Unit 7

Loop

Introduction

So far we have seen that each instruction is executed once and once only. Some time we may require that a group of instructions be executed repeatedly, until some logical condition has been satisfied. This is known as looping. You can repeat the statements until a condition is true or until a condition is false or a specified number of times. In Visual Basic 2008, there are three types of Loops, they are the **For.....Next** loop, the **Do** loop and the **While.....End while** loop. Through this we shall examine the structure of each of the loops in details as well as demonstrating them with examples.

Lesson 7.1

For… Next Statements

Upon completion of this unit you will be able to:

- *Define* loop.
- *Use* for next loop.

Outcomes

Loop

Looping is required when we need to process something repetitively until a certain condition is met. For example, we can design a program that adds a series of numbers until the sum exceeds a certain value. Loop structures allow you to execute one or more lines of code repetitively. So we can say that the process of repeating a series of instruction is called looping. The group of repeated instructions is called loop.
You can repeat the statements

- Until a condition is true
- Until a condition is false
- A specified number of times

In Visual Basic 2008, there are three types of Loops, they are the **For...Next** loop, the **Do** loop. and the **While...End while** loop. We shall

examine the structure of each of the loops in details as well as demonstrating them with examples

- For...Next
- Do...Loop
- While…End While

For… Next Statements

When we want to repeat the statements in a loop a specific number of times then for next loop is ideal. The for next loops use the for and next statement and a counter variable called loop index which determines the number of times the statements inside the loop will be executed. Here loop index must be a numeric variable.

```
For counter = start to end [Step increment]
        statements

Next
```

The **For** statement specifies the counter variable i, and its start and end values. The **Next** statement increases the counter variable i by one. With the **Step** keyword, you can increase or decrease the counter variable by the value you specify. When the step is omitted the increment is assumed to be 1. Each For statement has a corresponding Next statement which must be followed. All statements between the For and Next are considered to be the body of the loop and the statements will be executed the specified number of times.

In the example below, the counter variable i is increased by two, each time the loop repeats.

```
For i=2 To 10 Step 2
  some code
Next
```

To decrease the counter variable, you must use a negative **Step** value. You must specify an end value that is less than the start value. In the example below, the counter variable i is decreased by two, each time the loop repeats.

```
For i=10 To 2 Step -2
  some code
Next
```

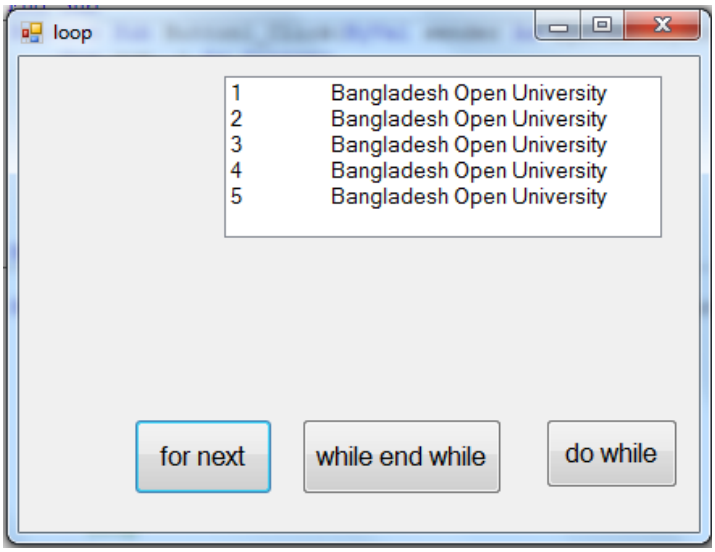
We may need to display any text several times. Will we write the text again and again? Let us understand the concept of for loop with the help of an example. In the following program we are going to display the text “Bangladesh Open University” for five times.

