Lab 4: Modules

This lab accompanies **Chapter 3** of *Starting Out with Programming Logic & Design.*

Lab 4.1 – Pseudocode and Modules

Critical Review

A Module is a group of statements that exists within a program for the purpose of performing a specific task.

Modules are commonly called procedures, subroutines, subprograms, methods, and functions.

The code for a module is known as a module definition. To execute the module, you write a statement that calls it.

The format for a module definition is as follows:

```
Module name( )
      Statement
      Statement
      Etc.
End Module
```

Calling a module is normally done from the Main ( ) module such as:

```
Call name( )
```

Generally, local variables should be used and arguments should be passed by reference when the value of the variable is changed in the module and needs to be retained. For example:

```
Module main( )
      Real Integer number
      Call inputData(number)
      Call printData(number)
End Module

//accepts number as a reference so the changed value
//will be retained
Module inputData(Real Ref number)
      Number = 20
End Module

//number does not to be sent as a reference because
//number is not going to be modified
Module printData(number)
      Display "The number is ", number
End Module
```

Demo Video: View *lab4-1.wmv* in the Lab 4 folder on the accompanying Lab Demo Media and Startup Files CD

This lab requires you to think about the steps that take place in a program by writing pseudocode. Read the following program prior to completing the lab.

```
Data Communications Corp wants a small program that will
calculate the cost of UTP it installs for their clients.
Write a program that will ask the user to input the name of
the client, the number of feet of cable installed.  The
program should then calculate and display a final bill.
Cost per foot of UTP is .21 cents.  Be sure to add on a tax
of 6%.  Final bill should include the total cost and client
name.  Be sure to add modules to your program.
```

Consider the following variables and modules in your program. (Reference: Defining and Calling a Module, page 78).

Variable Name
Declare String clientName
Declare Real feetUTP
Declare Real subTotal
Declare Real taxCost
Declare Real totalCost

Module Name
Module inputData ()
Module calcCosts()
Module displayBill()

Step 1: Complete the pseudocode by writing the missing lines. (Reference: Defining and Calling a Module, page 78-81). Also, when writing your modules and making calls, be sure to pass necessary variables as arguments and accept them as reference parameters if they need to be modified in the module. (Reference: Passing Arguments by Value and by Reference, page 97 – 103).

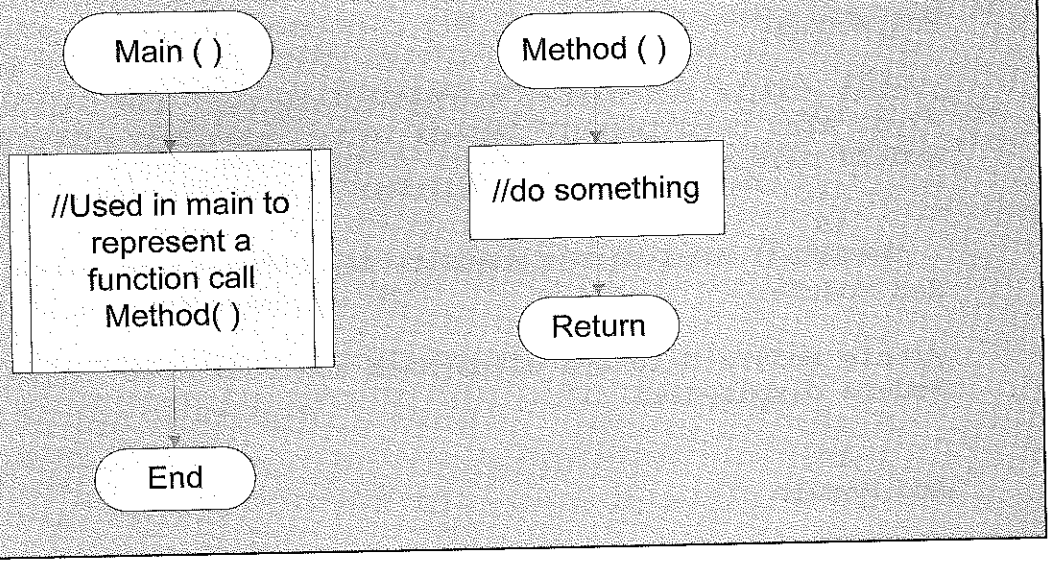
```
Module main ()
      //Declare local variables
      1.
      2.
      3.
      4.
      5.

      //Module calls
      6.
      7.
      8.
```

```
End Module

//this module takes in the required user input.  There will
//be a display and input for each variable
Module inputData(Real Ref feetUTP, String Ref clientName)
     9.
     10.
     11.
     12.
End Module

//this module calculates subTotal, taxCost, and totalCost
//you also need feetUTP passed in to calculate subTotal
Module calcCosts(13.   ,    ,    ,   )
     14.
     15.
     16.
End Module

//this module displays clientName and totalCost
Module displayBill (17.   ,   )
     18.
     19.
End Module
```

