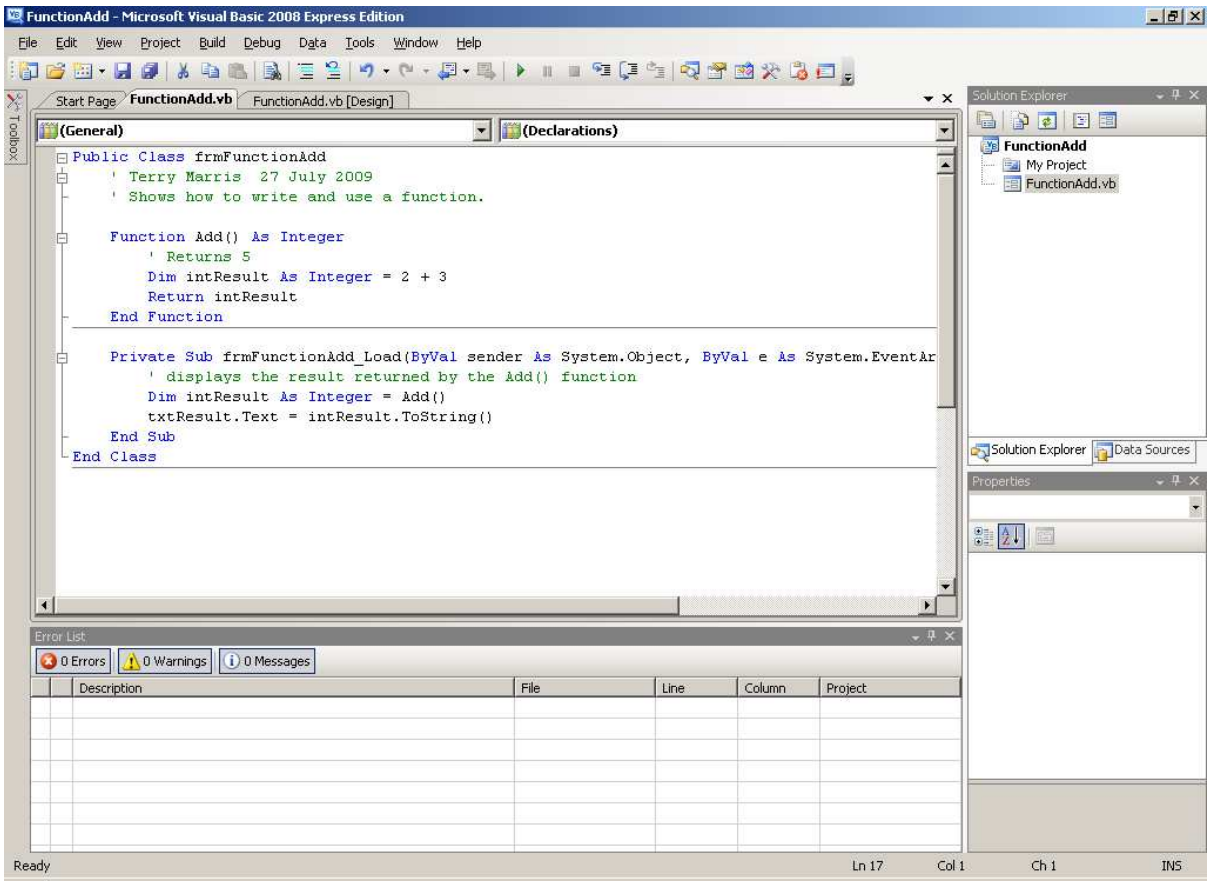
Your program should look something like this:

```
Public Class frmFunctionAdd
    ' Terry Marris  27 July 2009
    ' Shows how to write and use a function.

    Function Add() As Integer
        ' Returns 5
        Dim intResult As Integer = 2 + 3
        Return intResult
    End Function

    Private Sub frmFunctionAdd_Load(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles MyBase.Load
        ' displays the result returned by the Add() function
      Dim intResult As Integer = Add()
      txtResult.Text = intResult.ToString()

    End Sub
End Class
```



Exercise 7.1

1. Try out the program, shown above, that uses a function to display the result of adding 2 and 3. Remember to include appropriate comments.