Example 1

In the design stage, you need to insert a ListBox into the form for displaying the output, named List1 and a Button. The program uses the **Add** method to populate the ListBox. The statement ListBox1.Items.Add (i &vbTab & sum) will display the values of i and text and uses the vbTab function to create a space between the i and text.

```
Private Sub Button3_Click(ByVal sender As
System.Object, ByVal e As System.EventArgs) Handles
Button3.Click
        Dim txt As String, i As Integer
        txt = "Bangladesh Open University"
        For i = 1 To 5
            ListBox1.Items.Add(i & vbTab & txt)
        Next
End Sub
```

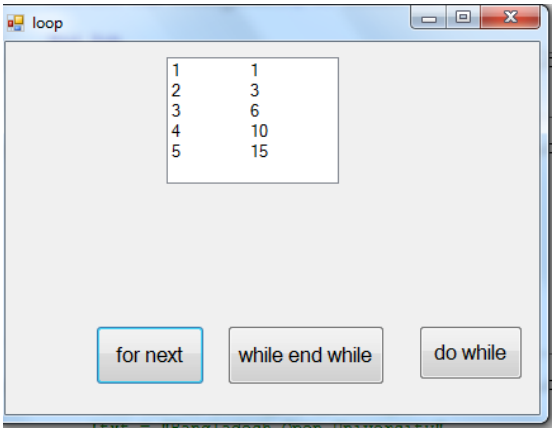
The output for this program is as follows:



Let us consider one more program to *compute the summation of 1+2+3……+5*

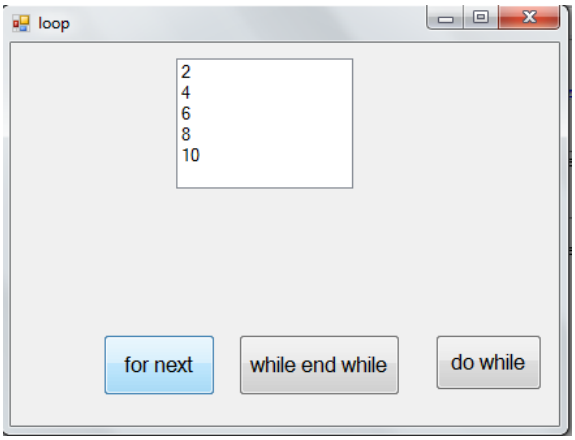
```
Private Sub Button3_Click(ByVal sender As
System.Object, ByVal e As System.EventArgs) Handles
Button3.Click
        Dim sum, i As Integer
        sum = 0
        For i = 1 To 5
            sum = sum + i
            ListBox1.Items.Add(i & vbTab & sum)
        Next
    End Sub
```

The output for this program is as follows:



Let us consider one more program to display even numbers between 2 to 10 through use of for loop.

```
Private Sub Button3_Click(ByVal sender As
System.Object, ByVal e As System.EventArgs) Handles
Button3.Click
        Dim i As Integer
        For i= 2 to 10 step 2
            ListBox1.Items.Add(i)
        Next
    End Sub
```



Counting backward

We may use a negative number for the step increment to decrease the loop index rather than increase it. Like

For i= 10 to 1 step -1

Activity

1. write a program to display a series

10+8 +6 +……..+0

Lesson 7.2

Do...Loop

Outcomes

Upon completion of this unit you will be able to:

- *Use* Do loop.

Do Loop

Use **Do...Loop** statements to run a block of statements an indefinite number of times. Execution of a Doo Loop continues while a condition is true until a condition is true. The general form of the statement is

Do While Condition

Statements

Loop

The loop operates in the following method:

If the result is true, then the program statement (the body of the loop) is executed. The statement may be a compound statement. The statement is evaluated again. If it is again true, the statement is executed once more. This process continues until the test expression becomes false.

Note it

As long as the Condition is true the Statements block will execute.

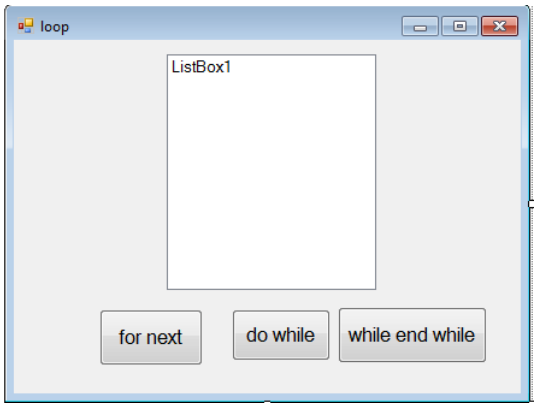
Tip

Use the Do Loop when the exact number of iterations is unknown.

