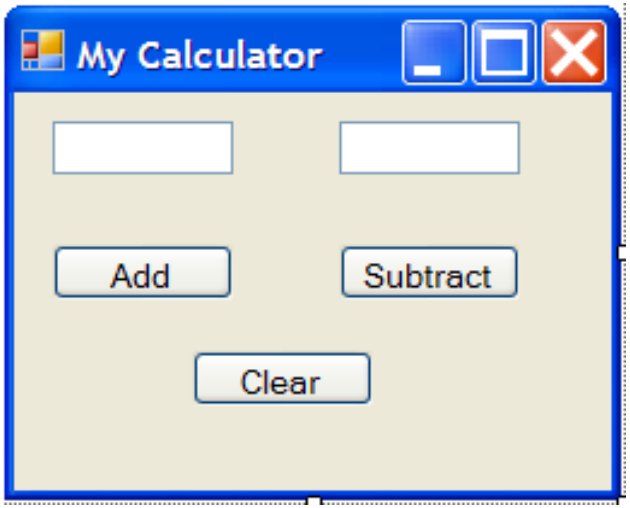
Creating A Calculator

Programs that provide calculations are among the more common type of projects you will encounter as programmer.
We will begin with a simple calculator program that adds and subtracts two numbers and then displays the answer in a MsgBox.



The user inputs two numbers into the text boxes. If the user clicks Add the sum of the two numbers appears in a message box. The difference of the two numbers appears if the Subtract button is clicked.
The Clear button removes all entries from the text boxes.

Using val to Convert the Input Values from Text to Numeric

When the user types in the values into a text box the values represent text even if they look like numbers.

By using the val() function, the text get converted into numeric form.

Since the values being collected are numeric, the variables holding these values must be numeric as well. In this case we will use variables of type integer.

Declaring Integer Variables

```
Dim numOne As Integer
Dim numTwo As Integer
Dim answer As Integer
```

We declare 3 variables, 2 to hold the input values from the text box and one to hold the answer.
Getting the values from the text boxes to the variables is very similar to our previous program PersonalInfo. The only difference is the use of the val() function.

```
numOne = Val(txtNum1.Text)
numTwo = Val(txtNum2.Text)
```

Adding

Lets focus on the addition button. Since we are adding the two values that the user inputs into the text box, the code will look like this.

```
answer = numOne + numTwo
```

Displaying the answer in a MsgBox is a simple matter of including the variable answer within the MsgBox brackets.

```
MsgBox("The answer is: " & answer)
```

The Code for Addition and Subtraction

```
Private Sub btnAdd_Click(ByVal sender As
    Dim numOne As Integer
    Dim numTwo As Integer
    Dim answer As Integer

    numOne = Val(txtNum1.Text)
    numTwo = Val(txtNum2.Text)
    answer = numOne + numTwo
    MsgBox("The answer is: " & answer)
End Sub

Private Sub btnSubtract_Click(ByVal sender
    Dim numOne As Integer
    Dim numTwo As Integer
    Dim answer As Integer

    numOne = Val(txtNum1.Text)
    numTwo = Val(txtNum2.Text)
    answer = numOne - numTwo
    MsgBox("The answer is: " & answer)
End Sub
```

Notice how the variables were declared twice, once for each button.

Declaring variables twice is not good programming.

Lets remove the 2nd declaration in btnSubtract and try and avoid this repetition.

Oops

1	Name 'numOne' is not declared.	Form1.vb	20	9	Calculator
2	Name 'numTwo' is not declared.	Form1.vb	21	9	Calculator
3	Name 'answer' is not declared.	Form1.vb	22	9	Calculator
4	Name 'numOne' is not declared.	Form1.vb	22	18	Calculator
5	Name 'numTwo' is not declared.	Form1.vb	22	27	Calculator
6	Name 'answer' is not declared.	Form1.vb	23	36	Calculator

When we try and run the program, the compiler complains that we have not declared the variables that we are using on lines 20-23.

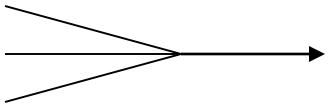
These lines fall within the btnSubtract_Click() Sub.

We know that we declared these variables up in the btnAdd_Click() Sub so what is going on.

The Issue of Scope

Variables that are declared within a Sub are said to be local variables. These local variables exist only within the area which they are declared.

```
Private Sub btnAdd_Click(ByVal sender As
    Dim numOne As Integer
    Dim numTwo As Integer
    Dim answer As Integer

    numOne = Val(txtNum1.Text)
    numTwo = Val(txtNum2.Text)
    answer = numOne + numTwo
    MsgBox("The answer is: " & answer)
End Sub
```



These 3 variables only have a life within the btnAdd_Click Sub. As soon as the program runs other code beyound this Sub, the variable cease to hold values.

We say the variable is no longer in Scope.

Extending Scope

We could simply re-declare the variables in every single Sub that we needed them but as stated before, this is bad programming practice.

Firstly, it is confusing to have two separate variable declarations of the same name. Also, anytime you are writing the same code again, you are adding waste and unnecessary complexity to your code.

Expanding the Scope

Declaring a variable within a Sub creates a local variable whose life exists only within the confines of the Sub.

If you declare the variable within a bigger confine, you are said to be expanding the scope of he variable.