2. Modify the program so that it displays the result of adding 5 and 2

9.2 Parameters and Arguments

Look at this function:

```
Function Sum(ByVal x As Integer, ByVal y As Integer) As Integer
     Dim intResult As Integer = x + y
     Return intResult
End Function
```

Its name is *Sum()* and it returns an *Integer*.

It has two Integers, named x and y, inside the brackets. These act like variables for the function. Whatever values they may have, they are added together and their sum is returned.

How do *x* and *y* get their values? They get their values from function calls. Here are some examples:

1. *Sum(2, 3)* gives 2 to *x* and 3 to *y*.

2. *Dim a As Integer = 3*
   Dim b As Integer = 4
   Sum(a, b) gives a copy of whatever is stored in *a* to *x*, and a copy of whatever is stored in *b* to *y*. So *x* receives 3 and *y* receives 4.

3. *intAnswer = Sum(intFirstNumber, intSecondNumber)*
   gives a copy of whatever is stored in *intFirstNumber* to *x*, and a copy of whatever is stored in *intSecondNumber* to *y*.

x and *y* in the function heading itself are known as *parameters*.

The 2 and 3 in the function call *Sum*(2, 3) are known as *arguments*.

The *a*, *b* in *Sum(a, b)* and the *intFirstNumber*, *intSecondNumber* in the function call *Sum(intFirstNumber, intSecondNumber)* are also examples of arguments.

User Interface

1. Start a new project and name it *FunctionSum*

2. Name the vb file *FunctionSum.vb*

3. Set
   a. Form
      i. Name = *frmFunctionSum*
      ii. Text = *Function Sum*

   b. Label
      i. Name = *lblFirstNumber*
      ii. Text = *First Number?*

   c. TextBox
      i. Name = *txtFirstNumber*

   d. Label
      i. Name = *lblSecondNumber*
      ii. Text = *Second Number?*

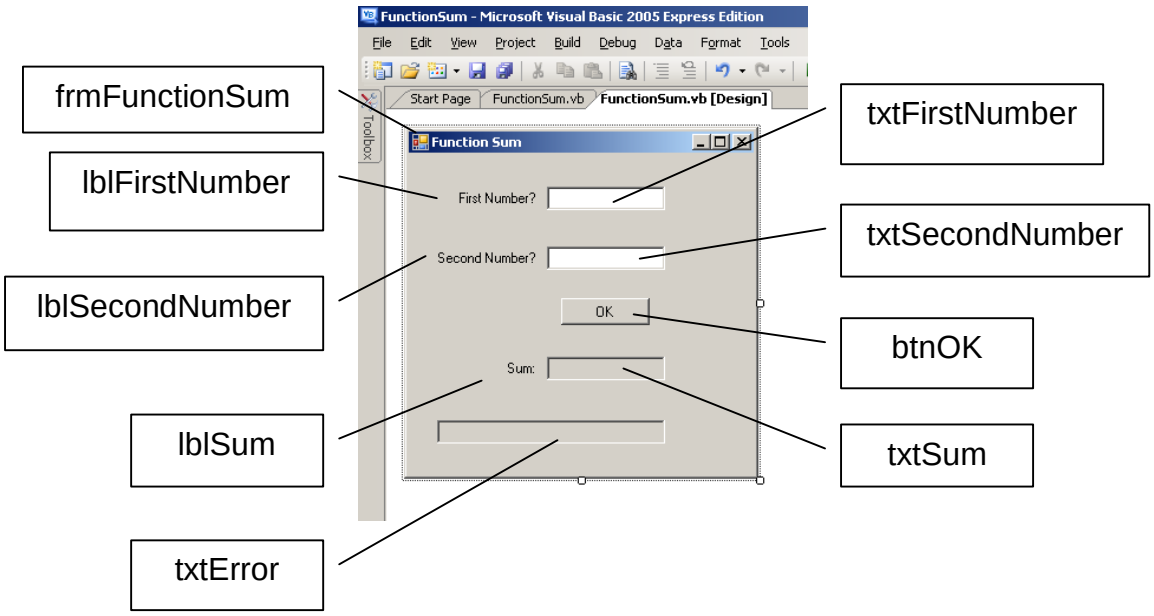
   e. TextBox
      i. Name = *txtSecondNumber*

