

TegoSoft Inc.

Copyright © 1994-1996 TegoSoft Inc. ® All Rights Reserved

P.O.Box 389, Bellmore, NY 11710

Web Site: <http://www.tegosoft.com>

 e-mail: tegosoft@msn.com

Lesson 21 - Functions


In this lesson you'll learn about functions in Visual Basic.


Creating the Directory of the Project and Saving the Files of the Project


As usual, you'll start by creating the directory of the project, and saving the files of the project.

 Create the **C:\VBMyProg\Lesson21** directory.


You'll save the files of this lesson to this directory.

 Select **New Project** from the **File** menu of Visual Basic.

 Make the window of Form1 the selected window, select **Save File As** from the **File** menu of Visual Basic, and save the file as **MyFunct.FRM** inside the **C:\VBMyProg\Lesson21** directory.

 Select **Save Project As** from the **File** menu of Visual Basic, and save the project as **MyFunct.VBP** inside the **C:\VBMyProg\Lesson21** directory.

Always Use Option Explicit

 Inside the general declarations section of Form1 type the following code:

```
' Force variable declarations
Option Explicit
```

The **Option Explicit** statement forces you to declare variables before using the variables.

Functions


You already used functions in your Visual Basic programs. For example, the **Int()** function converts the parameter that you supplied to it to an integer. For example, the following statement sets the value of **MyInteger** to **3**:

```
MyInteger = Int(3.6)
```

The **Int()** function "works" on the parameter that supplied to it, and then the **Int()** function returns a value. The returned value is assigned to the **MyInteger** variable.

The **Int()** function is an example of a function that is built into Visual Basic. Visual Basic include many functions which you can use.

Sometimes, you will want to write your own function. You'll now implement your own function. In the following section you are going to design Form1 as shown in Figure 21.2.

 Place a CommandButton inside Form1 and set its properties as follows:

Name: cmdExecuteFunction

=====

Caption: Execute Function

 Place a Label control inside Form1 and set its properties as follows:

Name: lblResult

Caption: make it empty

BorderStyle: 1-Fixed Single

Alignment: 2-Center

Your Form1 should now look as shown in Figure 21.1.

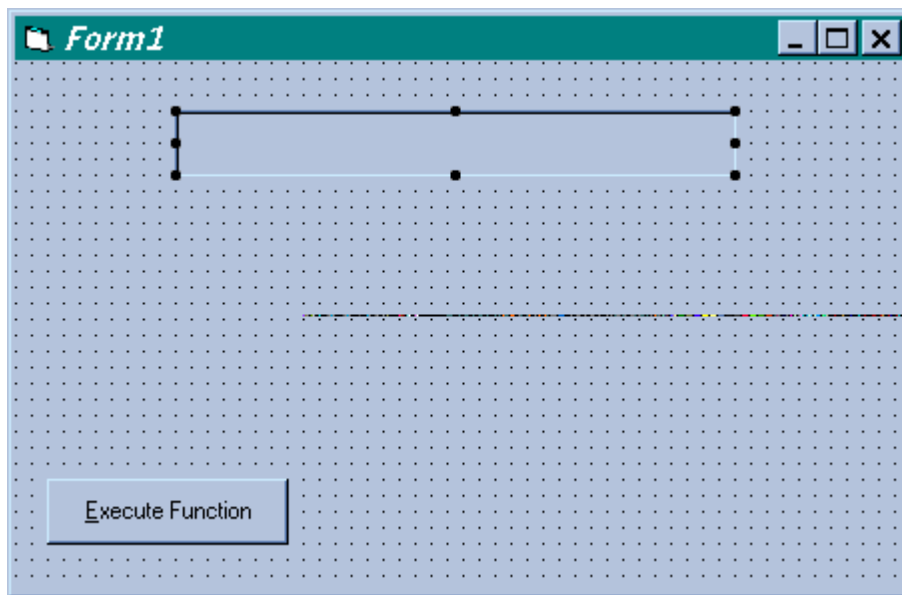


Figure 21.1. *Form in design time.*

Adding Numbers (Without a Function)

You'll now write code that adds two numbers. First, you'll accomplish the addition operation without the use of a function:

 Add code inside the **cmdExecuteFunction_Click()** procedure as follows:

```
Private Sub cmdExecuteFunction_Click()  
  
Dim A, B, C  
  
A = 10  
B = 20  
  
C = A + B  
  
lblResult.Caption = "  A=" + Str(A) + _  
                    "  B=" + Str(B) + _  
                    "  C=A+B=" + Str(C)  
  
End Sub
```

The code that you typed declares local variables:

```
Dim A, B, C
```

You assign values for the **A** and **B** variables:

```
A = 10  
B = 20
```

You add **A** to **B** and assign the result to the **C** variable:


```
C = A + B
```


Finally, you update the label with the value of **C**:

=====

```
lblResult.Caption = "  A=" + Str(A) + _  
                    "  B=" + Str(B) + _  
                    "  C=A+B=" + Str(C)
```

Let's see the code in action:


 Select **Save Project** from the **File** menu of Visual Basic to save your work. Then execute the MyFuncnt program.

