Software Design and Development

6 Repetitions

If you don't succeed, don't give up.

Terry Marris July 2009

Previously we looked at the primitive data types and selections. We continue our study of program structures by looking at repetitions or loops, or, as they are known by us programmers, iterations.

Visual Basic 2008 provides several different kinds of loops. We shall focus on just one because it will do all that we want, safely, and without overburdening our brains.

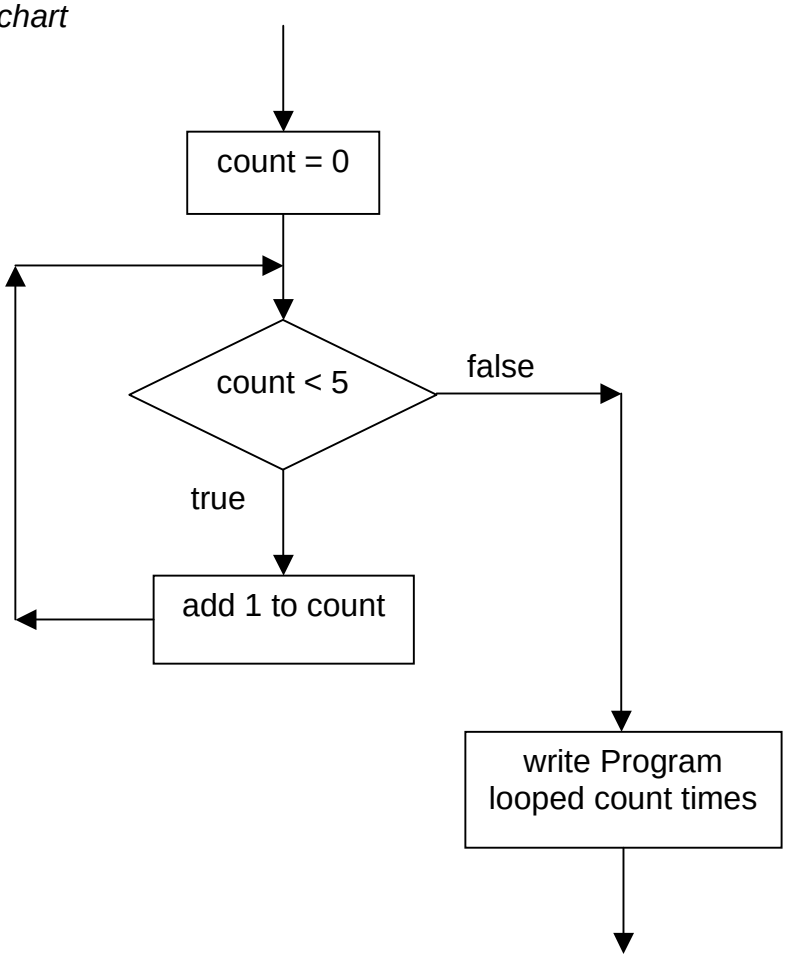
6.1 While ... End While

The design for our first program is:

Structured English

```
count = 0
while count < 5
  add 1 to count
endwhile
write out "program looped" count "times"
```

Program Flowchart



count starts with the value 0.
We come to the boolean that controls entry to the loop. *count* < *5* is true.
So we enter the loop and add 1 to *count*. *count* now has the value 1.

We come again to the loop control. *count* < *5* is true.
So we enter the loop and add 1 to *count*. *count* now has the value 2

We come again to the loop control. *count* < *5* is true.
So we add 1 to *count*. *count* now has the value 3

We come again to the loop control. *count* < *5* is true.
So we enter the loop and add 1 to *count*. *count* now has the value 4

We come again to the loop control. *count* < *5* is true.
So we enter the loop and add 1 to *count*. *count* now has the value 5

We come again to the loop control. *count* < *5* is now **false**.
So we exit from the loop and display the value of *count*.

We display *Program looped 5 times*.