Lab 4.2– Flowcharts and Modules

Critical Review

The flowchart symbol used for a function call is a rectangle with vertical bars on each side:



Demo Video: View *lab4-2.wmv* in the Lab 4 folder on the accompanying Lab Demo Media and Startup Files CD

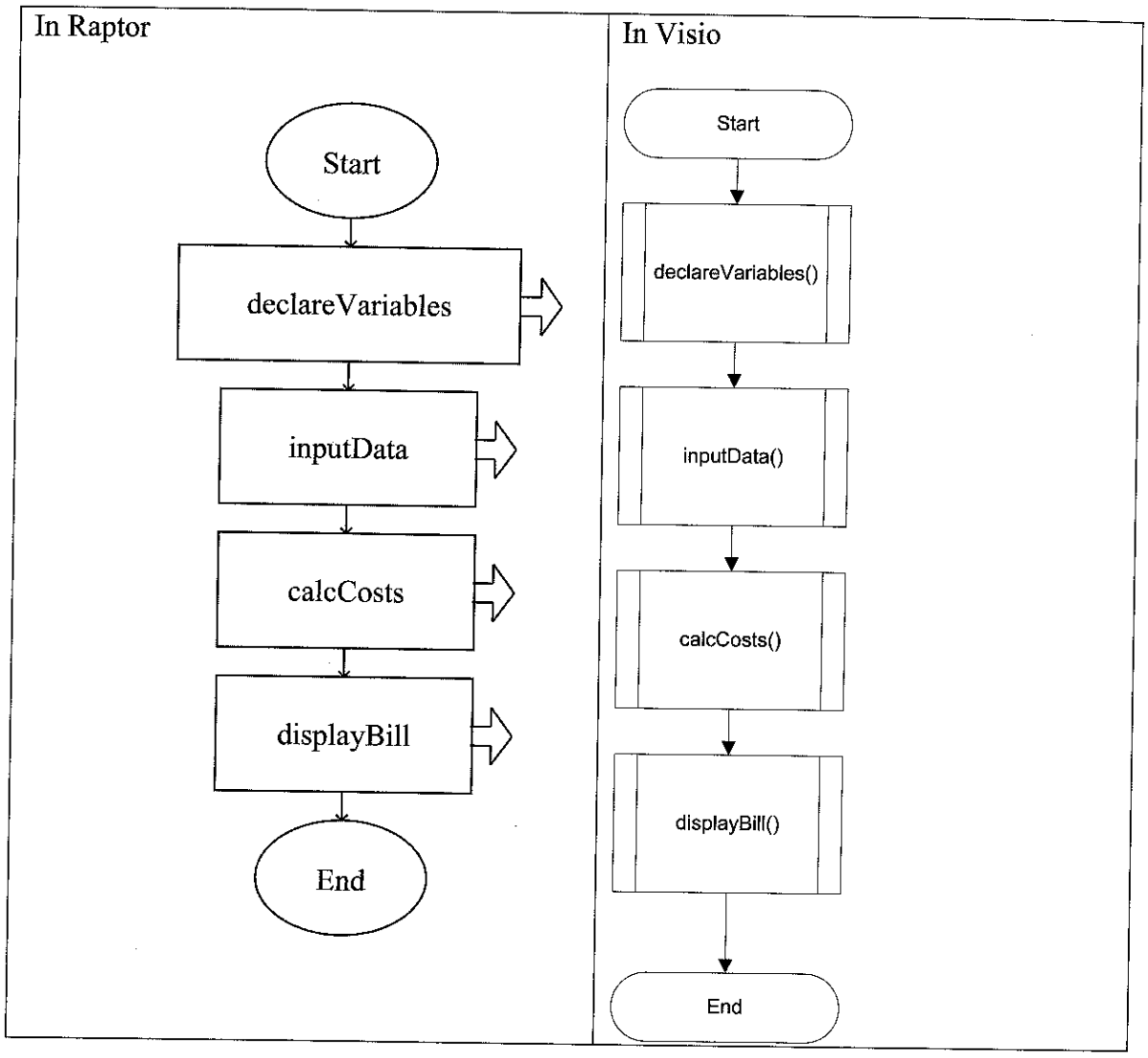
This lab requires you to think about the steps that take place in a program by designing a flowchart. Use an application such as Raptor or Visio. Read the following program prior to completing the lab.

```
Data Communications Corp wants a small program that will
calculate the cost of UTP it installs for their clients.
Write a program that will ask the user to input the name of
the client, the number of feet of cable installed.  The
program should then calculate and display a final bill.
Cost per foot of UTP is .21 cents.  Be sure to add on a tax
of 6%.  Final bill should include the total cost and client
name.  Be sure to add modules to your program.
```

