LAB GOALS

To learn how to write your own user defined modules (procedures and functions). Creating Drivers, and Stubs. Some simple ideas for debugging.

Open up Microsoft Visual Basic .NET.

Step 1: Create a new Console project named “User_Defined_Modules”. Don’t forget to add the following two lines at the beginning of the program:

Option Explicit On
Option Strict On

Step 2: Creating a Driver Program:

Using the main procedure, create a driver module that will call each individual procedure and function within your program. The driver should demonstrate the basic flow and overall structure of your program. For example, in our assignment 4, we are to create an amortization table. In order to do so, we must first obtain some information from the user about the loan (e.g. principal, interest rate and term), then we must calculate the monthly payment, then print the loan information (including the monthly payment which we calculated) and finally print the amortization table. (See assignment 4 for reference)

Below, the basic structure of assignment 4 is shown. Type this information in to your main() module.

```vbnet
Sub Main()
    Dim Principal As Double = 0.0
    Dim AnnualIntRate As Double = 0.0
    Dim TermInYears As Integer = 0
    Dim MonthlyPayment As Double = 0.0
    Dim PeriodsPerYear As Integer = 0
    ReadLoanInfo(Principal, AnnualIntRate, TermInYears, PeriodsPerYear)
    MonthlyPayment = Payment(Principal, AnnualIntRate, TermInYears, PeriodsPerYear)
    PrintLoanInfo(Principal, AnnualIntRate, TermInYears, PeriodsPerYear, MonthlyPayment)
    PrintAmortization(Principal, AnnualIntRate, TermInYears, PeriodsPerYear, MonthlyPayment)
    Console.ReadLine() 'Pause the program
End Sub
```

Step 3: Creating Stub Procedures and Functions:

Create a STUB module for each of the above procedures and functions. Note that the stubs must have the right NUMBER of parameters, the parameters must have the right TYPE, and the parameters should appear in the right ORDER. Furthermore, you must know if the parameters must be passed ByVal or ByRef.

Each stub must be designed such that, when it is called, it displays the fact that it has been entered, furthermore, it is a good idea to display the parameters that have been sent to that procedure. This way, you can verify that the values that have been sent to procedures have arrived correctly.

```vbnet
'-------------------------------------------------------------
' Get the loan information from the user and pass it back to the main program. All the variables are sent as pass ByVal, since the job of this procedure is to obtain the basic loan information from the user and make them available to the rest of the program. In other words, the data needs to be transferred back to the main() module.
Sub ReadLoanInfo(ByVal Principal As Double, ByVal AnnualIntRate As Double, ByVal TermInYears As Integer, ByVal PeriodsPerYear As Integer)
    Console.WriteLine("Entering Read Loan Info")
    Console.WriteLine(Principal)
    Console.WriteLine(AnnualIntRate)
    Console.WriteLine(TermInYears)
    Console.WriteLine(PeriodsPerYear)
    Console.WriteLine("Exiting Read Loan Info")
End Sub

'-------------------------------------------------------------
'Calculate the period (monthly) payment. This function is similar the builtin PMT()
'Function provided by VB, except in this case we are building our own.
'Please note the syntax of the function vs. a procedure. Also note that a function always returns a value, and the type of that value much match the function type.
```
Function Payment(ByVal Principal As Double, _
ByVal AnnualIntRate As Double, _
ByVal TermInYears As Integer, _
ByVal PeriodsPerYear As Integer) As Double

Dim MtlyPayment As Double
Console.WriteLine("Entering Payment Function")
Console.WriteLine("Exiting Payment Function")
Return MtlyPayment
End Function

Run: After each STUB procedure or function is added to the program, compile and run your program. If your program does not compile, fix the syntax errors and compile the program again. Once you are able to successfully run the program, check to see if the flow of control appears to be correct, and if the values sent to the procedure are properly sent.

Step 4: Refinement Step:
Now that the overall structure of the program is correct, we must concentrate on the individual procedures and functions. For example, we must ask ourselves, what is the purpose of ReadLoanInfo(). Then write the code to achieve that purpose.

Sub ReadLoanInfo(ByVal Principal As Double, _
ByRef AnnualIntRate As Double, _
ByRef TermInYears As Integer, _
ByRef PeriodsPerYear As Integer)

Console.WriteLine("Entering Read Loan Info")
' Your Code to read the Principal, AnnualInterestRate, etc. goes here....
',
Console.WriteLine("Exiting Read Loan Info")
End Sub

Let us assume that the purpose of the ReadLoanInfo() is to ask the user the basic information about a loan such as the principal, interest rate and term and then send that information back to the main program. Given this assumption, we must then write the corresponding code which interacts with the user and collects the loan information and sends it back through the procedure’s parameter list. At this point, we also have the opportunity to perform some input validation and error checking. For example we can check to see if the values provided by the user are proper. (Remember the IsNumeric() function)

Step 5: Debugging Step:
After step 4 is completed, your program should have read the essential loan information and that information should be available back in the main program. (because the data read is passed by Reference, and therefore it will change the variable back in the main() procedure. However, in order to be sure we must verify this. How would you verify that the ReadLoanInfo() procedure actually perform its task correctly?

You have two options available to you. The first option is to verify that the information brought back from ReadLoanInfo() is correct, by simply printing the values in each parameter (See below). The second option is to use the VB debugger. (More on using the debugger in a future lab).

Sub Main()
    '...
    ReadLoanInfo(Principal, AnnualIntRate, TermInYears, PeriodsPerYear)
    Console.WriteLine(Principal)
    Console.WriteLine(AnnualIntRate)
    Console.WriteLine(TermInYears)
    Console.WriteLine(PeriodsPerYear)
    '...
End Sub