User Interface

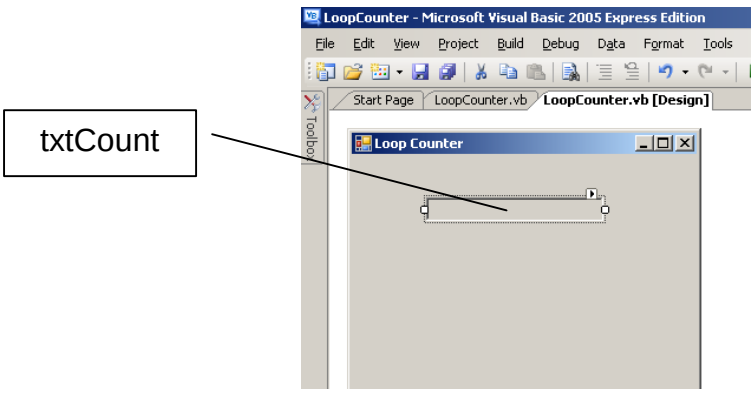
1. Start a new project and name it *LoopCounter*

2. Name the VB file *LoopCounter.vb*

3. Set

    a. Form
        i. Name = *frmLoopCounter*
        ii. Text = *Loop Counter*

    b. TextBox
        i. Name = *txtCount*
        ii. ReadOnly = *True*



4. Rapid double click on a blank part of the form and write, between *Sub frmLoopCounter_Load* .... and *End Sub*,

```
Dim intCount As Integer = 0
While intCount < 5
    intCount = intCount + 1
End While
txtCount.Text = "The program looped "+intCount.ToString()+" times"
```

An example of a program run is

Exercise 6.1

1. Try out the program, shown above, that reports the program looped 5 times.
2. Modify the program so that it will loop 7 times.

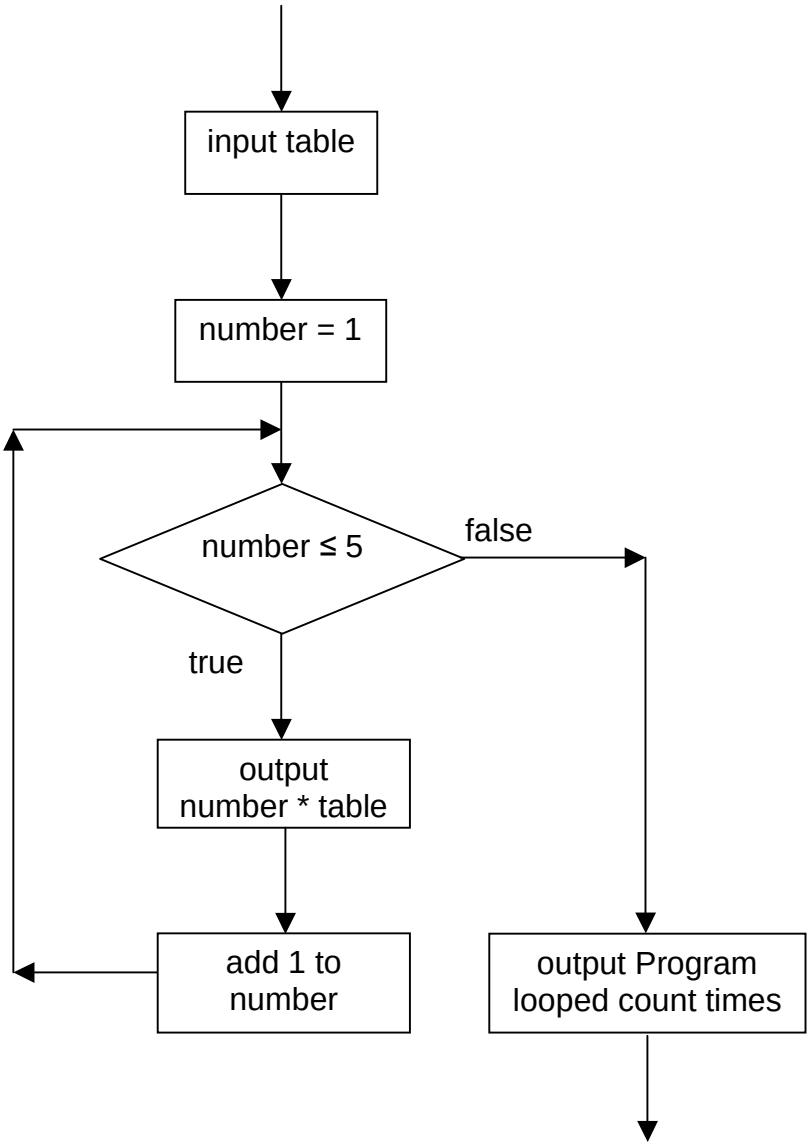
6.2 Multiplication Table

The design for our next program that displays a multiplication table is:

Structured English

```
read in table
number = 1
while number <= 5
    answer = number * table
    write out number x table = answer
    add 1 to number
endwhile
```

Program Flowchart



We input *table*; suppose it is 3.
number starts off at 1.