Step 1: In main, create a module called declareVariables( ) that will set your variables to 0 or " ". Click the Call Symbol on the Left and Drag and Drop to the flow lines between Start and Stop. Double click on the Call Symbol and type the name of your first module. For example, type declareVariables in the Enter Call box. Do not put the ( ) when using Raptor. Click the Done button. A new box will pop up that will ask you to create a new tab. Click Yes. A new tab will be created for your new method. Notice the new Tab

called declareVariables. **Watch the Help Video 4-2 to see how to add modules and initialize variables in Raptor and Visio.**

Step 2: Continue this process to add your additional methods, which are inputData( ), calcCosts( ), and displayBill( ). Main should look like this:



Step 3: Click on the inputData module and add the necessary code to input clientName and feetUTP. **Watch the Help Video 4-2 to see how to input variables in Raptor and Visio.**

Step 4: Click the calcCosts module and add the necessary code to compute calculations. **Watch the Help Video 4-2 to see how to add calculations in Raptor and Visio.**

Step 5: Click the displayBill module and add the necessary code to display the clientName and totalCost to the screen. **Watch the Help Video 4-2 to see how to display variables in Raptor and Visio.**

Step 6: If you are using Raptor, you can run your program. Click Run, then Execute to Finish. For your input, enter a client name such as Bumpco Inc and 3758 feet of cable installed. If your program is coded correctly, the output should be as follows:

The clients name is Bumpco Inc
The final cost is $836.5308
----Run complete.

Step 7: The final step is to insert your finished flowchart into a Word document. Inside Raptor, select File and the Print to Clipboard from the menu. Inside your Word document, select Edit and Paste. You will have to do this for each module you created. In Visio, select Edit and then Copy Drawing. Inside your Word document, select Edit and Paste.

Lab 4.3 – Visual Basic and Modules

Critical Review

To create a module, write its definition. The keyword *Sub* is used before a module name, followed by parentheses. Here is the general format of a function definition in Visual Basic:

Sub moduleName( )
statement
statement
End Sub

Calling a function is done in order to make the module execute. The general format is:

ModuleName( )

Variables can be passed to modules by using the ByRef or ByVal keywords such as:
Sub moduleName(ByRef variable1Name as dataType, ByRef variable2Name as dataType)

Demo Video: View *lab4-3.wmv* in the Lab 4 folder on the accompanying Lab Demo Media and Startup Files CD

This lab requires you to write the following program in Visual Basic, console application using modules. Read the following program prior to completing the lab.

```
Data Communications Corp wants a small program that will
calculate the cost of UTP it installs for their clients.
Write a program that will ask the user to input the name of
the client, the number of feet of cable installed.  The
program should then calculate and display a final bill.
Cost per foot of UTP is .21 cents.  Be sure to add on a tax
of 6%.  Final bill should include the total cost and client
name.  Be sure to add modules to your program.
```

Step 1: Start Visual Basic, creating a new Console Application, and save your program. This will automatically add the following:

```
Module Module1

    Sub Main()

    End Sub
```

```
End Module
```

Step 2: Inside Main( ), declare your variables for this program. This should include the following declarations:

```
Dim clientName As String = "NO VALUE"
Dim feetUTP As Double = 0
Dim subTotal As Double = 0
Dim taxCost As Double = 0
Dim totalCost As Double = 0
```

Step 3: After the End Sub command in main, add a new module for inputData( ). This should look like:

```
Sub inputData(ByRef clientName As String, ByRef feetUTP As
Double)
```

The End Sub command will automatically be added. clientName and feetUTP are passed using the ByRef command because you need to retain the value of the variable.

