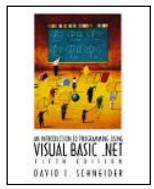
Chapter 6: Repetition

- **Do Loops (6.1)**
- **Processing Lists of Data with Do Loops (6.2)**
- **For...Next Loops (6.3)**



David I. Schneider, *An Introduction to Programming using Visual Basic.NET, 5th Edition*, Prentice Hall, 2002.

Do Loops (6.1)

- A loop is one of the most important structures in programming.
- Used to repeat a sequence of statements a number of times.
- The Do loop repeats a sequence of statements either as long as or until a certain condition is true.

Do Loops (6.1) (cont.)

✶Do Loop Syntax

```
Do While condition
   statement(s)
Loop
```

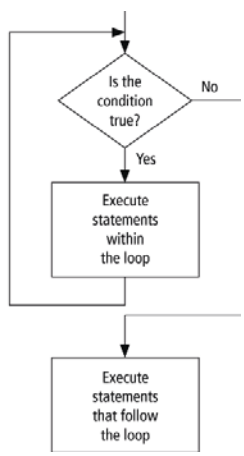
Condition is tested, If it is True, the loop is run. If it is False, the statements following the Loop statement are executed.

These statements are inside the body of the loop and are run if the condition above is True.

Do Loops (6.1) (cont.)

✶Pseudocode and Flow Chart for a Do Loop

```
Do While condition is true
   Processing step(s)
Loop
```



Do Loops (6.1) (cont.)

✶Example 1

```
Private Sub btnDisplay_Click(...) _
    Handles btnDisplay.Click
'Display the numbers from 1 to 7
  Dim num As Integer = 1
  Do While num <= 7
    lstNumbers.Items.Add(num)
    num += 1 'Add 1 to the value of num
  Loop
End Sub
```

Do Loops (6.1) (cont.)

✶Example 2

```
passWord = ""
Do While passWord <> "SHAZAM"
  passWord = InputBox("What is the password?")
  passWord = passWord.ToUpper
Loop
```

passWord is the loop control variable because the value stored in passWord is what is tested to determine if the loop should continue or stop.

Do Loops (6.1) (cont.)

- Post Test Loop

```
Do
   statement(s)
Loop Until condition
```

Loop is executed once and then the condition is tested. If it is False, the loop is run again. If it is True, the statements following the Loop statement are executed.

Do Loops (6.1) (cont.)

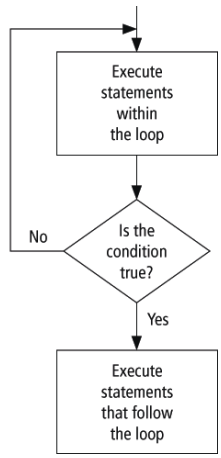
- Example 3

```
Do
  passWord = InputBox("What is the password?")
  passWord = passWord.ToUpper
Loop Until passWord = "SHAZAM"
```

Do Loops (6.1) (cont.)

✶Pseudocode and Flowchart for a Post-Test Loop

```
Do
  statement(s)
Loop Until condition is true
```



Do Loops (6.1) (cont.)

✶Comments

- Be careful to avoid **infinite** loops – loops that never end.
- VB.NET allows for the use of either the **While** keyword or the **Until** keyword at the top or the bottom of a loop.
- (This text will use only While at the top and only Until at the bottom.)

Processing Lists of Data with Do Loops (6.2)

- Display all or selected items from lists
- Search lists for specific items
- Perform calculations on the numerical entries of a list

Processing Lists of Data with Do Loops (6.2) (cont.)

- Terminology
  - Counters calculate the number of elements in lists
  - Accumulators sum numerical values in lists
  - Flags record whether certain events have occurred
  - Peek method can be used to determine when the end of a text file has been reached

Processing Lists of Data with Do Loops (6.2) (cont.)

✶Peek Method

– Data to be processed are often retrieved from a file by a Do loop

– To determine if we have reached the end of the file from which we are reading, use Peek

Processing Lists of Data with Do Loops (6.2) (cont.)

✶Peek Example

– a file has been opened as a StreamReader object named *s*r.