We come to the boolean that controls entry to the loop. *number* ≤ *5* is *true*.
So we enter the loop.
We output 1 x 3 = 3 and add 1 to *number*. *number* now has the value 2.

We come again to the loop control. *number* ≤ *5* is *true*.
So we enter the loop.
We output 2 x 3 = 6 and add 1 to *number*. *number* now has the value 3.

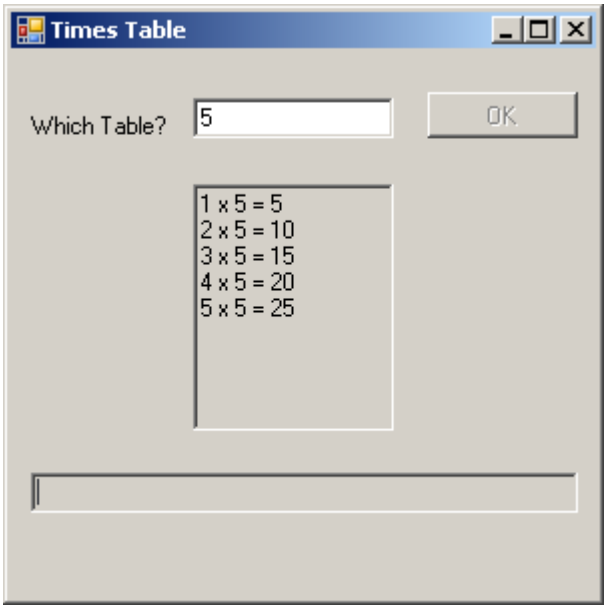
We come again to the loop control. *number* ≤ *5* is *true*.
So we enter the loop.
We output 3 x 3 = 9 and add 1 to *number*. *number* now has the value 4.

We come again to the loop control. *number* ≤ *5* is *true*.
So we enter the loop.
We output 4 x 3 = 12 and add 1 to *number*. *number* now has the value 5.

We come again to the loop control. *number* ≤ *5* is *true*.
So we enter the loop
We output 5 x 3 = 15 and add 1 to *number*. *number* now has the value 6.

We come again to the loop control. *number* ≤ *5* is now **false**.
So we exit the loop.

An example of a program run is:



The user enters 5

The program prints

1 x 5 = 5
2 x 5 = 10
3 x 5 = 15
4 x 5 = 20
5 x 5 = 25

number ranges in value from 1 up to 5 inclusive.

table is 5

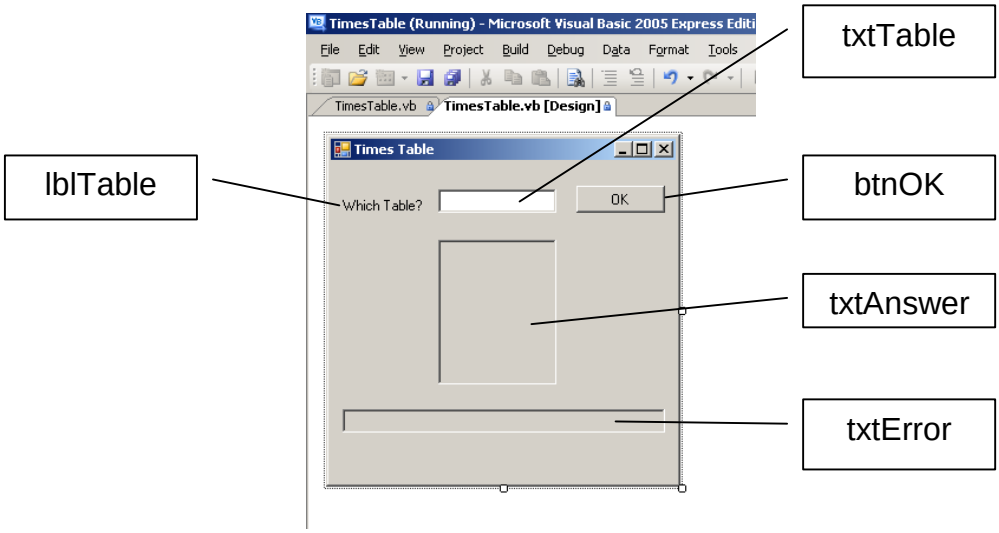
answer ranges in value from 5 up to 25 in steps of 5.