 Click the **Execute Function** button, and verify that the label reports the result of adding the number **A** to **B**.

 Terminate the MyFuncnt program.

Creating Your Own Function

As stated, Visual Basic has many functions in it (e.g., **Int()** to convert numbers to integers, **Str()** to convert numbers to **strings**, **Rnd()** to generate random numbers). You'll now create your won function:

 Make sure that the Code window is the selected window and then select **Procedure** from the **Insert** menu of Visual Basic. (You must make the Code window the selected window, otherwise, the **Procedure** menu item of the **Insert** menu is dimmed).

Visual Basic responds by displaying the **Insert Procedure** dialog box (see Figure 21.2.).

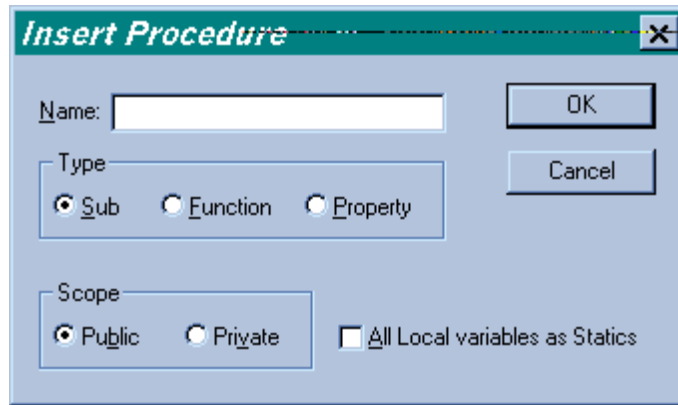







Figure 21.2. *The Insert Procedure dialog box.*

 Set the **Insert Procedure** dialog box as follows:

-  Inside the Name box type: **AddNumbers**
-  Select the **Function** radio button.
-  Make sure the **Public** radio button is selected.
-  Make sure that the **All Local variables as Statics** check box is **not** checked.

Your **Insert Procedure** should now look as shown in Figure 21.3.

 Click the OK button of the **Insert Procedure** dialog box.

Visual Basic responds by inserting the **AddNumbers** function to the general section of Form1 as follows:

```
Public Function AddNumbers()
```

```
End Function
```

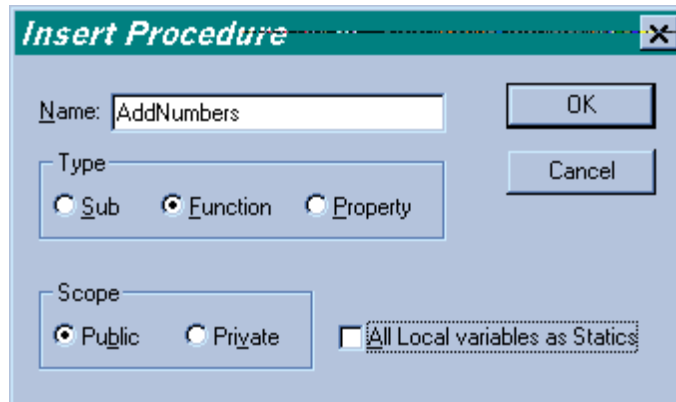


Figure 21.3. Setting the Insert Procedure dialog box.

Executing the AddNumbers() Function

Before writing the code of the **AddNumbers()** function, let's modify the **cmdExecuteFunction_Click()** procedure:

 Modify the code inside the **cmdExecuteFunction_Click()** procedure so that it will look as follows:

```
Private Sub cmdExecuteFunction_Click()
```

```
Dim A, B, C
```

```
A = 10
```

```
B = 20
```

```
C = AddNumbers(A, B)
```

```
lblResult.Caption = " A=" + Str(A) + _
```

```
                " B=" + Str(B) + _
```

```
                " C=A+B=" + Str(C)
```

```
End Sub
```

In the preceding code you assigned values for the **A** and **B** variables, and then you execute the **AddNumbers** function:

```
C = AddNumbers(A, B)
```

Note how the **AddNumbers()** function is executed. You assign the return value of the function to the **C** variable. You also specify two parameters, the first parameter is **A**, and the second parameter is **B**.