– **`sr.Peek`** is the ANSI value of the first character of the line about to be read with ReadLine. If the end of the file has been reached, the value of **`sr.Peek`** is -1

Processing Lists of Data with Do Loops (6.2) (cont.)

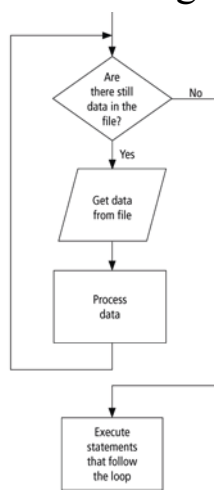
✶ Example 1

```
Dim sr As IO.StreamReader = _
    IO.File.OpenText("PHONE.TXT")
lstNumbers.Items.Clear()
Do While sr.Peek <> -1
  name = sr.ReadLine
  phoneNum = sr.ReadLine
  lstNumbers.Items.Add(name & " " _
                       & phoneNum)
Loop
sr.Close()
```

Processing Lists of Data with Do Loops (6.2) (cont.)

✶ Pseudocode and Flow Chart for Processing Data from a File

```
Do While there are still data in the file
   Get an item of data
   Process the item
Loop
```



Processing Lists of Data with Do Loops (6.2) (cont.)

✶Example 2

```
Do While (name <> txtName.Text) _
      And (sr.Peek <> -1)
   name = sr.ReadLine
   phoneNum = sr.ReadLine
Loop
```

As long as the name being searched for has not been found AND the end of the file has not been reached, the loop will continue

Processing Lists of Data with Do Loops (6.2) (cont.)

✶Counters and Accumulators

– A counter is a numeric variable that keeps track of the number of items that have been processed.

– An accumulator is a numeric variable that totals numbers.

Processing Lists of Data with Do Loops (6.2) (cont.)

✶Example 3

```
numCoins = 0
sum = 0
Do While sr.Peek <> -1
   coin = sr.ReadLine
   numCoins += 1
   sum += CDbl(coin)
Loop
```

Sum is an accumulator. It is used to total up the values of the coins.

numCoins is a counter, it increases by 1 each time through the loop

Processing Lists of Data with Do Loops (6.2) (cont.)

✶Flags

– A flag is a variable that keeps track of whether a certain situation has occurred.

– The data type most suited to flags is Boolean.

Processing Lists of Data with Do Loops (6.2) (cont.)

✶Example 4

```
Do While (sr.Peek <> -1)
  word2 = sr.ReadLine
  wordCounter += 1
  If word1 > word2 Then
    orderFlag = False
  End If
  word1 = word2
Loop
```

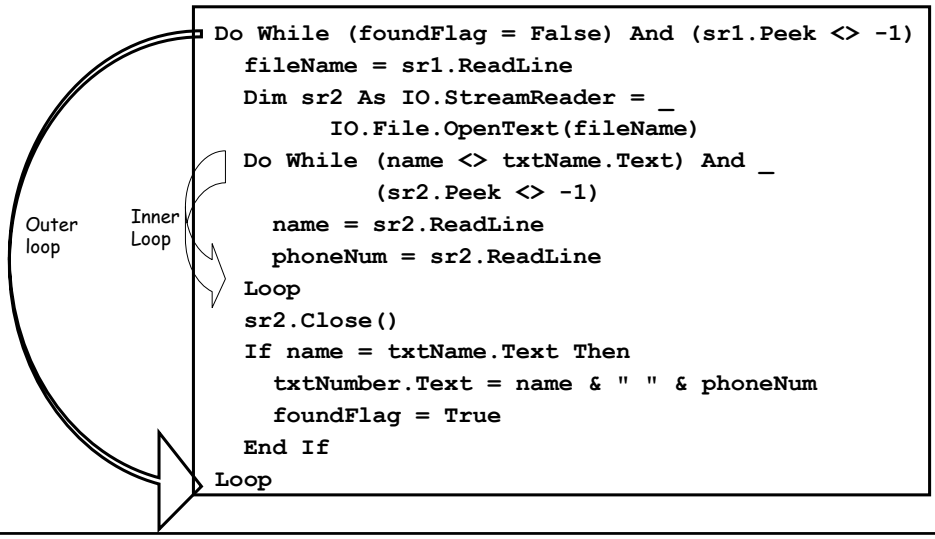
Processing Lists of Data with Do Loops (6.2) (cont.)