User Interface

1. Start a new project and name it *TimesTable*

2. Name the VB file *TimesTable.vb*

3. Set
   - **a.** Form
     - **i.** Name = *frmTimesTable*
     - **ii.** Text = *Times Table*
   - **b.** Label
     - **i.** Name = *lblTable*
     - **ii.** Text = *Which Table?*
   - **c.** TextBox
     - **i.** Name = *txtTable*
   - **d.** Button
     - **i.** Name = *btnOK*
     - **ii.** Text = *OK*
   - **e.** TextBox
     - **i.** Name = *txtAnswer*
     - **ii.** Multiline = *True* ← **Notice**. We want a multiline text box
     - **iii.** ReadOnly = *True*
   - **f.** TextBox
     - **i.** Name = *txtError*
     - **ii.** ReadOnly = *True*



Programming Code

4. Enter the VB code, shown below, under the OK button.

```
Try
   Dim strAnswer As String = ""
   Dim intTable As Integer = Convert.ToInt32(txtTable.Text)
   Dim intAnswer As Integer = 0
   Dim intNumber As Integer = 1
   While intNumber <= 5
       intAnswer = intNumber * intTable
       strAnswer = strAnswer + intNumber.ToString() + " x " + _
       intTable.ToString() + " = " + intAnswer.ToString() + vbCrLf
       intNumber = intNumber + 1
   End While
   btnOK.Enabled = False
   txtAnswer.Text = strAnswer
Catch ex As FormatException
   txtError.Text = "Error: not a number"
Catch ex As OverflowException
   txtError.Text = "Error: number too large"
End Try
```

Dim strAnswer As String = "" is used to store our multiplication table line by line. We add to it every time we go round the loop.

Dim intTable As Integer = Convert.ToInt32(txtTable.Text) gets the table that the user wants to see. In our example this was 5.

Dim intAnswer As Integer = 0 holds the answer for each line of our multiplication table.

Dim intNumber As Integer = 1 counts our repetitions through the loop. Its value ranges from 1 up to 5 inclusive.

While intNumber <= 5 loops for as long as *intNumber* is less than (or equal to) 5. Expect its value to increase every time round the loop.

*intAnswer = intNumber * intTable* records the answer to each multiplication.

strAnswer = strAnswer + intNumber.ToString() + " x " + _
intTable.ToString() + " = " + intAnswer.ToString() + vbCrLf

Adds a line of the table, in form of m x n = a, e.g. 2 x 5 = 10, to the string, *strAnswer*. The string gets longer everytime we go round the loop.

The + is known as the concatenation operator . It joins (adds) strings (text) together. The numbers (*intNumber* and *intTable* and *intAnswer*) are converted to strings by the *ToString()* method.

The _ (underscore) is known as the continuation operator. You use it when you want to continue a VB statement onto the next line.

vbCrLf is VB for a newline (Carriage return, Line feed). We write a line of the answer. Then we write a *newline* so that the next answer starts on a new line.

intNumber = intNumber + 1 adds 1 to *intNumber*. Its value increases every time you go round the loop. Eventually, its value reaches 6 and the loop terminates.

txtAnswer.Text = strAnswer copies the string that contains our answer into the output text box so that we can see it.

btnOK.Enabled = False disables the OK button when the table is displayed on the screen.

```
Catch ex As FormatException
    txtError.Text = "Error: not a numberr"
Catch ex As OverflowException
    txtError.Text = "Error: number too large"
```

catches out the idiots who cannot use your program sensibly. You include it every time you deal with number input.

Exercise 6.2

1. Try out the program, shown above, that prints a multiplication table up to 5 X n where n is a number input by the user.

2. Modify the program so that it displays a multiplication table up to 12 X n.

Bibliography

None

Next we look at functions