 **NOTE**

Note the difference between procedures and functions. A procedure does not return a value. A function returns a value, and you must assign the return value to a variable as in:

```
C = AddNumbers(A, B)
```

The Code of the AddNumbers() Function

You'll now write the code of the **AddNumbers()** function.

 Double click inside a free area of Form1 to display the Code window, set the **Object** list box to **(general)**, set the **Proc** list box to **AddNumbers**, and then type the following code inside the **AddNumbers** function:

```
Public Function AddNumbers(ByVal Num1, ByVal Num2)
```

```
Dim Result
```

```
Result = Num1 + Num2
```

```
AddNumbers = Result
```

End Function

Note that originally, the first line of the **AddNumbers()** function was as follows:

```
Public Function AddNumbers()
```

You modified the first line of the **AddNumbers()** function so that it will look as follows:

```
Public Function AddNumbers(ByVal Num1, ByVal Num2)
```

You specified on the first line of the **AddNumbers()** function two parameters. The first parameter is **Num1** and the second parameter is **Num2**. You prefixed these parameters with the keywords **ByVal**. This means that the **AddNumbers()** function cannot modified the original **A** and **B** variables that were passed to it.

You then add the numbers:

```
Result = Num1 + Num2
```


And now to the most important statement:

```
AddNumbers = Result
```

At this point in the function, **Result** holds the number that represents the result of the addition operation. You must set the value of a variable called **AddNumbers** with the value that you want to return from the **AddNumbers** function. You do not have to declare the **AddNumbers** variable. This variable is automatically declared and the name of the variable is identical to the name of the function. Remember that you are now designing a function, and a function always returns a value. The mechanism of returning the value from the function is implemented by simply assigning the returned value to a variable that has the same name as the function.

Let's see you code in action:

```
=====
```

 Select **Save Project** from the **File** menu of Visual Basic to save your work. Then execute the MyFuncnt program.

 Click the **Execute Function** button.

The MyFuncnt program responds by reporting the result of the addition inside the label.

 Terminate the MyFuncnt program.

Passing Parameters By Reference

In the preceding, you designed the **AddNumbers()** function so that the parameters that are passed to the function are passed by value (**ByVal**) This means that the **AddNumbers()** function cannot change the values of the original parameters that were passed to it. In the Project section of this lesson you'll learn how to pass parameter by reference. When you pass parameters by reference, the function can modify the values of the original variables that were passed to it.

What You Accomplished in This Lesson



You completed Lesson 21 of the Self Study Visual Basic tutorial. In this lesson you learned about functions in Visual Basic. Functions are very similar to procedures. The difference is that a function returns a value.

Frequently Asked Questions



Q1. Should I use a procedure or a function?

A1. A procedure does not return a value. A function returns a value. If you need to perform a certain task and a returned value is needed, then use a function. Otherwise, use a procedure.

Exam



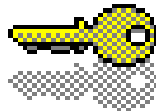
1. What is wrong with the following function?

```
Public MyFunction (ByVal I)
```

```
I = I * I
```

```
End Function
```

Answers to Exam



1. The name of the function is MyFunction. A function must return a value. The return value must be assigned to a variable that has the same name as the function. Thus, the **MyFunction()** function should be written as follows:

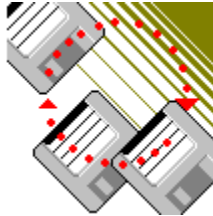
```
Public MyFunction (ByVal I)
```

```
I = I * I
```

```
MyFunction = I
```


```
End Function
```

Project



Modify the MyFuncnt program so that the **AddNumbers()** function will be able to modify the original parameters that were passed to it.

Let's do it in steps. First you'll prove to yourself that in the current state of the **AddNumbers()** function, the **AddNumbers()** function is not capable of modifying the original values of **A** and **B** that were passed to it:

 Add code inside the **cmdExecuteFunction_Click()** procedure so that the procedure will look as follows:

```
Private Sub cmdExecuteFunction_Click()
```

```
Dim A, B, C
```

```
A = 10
```

```
B = 20
```

```
C = AddNumbers(A, B)
```

```
lblResult.Caption = " A=" + Str(A) + _  
                    " B=" + Str(B) + _  
                    " C=A+B=" + Str(C)
```

```
Print A
```


```
End Sub
```

You added the statement:

```
Print A
```

So the **cmdExecuteFunction_Click()** procedure will display the value of **A**.


Let's see this in action:

 Execute the MyFuncnt program. Then click the **Execute Function** button.

As you can see, the number **10** (which is the value of **A**) is displayed every time you click the **Execute Function** button.

