LAB GOALS

To create user defined procedures. To understand parameter passing.

1) Passing Parameters by Value
2) Passing Parameters by Reference

Throughout this semester, we have been using many built-in procedures. For example Console.WriteLine() has been a built-in procedure that we have used for weeks. In fact the Main() procedure in a console application is another example of using a procedure. Also, when we build “Windows Applications”, we routinely create “Event Handlers” which are procedure that are called when a particular event (such as mouse click, etc.) occurs. In this lab we will extend our knowledge about procedures by creating a few user-defined procedures. We will also learn about passing parameters to our procedures.

Step 1: Create a new console project named AConsole_Application_Procedures. Make sure to save your new project in your A290 folder.

Step 2: Create the following shell structure for a console application. Then compile and run it to make sure it works ok.

Option Explicit On
Option Strict On

Module Module1
    Sub Main()
        Console.ReadLine() ' pause the program
    End Sub
End Module

Step 3: Add the following lines right after the “End Sub” line.

'Procedure without a Parameter
Private Sub DisplayMessage1()
    Console.WriteLine("Hello World")
End Sub

The above is a new user-defined procedure. When called upon, this procedure will simply execute its contents and returns the control back to the calling module.

Run: Compile and Run your program. What is the output? Why aren’t you getting any output?

Step 4: To execute the module built in Step 3, we must call it. In order to call this module, type the following line right before the Console.Readline() statement in the main() module.

DisplayMessage1()

Run: Compile and Run your program. What is the output? Are you getting any output? Try calling the same DisplayMessage1() procedure a second time, and run the program again. What do you see? Can you explain what is happening?
Step 5: Now let’s create a more useful build-in procedure, a procedure which accepts an argument/parameter, and uses its parameter in the body of the procedure. Type the following code before the line “End Module”.

'Procedure with one Parameter (Pass By Value)

Private Sub DisplayMessage2(ByVal Message As String)
    Console.WriteLine(Message)
End Sub

Run: Compile and Run your program. Note that by simply adding a procedure to our program, the procedure will not be automatically called. We must explicitly call the procedure.

Step 6: To call the module built in Step 5, we must type the following line right before the Console.ReadLine() statement in the main() module.

DisplayMessage2(“Hello”)

Step 7: Let’s make our procedure a little more interesting. Let’s have the procedure display its message 5 times:

Hello
Hello
Hello
Hello
Hello

Ok, now, let’s make it display “Hello” 5000 times. (Yes, you need to use a loop)

Step 8: Let’s put in a little more logic in our procedure. Let’s have the procedure display its message 5 times, if the content of the “Message” parameter is the word “Hello”, and display its message 10 times, if the content of the “Message” parameter is “Good Bye”. (Yes, you need to use both loops and conditionals)

Step 9: Passing parameters to procedures make them a lot more powerful. But we can do even more! In fact we can even receive information back from a procedure through the same parameters. This is known as passing parameters “By Reference”. Type the following code before the line “End Module”.

Private Sub GetUserName(ByRef strName As String)
    Console.Write(“Enter your name: “)
    strName = Console.ReadLine()
End Sub

Now add the following lines to the Main() program (right before the Console.ReadLine() statement) to call the newly created GetUserName() module:

Dim Name As String
Name = “xxx”

GetUserName(Name)
Console.WriteLine(“Name = {0}”, Name)

Run: Compile and Run your program. Note that by executing the program, the user name that was supplied inside the GetUserName() procedure is brought back to the main() module and printed. So, we are bring back information from the procedure, through the parameter “strName”. Try changing the “ByRef” to “ByVal”, and run the program again. What do you see?