   f. Button
      i. Name = *btnOK*
      ii. Text = *OK*

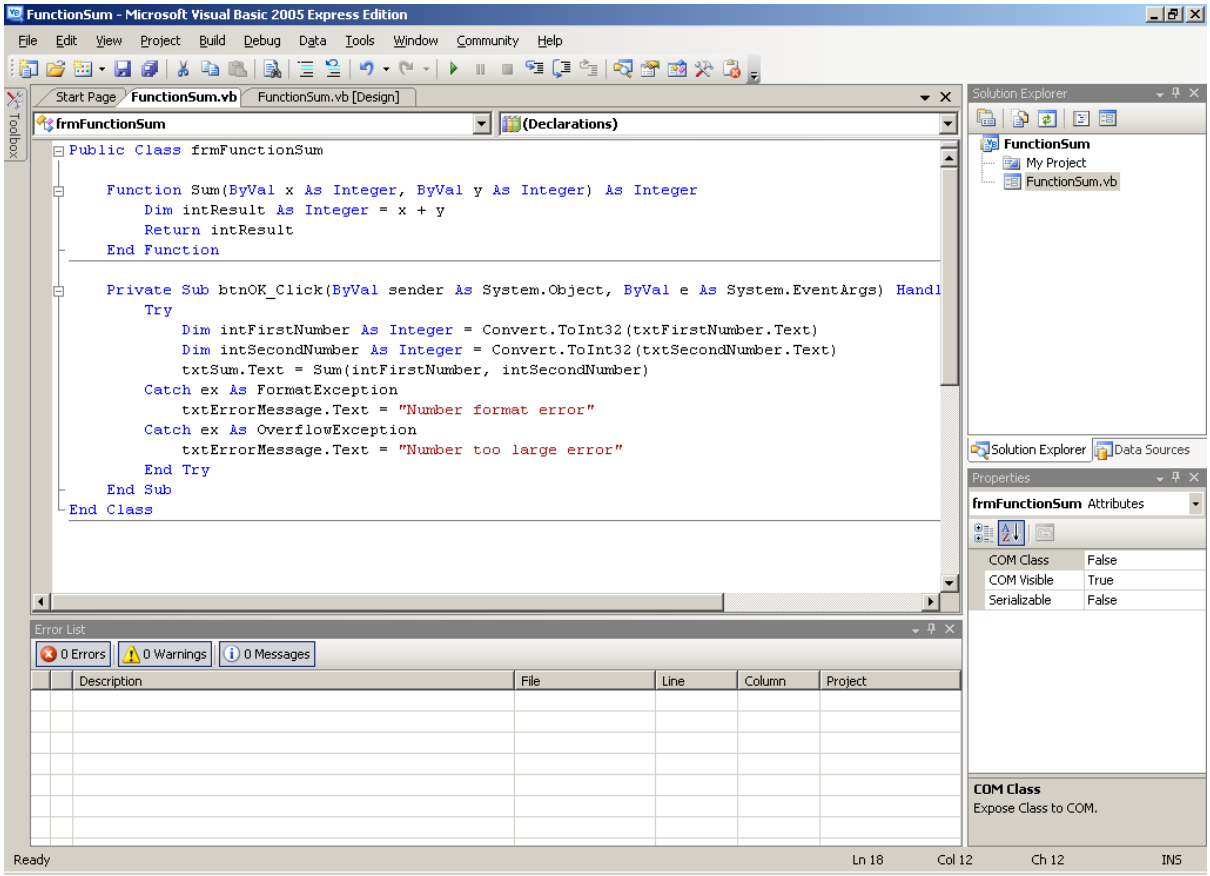
   g. Label
      i. Name = *lblSum*
      ii. Text = *Sum:*