✶Nested Loops

– Statements inside a loop can contain another loop.

Processing Lists of Data with Do Loops (6.2) (cont.)

✶ Example 5




```
Do While (foundFlag = False) And (sr1.Peek <> -1)
   fileName = sr1.ReadLine
   Dim sr2 As IO.StreamReader = _
         IO.File.OpenText(fileName)
   Do While (name <> txtName.Text) And _
            (sr2.Peek <> -1)
     name = sr2.ReadLine
     phoneNum = sr2.ReadLine
   Loop
   sr2.Close()
   If name = txtName.Text Then
     txtNumber.Text = name & " " & phoneNum
     foundFlag = True
   End If
Loop
```

Processing Lists of Data with Do Loops (6.2) (cont.)

✶ More About Flags

– When *flagVar* is a variable of Boolean type, the statements

```
If flagVar = True Then
```

and

```
If flagVar = False Then
```

can be replaced by

```
If flagVar Then
```

and

```
If Not flagVar Then
```

Processing Lists of Data with Do Loops (6.2) (cont.)

- Flags continued
  - The statements

```
Do While flagVar = True
```

and

```
Do While flagVar = False
```

can be replaced by

```
Do While flagVar
```

and

```
Do While Not flagVar
```

For…Next Loops (6.3)

- Used when we know how many times we want the loop to execute
- A counter controlled loop

For…Next Loops (6.3)

✶Sample

```
For i = 1 To 5
  lstTable.Items.Add(i & " " & i ^ 2)
Next
```

The loop control variable, i, is

- Initialized to 1
- Tested against the stop value, 5
- Incremented by 1 at the Next statement

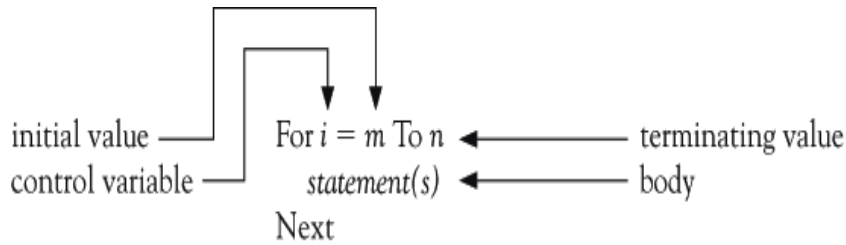
For…Next Loops (6.3)

✶Do While equivalent

```
i = 1
Do While i <= 5
  lstTable.Items.Add(i & " " & i ^ 2)
  i += 1
Loop
```

For…Next Loops (6.3)

✶For…Next Loop Syntax



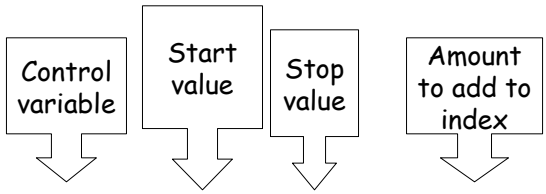
For…Next Loops (6.3)

✶Example 1

```
lstTable.Items.Clear()
For yr = 2002 To 2006
  lstTable.Items.Add(String.Format _
      (fmtStr, yr, pop))
  pop+= 0.03 * pop
Next
```

For…Next Loops (6.3)

✶Example 2




```
For index = 0 To n Step s
  lstValues.Items.Add(index)
Next
```