Inside the module, add the following:

```
Console.Write("Enter the clients name: ")
clientName = Console.ReadLine()
Console.Write("Enter the number of feet of UTP installed: ")
feetUTP = Console.ReadLine()
```

This allows the user to enter the necessary data.

Step 4: Below that module, add a new module for calcCosts( ). Pass feetUTP as ByVal, and subTotal, taxCost, and totalCost as ByRef. Inside this module, process your calculations for subTotal, taxCost, and totalCost.

Step 5: Below that module, add a new module for finalBill( ). Pass clientName and totalCost as ByVal. Remember, ByRef is only needed if you are changing the value of a variable in a module. Since we are simply using the value of the variable in this case, ByVal works more efficiently. Inside the module, use Console.WriteLine( ) to display the values of the variables.

Step 6: In main under the variable declarations, add your module calls, passing the appropriate variables such as:

```
inputData(clientName, feetUTP)
calcCosts(feetUTP, subTotal, taxCost, totalCost)
finalBill(clientName, totalCost)

You might also include a pause by adding the following:
Console.Write("Press enter to continue...")
Console.ReadLine()
```

Step 7: Execute your program so that it works and paste the final code below. Sample output might look like:

Enter the client's name: Bumpco Inc.
Enter the number of feet of UTP installed: 4593
The client's name is Bumpco Inc.
The final cost is $1022.4018
Press enter to continue...

Submit your Visual Basic application (use the upload tool!)

Lab 4.4 – Challenge: Ping and Website Launches

Demo Video: View *lab4-4.wmv* in the Lab 4 folder on the accompanying Lab Demo Media and Startup Files CD

This lab requires you to write a Visual Basic program that includes modules to do the following tasks important in the field of networking and programming:

What is ping.exe?

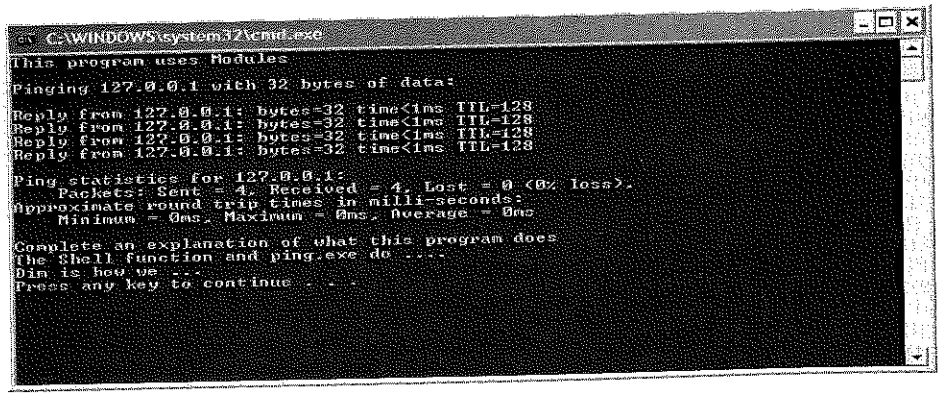
Ping.exe is a simple network utility to test whether or not a device such as a router, server or switch is contactable. When the device receives this information it then sends a reply saying "yes, I am here". Ping is used a lot in IT and network troubleshooting.

What is System.Diagnostics.Process.Start()?

This is a system function/module built into Visual Basic. This function/module can be used to start a process resource such as launching a website.

- Write a module called pingMe( ) that will use Ping.exe to get a response from your IP loopback address of 127.0.0.1. This should include the following line of code.
  - `Shell("Ping.exe 127.0.0.1", , True)`
- Write a module called openWebsite( ) that will use a string variable called myTargetURL to launch any website such as www.microsoft.com. This should include the following lines of code.
  - `Dim myTargetURL As String = "http://www.microsoft.com"`
  - `System.Diagnostics.Process.Start(myTargetURL)`
- Write a module called finalOutput( ) that explains what the Shell function does, what ping.exe does, and how you declare variables. Use additional resources to find explanations.
- In main( ), write calls to all three modules.

Your sample output might look as such. Additionally, a web browser should launch displaying the requested website.



The Visual Basic Code

Submit your Visual Basic application (use the upload tool!)