   h. TextBox
      i. Name = *txtSum*
      ii. ReadOnly = *True*

   i. TextBox
      i. Name = *txtError*
      ii. ReadOnly = *True*



Programming Code



4. Rapid double click on the OK button to open a VB code window.

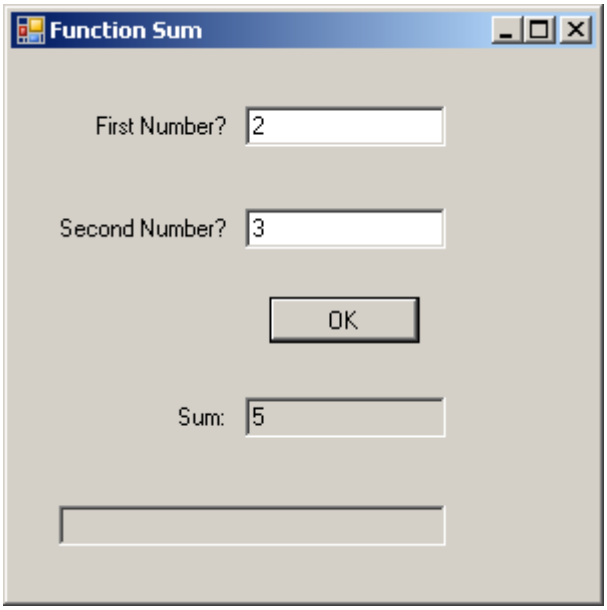
5. Enter the function under the Public class frmFunctionSum (as shown above)

```
Function Sum(ByVal x As Integer, ByVal y As Integer) As Integer
    Dim intResult As Integer = x + y
    Return intResult
End Function
```

6. Enter the VB code shown below under the OK button

```
Try
     Dim intFirstNumber As Integer =
          Convert.ToInt32(txtFirstNumber.Text)
     Dim intSecondNumber As Integer =
          Convert.ToInt32(txtSecondNumber.Text)
     txtSum.Text = Sum(intFirstNumber, intSecondNumber).ToString()
Catch ex As FormatException
     txtErrorMessage.Text = "Number format error"
Catch ex As OverflowException
     txtErrorMessage.Text = "Number too large error"
End Try
```

An example of a program run is:



Exercise 7.2

1. Try out the program, shown above, that uses a function to add 2 numbers. Remember to include a comment describing what the function does.

2. Design, write and test a program that uses a function to multiply two integers

7.3 KISSS

Keep it small, sweet and simple.

We can do anything we like with the value returned by a function, even ignore it if we want to.

Programmers love writing small, simple functions because they are so easy to design, write and get right.

Exercise 7.3

1. Design, write and test programs that will:

    a. input the dimensions of a rectangular window space, use functions to calculate its area and its perimeter, and output the area and perimeter

    b. input the diameter of a circular pond, use functions to calculate its circumference and surface area, and output the circumference and surface area

    c. input two numbers, use a function to return the larger, and output the larger

    d. input a bill, use a function to add VAT at 17.5% to the bill, and output the bill plus VAT

    e. input an exam mark, use a function to return *unsatisfactory* if the mark is less than 40, or return *satisfactory* if the mark is more than 40, and output the grade

    f. input a marathon runner's age, use a function to return junior if their age is less than 17, regular if their age is between 17 and 39 inclusive, and senior if their age is 40 or more, and output their category

    g. input room capacity and class size, use a function to return whether the room capacity is big enough for the class, and output whether the room is, or is not, suitable

h. input the details of two people - their ages and whether they like computer games, use a function to determine whether their ages are within 5 years of each other and they have the same interest, and output either *match* or *no match*

i. input whether or not a suspect to a crime has a motive and an alibi, use a function to determine whether the suspect is worth investigating further (has a motive and has no alibi) and output *worth investigating* or *not worth investigating*

Bibliography

Deitel, Deitel & Yaeger *Simply visual Basic.NET 2003* Pearson 2004 pp 276

Next we look at arrays. The programmer said to the boss: *I want arrays*. The boss said to the programmer: *Look at the next handout*.