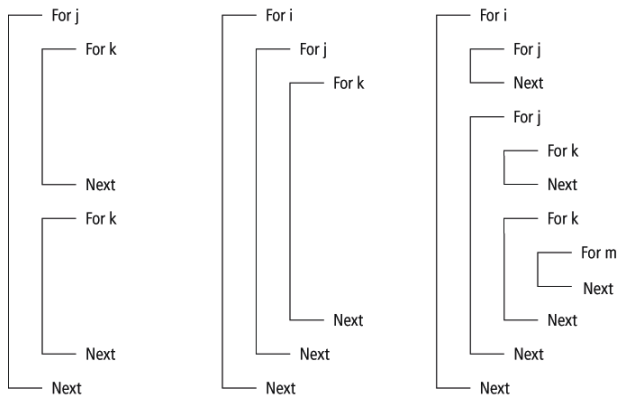
For…Next Loops (6.3)

✶Example 3

```
For j = m-1 To 0 Step -1
  temp &= info.Substring(j, 1)
Next
```

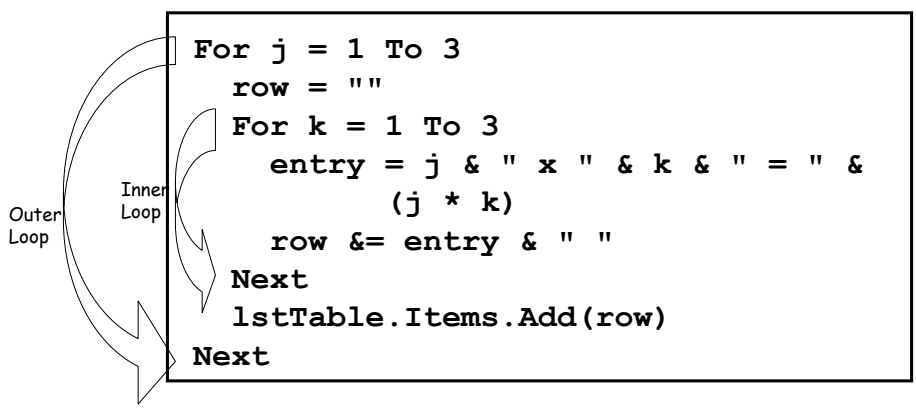
For…Next Loops (6.3)

✶Nested For…Next Loops



For…Next Loops (6.3)

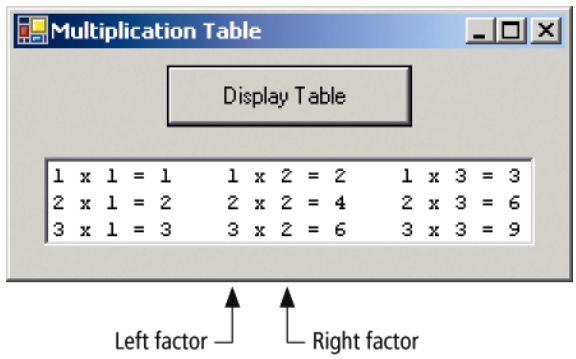
✶Example 4




```
For j = 1 To 3
   row = ""
   For k = 1 To 3
      entry = j & " x " & k & " = " &
              (j * k)
      row &= entry & " "
   Next
   lstTable.Items.Add(row)
Next
```

For…Next Loops (6.3)

✶Example 4 Output



For…Next Loops (6.3)

✶For and Next Pairs

– For and Next statements must be paired.

– If one is missing, the automatic syntax checker will complain with a wavy underline and a message such as

"A 'For' must be paired with a 'Next'."

For…Next Loops (6.3)

✶Start, Stop, and Step values

- Consider a loop beginning with
  For *i* = *m* To *n* Step s.
- The loop will be executed exactly once if *m* equals *n* no matter what value *s* has.
- The loop will not be executed at all if *m* is greater than *n* and *s* is positive,
- or if *m* is less than *n* and *s* is negative.

For…Next Loops (6.3)

✶Altering the Control Variable

- The value of the control variable should not be altered within the body of the loop;
- doing so might cause the loop to repeat indefinitely
- or have an unpredictable number of repetitions.

For…Next Loops (6.3)

✶Non-integer Step Values

– Can lead to round-off errors with the result that the loop is not executed the intended number of times.

For…Next Loops (6.3)

✶Non-integer Step Example

– A loop beginning with

```
For i = 1 To 2 Step .1
```

will be executed only 10 times instead of the intended 11 times. It should be replaced with

```
For i = 1 To 2.01 Step .1
```