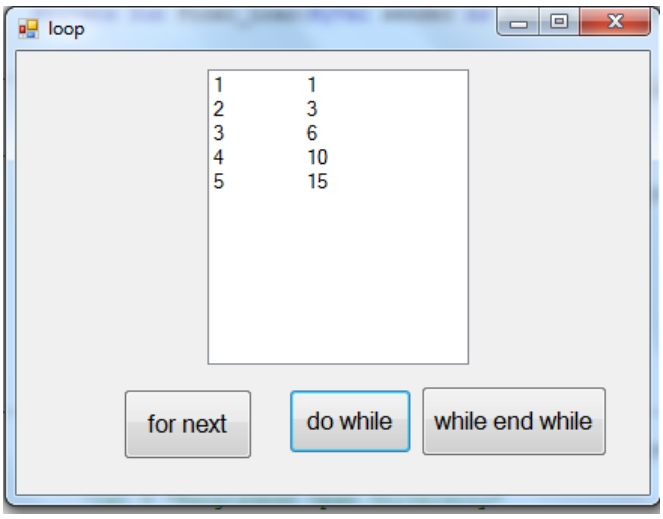
Let us consider an example, which will calculate the sum of first 10 digits.

sum = 1+2 +3 +……+10
here, we find the summation of 1+2+3+4+……+10.

In the design stage, you need to insert a ListBox into the form for displaying the output, named List1. The program uses the **Add** method to populate the ListBox. The statement ListBox1.Items.Add(n &vbTab& sum) will display the values of n and sum and uses the vbTab function to create a space between the headings n and sum.




```
Private Sub Button2_Click(ByVal sender As
System.Object, ByVal e As System.EventArgs) Handles
Button2.Click
        Dim n, i, sum As Integer
        n = 5
        Do While i < n
            i += 1
            sum = sum + i
            ListBox1.Items.Add(i & vbTab & sum)
        Loop
 End Sub
```



Lesson 7.3

While loop

Outcomes

Upon completion of this unit you will be able to:

- *Use* while loop.

While ...End While Loop

A while loop performs its test before the body of the loop is executed, whereas a do...loop makes the test after the body is executed. The structure of a While…End While is very similar to the Do Loop. The following is the format:

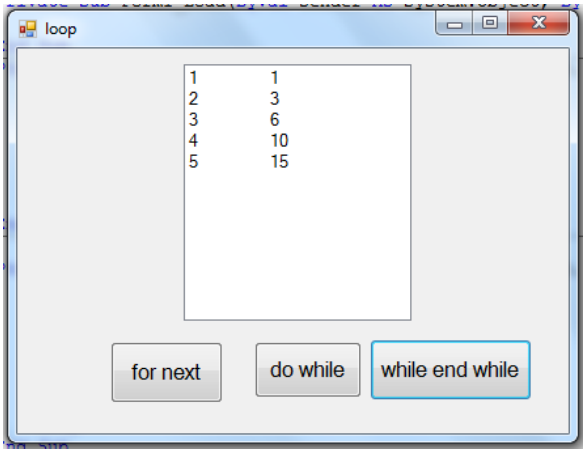
While *Condition*
    Statements
End While

As long as the *Condition* is true the *Statements* block will execute
One important difference between the while loop and the do-while loop is that in the while loop, the loop repetition test is performed before each execution of the loop body; the loop body is not executed at all if the initial test fails. In the do-while loop, the loop termination test is performed after each execution of the loop body; hence, the loop body is always executed at least once. Let us take an example to explain it further.

Example

sum = 1+2 +3 +……+10

```
Private Sub Button1_Click(ByVal sender As
System.Object, ByVal e As System.EventArgs) Handles
Button1.Click
        Dim sum, i As Integer
        While i <> 5
            i += 1
            sum = sum + i
            ListBox1.Items.Add(i & vbTab & sum)
        End While
End Sub
```



Assessment

Assessment

1. What is meant by looping? Describe two different forms of looping.
2. What happens if the condition in a while loop is initially false?
3. What is the minimum number of times the body of a do… while loop is executed?
4. Define syntax for do…. While loop?
5. Which is the better loop to use, the for loop or the while loop?
6. What is a special advantage of the for loop?
7. How does the for loop operate?
8. What is the difference between while and do-while loops.
9. Explain the differences between doo loop and for next loop.
10. Explain all looping statements in VB with simple example.
11. Write a program that will calculate the sum of odd numbers and sum of even numbers for numbers 1 to 10.
12. Write a while loop that displays numbers 2, 4, 6, 8 ..... 12.
13. Write a do-while loop that displays numbers 1, 3, 5, 7 ..... 11.
14. Write a for-loop that displays numbers from 10 to 20.
15. Write a program that will print the factorial of any number.