French Francs and European Euros

Imagine a currency to be a variable. Printing the currency is going to represent an analogy of declaring a variable.

Imagine having 100 French francs. This money is printed only within the borders of France. It, co-incidentally only has value within France. Trying to spend this currency outside the borders of France will not be possible.

Now consider the effects of the European Euro. Europe is the continent which contains various countries like France, Britain, Germany, Italy etc.

The new European currency is created in Europe, it has a broader spectrum and is accepted ats full value within all the countries within Europe.

A variable may be declared within a whole project rather than inside a Sub, we then call this variable a global variable rather than a local variable. Its value or life is said to be in scope for the entire program.

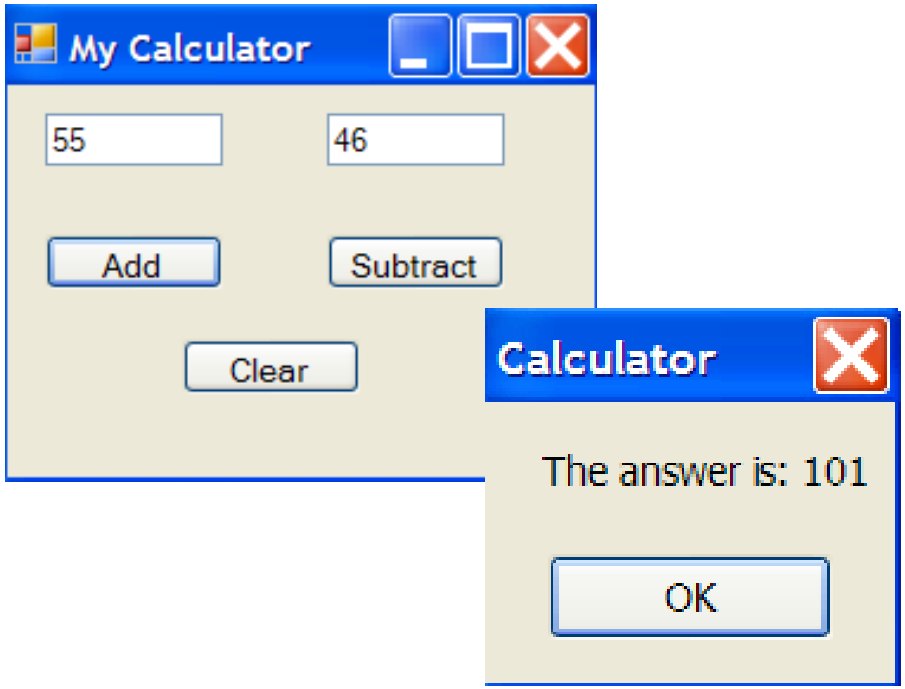
Declaring Global Variables

```
Public Class Form1
    Dim numOne As Integer
    Dim numTwo As Integer
    Dim answer As Integer

    Private Sub btnAdd_Click(ByVal sender As System.
        numOne = Val(txtNum1.Text)
        numTwo = Val(txtNum2.Text)
        answer = numOne + numTwo
        MsgBox("The answer is: " & answer)
    End Sub

  Private Sub btnSubtract_Click(ByVal sender As Syste
        numOne = Val(txtNum1.Text)
        numTwo = Val(txtNum2.Text)
        answer = numOne - numTwo
        MsgBox("The answer is: " & answer)
    End Sub

    Private Sub btnClear_Click(ByVal sender As Syster
        txtNum1.Text = ""
        txtNum2.Text = ""
    End Sub
End Class
```

In this code, the declarations have been made outside any individual Sub, but inside the Form class. What this means is that anywhere inside this form, including the Subs, the value of the variables will be in scope.

The result in this case is that the variables are declared only once but are used in a both Subs.

Running The Calculator



Global Variables Everywhere?

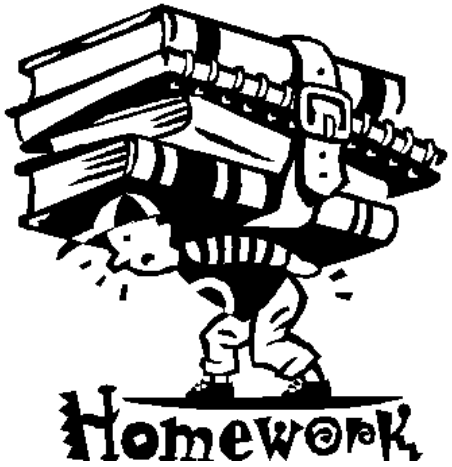
So if global variables can be used everywhere, why even bother with local variables? Why not just declare everything global?

For the same reason that Canadian dollars are NOT just used in every country. Canadian dollars are used only within Canada. This allows the country to control the amount of currency that is printed and generally to retain control on monetary issues.

The rule in programming is to declare variables so that they have enough scope to do their jobs.

Restricting the scope of a variable helps make the program more secure and less prone to errors.

Think of it it terms of secret agents. Variables are used on a need-to-know basis. If a Sub doesn’t need to use a variable, don’t give it access to it.



Modify the Calculator program so that it has + and – icons instead of text on the buttons.

Add Multiplication and Division buttons to the program.

What accuracy problems do you run into with division.
Hint: Integers accept what type of numerical data

Home Work