 Terminate the MyFuncnt program.

Now write code that modifies the **Num1** parameter:

 Modify the code inside the **AddNumbers()** function so that the **AddNumbers()** function will look as follows:

```
Public Function AddNumbers(ByVal Num1, ByVal Num2)
```

=====

```
Dim Result
```

```
Result = Num1 + Num2
```


```
AddNumbers = Result
```

```
Num1 = 123
```

```
End Function
```

You added the following statement:


```
Num1 = 123
```

 Execute the MyFunc program. Then click the **Execute Function** button.

As you can see, the number **10** (which is the value of **A**) is displayed every time you click the **Execute Function** button. In other words, the fact that you changed the value of **Num1** to **123** does not effect the original value of the **A** variable.

 Terminate the MyFunc program.

Now modify the code inside the **AddNumbers()** function so that parameters will be passed to the function by reference:

 Modify the code inside the **AddNumbers()** function so that the **AddNumbers()** function will look as follows:

```
Public Function AddNumbers(ByRef Num1, ByRef Num2)
```

```
Dim Result
```

```
Result = Num1 + Num2
```

```
AddNumbers = Result
```


```
Num1 = 123
```

```
End Function
```

You changed the first line of the function as follows:

```
Public Function AddNumbers(ByRef Num1, ByRef Num2)
```

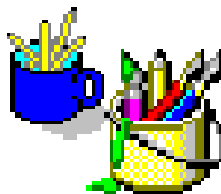
So parameters are now passed to the function by reference, not by value.

 Execute the MyFunc program. Then click the **Execute Function** button.

As you can see, the number **123** (which is the new value of **A**) is displayed every time you click the **Execute Function** button. In other words, the fact that you changed the value of **Num1** to 123 effects the original value of the **A** variable.

 Terminate the MyFunc program.

Cosmetic Considerations



Again, note the danger when passing parameters by reference. Here you are inside the **cmdExecuteFunction_Click()** procedure thinking that **A** is equal to **10**. Yet, because you executed the **AddNumbers()** function and passed the parameters to the function by reference, the **AddNumbers()** function was able to change the value of **A**.

If you ever execute a function that has parameters which are passed to it by reference, add a comment to remind you that the function can modify the parameters:

```
' AddNumbers() can modify the A variable!  
C = AddNumbers(A, B)
```

How To Contact TegoSoft



You can contact TegoSoft Inc. by any one of the following methods:

- Use TegoSoft Internet Web site:

<http://www.tegosoft.com>

- Send TegoSoft an e-mail:

tegosoft@msn.com

- Send TegoSoft a letter:

TegoSoft Inc.

P.O.Box 389

Bellmore, NY 11710


USA

Technical Support



If you have a technical question, you can post the question to the TegoSoft Technical Support staff, and they will try to answer your question.

The **best** way to post a technical question is by sending TegoSoft an e-mail.

 The e-mail of TegoSoft is:
tegosoft@msn.com

When sending TegoSoft an e-mail with a technical question, please follow the following format:

Date: _____
Your name: _____
Company (if applicable): _____
Your phone number: _____
Your e-mail: _____
Country (if not USA): _____
State (if inside USA): _____

Operating System used: _____
Programming language and version : _____

My technical question is:

Copyright © and Notices

Copyright © 1994, 1995, 1996 by TegoSoft Inc. ® All Rights Reserved

TegoSoft Self Study Tutorials & Software

=====

TegoSoft Visual Basic 4 Self Study Tutorial - Lesson 21

Page 17 of 18

Copyright ©1994, 1995, 1996 by TegoSoft Inc. ® All Rights Reserved

Although precaution was taken in the preparation of this document, TegoSoft assumes no responsibility for errors or omissions. TegoSoft is not liable for damages resulting from the use of the information contained in this document. Use this document at your own risk. This document is copyright protected. This means that you should treat this document like any other copyright material. No part of this document should be copied in any way. You are not allowed to use this document or any part of this document for any purpose other than read it. You are not allowed to publish this document or any part of this document in any way. You are not allowed to sell this document or any part of this document, you are not allowed to incorporate this document or any part of this document in any book, magazine, Web Site, Electronic forums, disks, CDs, or any other media. The only thing that you are allowed to do with this document is read it for the sole purpose of studying the material that is presented in this document. If this document includes software, the software must be treated in the exact same way that this document is treated. You are not allowed to distribute the software in any way. The sole purpose of supplying the accompanying software is to enable you to experience with it. No part of this document should be modified. If accompanying software is included with this document, you are not allowed to modify the software.

Rev. 031796-1