LAB GOALS

To develop console application using the following constructs:
1) Variables (integer, etc.)
2) Operators (e.g. MOD)
3) Console I/O (Readline(), Writeline())
4) Type Conversion (Cint(), Cstr(), etc.)

Proceed to open up Microsoft Visual Basic .NET.

Step 1: Create a new console project named “Console_Application_Variables_MOD_IO”

Step 2: Note the difference between a Windows and a Console application. In a console application:

1) there is no GUI form
2) you are placed directly in the code window
3) the toolbox does not have any tools in it.
4) the code window starts with the line: “Module Module1”, instead of starting with the line “Class Form1”
5) The code window already has a sub program in it. It is called Main().

Some questions that comes to mind, is what good is a console application? Why would we want to use a console application? Can one write useful programs using just console applications?

Well, the answer to these questions are: Yes, Console applications are very useful, they are just as powerful in allowing the programmer to solve problems, and people use them quite a bit. In fact before the popularity of GUI based operating systems such as Mac OS and Windows, most applications were developed as console mode applications.

The primary differences between programming in console mode and windows mode is that windows application use a GUI to interact with the user and are therefore more esthetically pleasing. Also, windows applications are event driven, so it is the user who has some control over the interaction with the program (e.g. by clicking on different controls, or by typing in a certain textbox, etc). Finally, I/O in windows applications is usually done through GUI objects such as Textboxes or Labels.

Step 3: Ok, enough about that, let’s begin by including the familiar lines (below) at the beginning of our program. Place these lines right before the line “Module Module1”.

Option Explicit On
Option Strict On

Run: Compile and Run your program. You should see a console window (black window) appear, and quickly disappear. If this happens, your program is working!! The reason it disappears, is that there is no reason for it to stay around. It has compiled and done what you asked it to do. After that it simply closes the console window.
Step 4: Well, let’s put in some code in the program that makes it pause until we tell it to close the window. Place the following line in the code right before the line “End Sub”

```
Console.ReadLine()  'pause the program until the user hits the enter key
```

Run: Compile and Run your program. You should see a console window (black window) appear and wait for you to hit the ENTER key. After the enter key is pressed the console window will disappear.

Step 5: Ok, now we have a very simple and functioning console application. Let’s add more functionality to it. We would like to create a program that asks the user to enter two integers we then want to display those numbers to verify that they are correctly inputted. After that verification, we would like to do some calculation and place the result in a new integer variable. Then print the result and verify that it is correct.

- Create three integer variables with the following names: (first, second, and result). Place these declaration lines right after the “Sub Main()” procedure.

  ```
  Dim first, second As Integer
  Dim result As Integer
  ```

- Tell the user to enter the value for the first number.

  ```
  Console.Write("Please enter the 1st number: ")
  ```

Run: Compile and Run your program. You should see a console window (black window) appear, display the above line and wait for you to hit the ENTER key. After the enter key is pressed the console window will disappear. Note that if you enter a number at this point it will be ignored!

- In order for us to actually read the number from the keyboard, we must use the Readline() function. Type the following line and compile the program again.

  ```
  first = Console.ReadLine()
  ```

Run: Compile and Run your program. Even before compiling your program, you should have noticed a problem with the Console.ReadLine() statement. As you can see it is underlined by the compiler. This is compiler’s way of telling you that you have a syntax error. The compiler is telling you that the result of the Readline() statement is a string, but you are trying to put that string into a variable of type integer. In order to correct this problem, we must convert the result of the readline() to an integer. Let’s do that below, using the Cint() function. This function converts the string to an integer, so it can be placed in our variable.

  ```
  first = CInt(Console.ReadLine())
  ```

Run: Compile and Run your program one more time. Your program should now compile and run. It should wait for you to enter the number, and after entering the number it should pause until you hit the enter key.

- Repeat the above for the second number.

  ```
  Console.Write("Please enter the 2nd number: ")
  second = CInt(Console.ReadLine())
  ```

Run: Compile and Run your program.

- To verify the two numbers are entered correctly, let’s print them out:

  ```
  Console.WriteLine("{0}, {1}", first, second)
  ```
• Ok, now that we verified that the numbers we read in are actually correct, let's do some calculation. Let's write some code which will take the first number and MOD it by the second number, and places the result of the calculation in the variable named "result". Once the calculation is complete, we should print the result to make sure it is working.

```csharp
result = first Mod second
Console.WriteLine("Result: {0}" , result)
```

Run: Compile and Run your program.